Internet and Web Programming

Study Material for MS-18

Directorate of Distance Education
Guru Jambheshwar University of Science & Technology, Hisar

Study Material Prepared by

R. K. Jaiswal

Published by

EXCEL BOOKS

A-45, Naraina, Phase-I,

New Delhi-110 028

Published by Anurag Jain for Excel Books, A-45, Naraina, Phase I, New Delhi-110 028 and printed by him at Excel Printers, C-206, Naraina, Phase I, New Delhi - 110 028

CONTENTS

UNIT

1

INTERNET FUNDAMENTALS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe CGI
- Understand how CGI works
- Understand how to pass data from forms to server scripts
- Describe Internet hardware and software
- Define ISP and Internet Account
- Describe Web Browsers
- Describe URL and its various types
- Describe Internet Security
- Understand how to connect to Telnet
- Describe E-mail and its applications
- Describe Gopher
- Describe search engines, its various types and usage
- Describe plug-Ins and Helper programs

U N I T S T R U C T U R E

1.1 INTRODUCTION

INTERNET is the revolutionary phenomenon heralding the dawn of a new era in human civilization- the Information Age. With over 30 million computers connected to it and the number growing exponentially doubling in size approximately every 10 months, its potential for affecting every facet of life, be it communication, education, entertainment, business and government is mind-boggling.

Nowadays, most of the hype about the Internet is focused on the World Wide Web. It has existed for less than 10 years but it has become the fastest growing and most popular part of the Net. The Web is an interface-a window on the Net. It wraps most of the different features of the Internet into a single interface used by Web applications.

1.2 CGI

For some Web applications, you may need programs talking to each other at both locations: on the Web page and one the Web server. If the Web page program is a Java applet, it is even possible to make your Web page communicate with an e-mail or chat server instead of the Web server.

Web pages run your server programs through a server feature called the Common Gateway Interface (CGI). The programs that run on the server under this feature are called CGI scripts for historical reasons (not because they have anything to do with Web page script languages such as JavaScript or VBScript). CGI scripts are generally written in the languages of either C or Perl, although other languages are also used to create the executable script file.

Creating a CGI script for a Web server is not a simple task. Most Internet service providers require that you submit such scripts to them and pay for them to review it before you (or they) install it on the server. This precaution is to make sure that the script will not cause problems for others. For most Web page developers, attractive alternatives include using pre-designed or pre-installed scripts that their Webmaster has already approved and has provided easy access to. Another alternative is to buy special tools that create the scripts for you.

How CGI Works

How do CGI programs operate? The easiest way to envision a CGI program is to imagine that it is part of a Web server. The browser sends a request to the server. If the requested URL corresponds to a CGI program, the server starts the appropriate program and passes to the program a copy of the request. The server then sends the output from the CGI program back to the browser in the form of a reply.

Interestingly, from a browser's point of view, there is no difference between a URL that corresponds to a static document and one that corresponds to a CGI program. Requests for both static documents and CGI output have the same syntactic form. Similarly, all replies that a browser receives from a server have the same form – there is nothing to tell the browser whether the server is returning a copy of a static document or the output from a CGI program.

Professional Programmers Build

CGI Programs

Who builds CGI programs? Only someone who knows how to write a computer program can create a CGI program. Thus, the technology is beyond the capabilities of most users. More important, CGI programs are somewhat unusual because they interact directly with a Web server. The CGI program must be written to follow the server's guidelines. For example, unlike a conventional computer program, a CGI program does not write output on a user's screen.

Instead, a Web server captures the output from a CGI program and sends it back to the browser. Similarly, a CGI program does not receive input from a keyboard or a mouse - the browser handles all interaction with the user. Interestingly, the interaction between a Web server and a CGI program also depends on the computer's operating system. Therefore, like conventional programs, each CGI program is written for one brand of Web server and for one brand of computer operating system. Before the CGI program can be used on a different computer system, it may need to be rewritten.

CGI and Advertising

Corporations quickly realized that CGI technology can be used to enhance advertising. Recall that frame technology makes it possible to separate advertisements from the Web pages themselves. A CGI program can store information about previous contacts from a given browser, and use the information when selecting an advertisement. That is, whenever it receives a request for a page, the CGI program can select an advertisement to place in one frame, while sending the requested page in another. As a result, a user will see a different advertisement each time he visits the corporate site. CGI programs can also keep a record of which corporate Web pages a user visits and choose advertisements that suit the user. If the user has browsed pages about furniture and appliances, the CGI program might choose to include ads appropriate for someone who is furnishing a house. If a user browses pages of popular music, the CGI program might choose to include advertisements about events such as upcoming rock concerts.

Web Pages can Interact

Although the CGI technology discussed above can be used to create Web pages that change, CGI programs run only at the server. Thus, a CGI program cannot interact directly with a user. To make it possible for a user to enter data, another technology was invented. Known as FORMS, the technology permits a Web page to contain blank areas in which the user must enter information. After a user fills in the required information, the browser sends the information to the server when requesting another page.

The advantage of forms technology should be clear: instead of merely selecting items from a list, FORMS make it possible to enter data directly. For example, suppose a user wishes to purchase an item over the Web. The user must supply a credit card number and a postal address to which the item should be mailed. Without FORM technology, entering such information is almost impossible. With FORM technology, a single Web page can be displayed for the user that contains a form for the credit card number and another form for the mailing address. Once the user has filled in the forms and clicked on a selectable item (e.g., an item labeled purchase), the browser sends the information to the server.

Active Documents are More Powerful

Newer, more powerful Web technologies have been developed to help solve the problems of providing smooth motion and animation. The new technologies take an entirely different approach by making documents active. An active document is a computer program that paints an image on a screen. Active documents are stored on Web servers, and each active document is associated with a URL just like conventional Web pages. Furthermore, retrieval works as usual; a browser receives a copy of an active document when it requests the document's URL just like it receives a copy of a conventional Web page. Once a browser receives a copy of an active document, the browser runs the program on the local computer. The running program paints an image on the browser's screen.

How Do You Make Web Pages Run Server Scripts?

To write a Web page command that runs a server script, you must know a few things about that script. Pages that run server scripts refer to the scripts by the script's URL. To run a server script, therefore, you must know the script's URL: its name and the sub-directory where it is located on your Web server. (Note that it must be your Web server – the server the Web page comes from. You are not usually permitted to run a script on a different server.)

You must also know what kind of data the script wants passed to it, and what method must be used for passing the data. There are two basic ways of running a CGI script located at a particular URL, depending on how that script requires data to be passed to it. (The way you pass data is officially called the method.) The two methods of passing data that a script may use are as follows:

- GET for transmitting a line of data
- POST for transmitting data from forms

S0ome scripts require one particular method to be used. Others can use either method. The GET method in its simplest form uses a conventional FIREF (hyperlink reference) tag in HTML, but

instead of another Web page's URL, it uses the script's URL. Following is an example of a link you click to run a program using the GET method. Although this example runs a script when the user clicks a link, your page could alternatively use JavaScript to run a script. In this hypothetical example, the script receives data about what Canadian province is being discussed and what language is to be used for script output.

<A HREF="/cgi-bin/myscript.bin?provinc=quebec&language=french"> Click here to run the script.<P></A>

The URL is cgi-bin/myscript.bin. (The script file's name is myscript.bin and it is in the cgi-bin directory of the Web server.) Not only is the URL given, but in the same line, following a question mark symbol (?), values (parameters) are passed to the CGI script for it to act upon. The parameters in the example are named province and language, and they are given the values quebee and french, respectively, in this case. The ampersand (&) separates the two values.

The CGI script and the HTML code that runs the script must both be designed to work together, as they both must use the same method and variables. To read the data in the example, for instance, the CGI script must know or ascertain that the GET method was used to transmit the data. It then looks within a special variable (an environment variable) reserved for this purpose that usually has the name QUERYSTRING (or QUERY-STRING) to find the data that was passed. The data is passed as a single string of characters, so the script must separate (parse) the data. It does so by looking for the variables by their names (in this example, pros-ince and language) or by their position in the string, and reading whatever data follows each variable's equal sign (=).

Student Activity 1

1. Define CGI.
2. How does CGI works ?
3. How can CGI technology enhance advertising ?
4. Define FORMS. Describe its usage.
5. How do you make web pages run server scripts?

1.3 HOW DO YOU PASS DATA FROM FORMS TO SERVER SCRIPTS

To pass data from forms to server scripts, you use special attributes within the various form tags. Following is an example of the HTML for a Web page in which a form is used. The user fills out a form and then clicks a button labeled "Submit" to run the CGI script and transmit form data to it.

```
<HTML>
<HEAD>
<TITLE>Guest Book</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1 ALIGN="CENTER">Guest Book</H1>
<HR>
<FORM ACTION="/scripts/perl-cgi/guest.pl" METHOD="POST">
First Name :<INPUT TYPE="text" NAME="fn" SIZE="24" MAXLENGTH="30"><BR>
Last Name :<INPUT TYPE="text" NAME="ln" SIZE="24" MAXLENGTH="30"><BR>
City : <INPUT TYPE="text" NAME="fn" SIZE="24" MAXLENGTH="30"><BR>
State :<INPUT TYPE="text" NAME="fn" SIZE="24" MAXLENGTH="30"><BR>
```

```
Email Address :<INPUT TYPE="text" NAME="fn" SIZE="24"
MAXLENGTH="30"><BR>

Comments :<TEXTAREA NAME="comments" COLS=36" ROWS="7"></TEXTAREA><BR>

<INPUT TYPE="submit" VALUE="Submit">

<INPUT TYPE="reset" VALUE="Clear Form">

<P>

</FORM>

</CENTER>

</BODY>

</HTML>
```

Internet Hardware and Software

A modem (short form of modulator-demodulator) is a device, which is used to convert digital signals to analog signals so that they can be transferred over the standard telephone line. At the receiving end another modem is connected which reconverts the analog signals back to the digital signals (Figure 1.1).

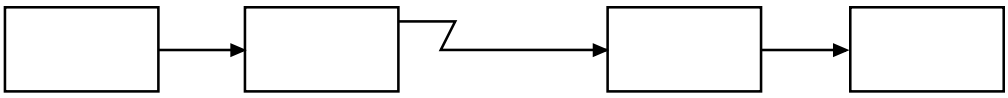


Figure 1.1 Modulation-Demodulation Process

Digital signals are converted to analog signals by the first modem so that information can be transferred over analog telecommunication lines. At the receiving end, it is reconverted by the second modem to digital signals so that the message is available in its original form. In order to connect to the Internet, you have to first connect your computer system with a modem, which in turn helps to establish a connection to the Internet with the help of telephone lines.

Factors to be considered while buying a modem

While buying modems the following factors are to be taken into account :

(a) Transmission speed

(b) Data compression schemes

(c) Error Correction Protocols

(d) Type of Modem, i.e., Internal vs. External

Transmission Speed

Transmission speed is the speed with which the data is transmitted. It is measured in bits per second (bps). Modems come in different speeds like 300 bps, 1200 bps, 2400 bps, 4800 bps, 9600 bps, 14,400 bps (14.4 kbps) and 28,800 bps (28.8 kbps). One kbps (1 kilo bits per second) stands for 1000 bits per second. The faster the transmission speed of the modem, higher will be its cost, the advantage being that data is transferred at faster speeds. This implies that if the transmission channels permit these speeds then the modem will attain that speed, otherwise it will depend on the speed of the media. These days modems of speeds 28.8 kbps, 33.6 kbps, and 56 kbps are gaining popularity.

Data Compression Schemes

Since voluminous data is to be transferred, data has to be compressed. In order to bring about standardization of the compressed codes generated by different modems various standards have evolved.

- V.32 standard has achieved a transmission speed of 9,600 bps.
- V.32 b transmits data at a speed of 14,400 kbps.
- V.34 standard allows a transmission speed of upto 28,800 bps.
- V.42 provides four-fold data compression.

Error Correction Protocols

Since data travels over telephone lines many errors can be introduced in the signal because of noise, which also travels in waves along with the data. Therefore, error correction protocols are used to free data from errors. Some popular systems are MNP (Microcom Networking Protocol) and V. 42. Examples of error-correction protocols of MNP are MNP2, MNP3, MNP4. V.42 is a standard for error correction based on Link Access Procedure for Modem (LAPM) standards. These standards are available in all modems so that the user is not troubled.

Types of Modems

There are two types of modems (Figure 1.2)

Figure 1.2 Types of Modems

1.1:

1. 2. 3. 4. 5.	1. 2. 3. 4. 5.

Cable modems (subscriber unit)

A Cable Modem is a device that enables you to attach your PC to a local cable TV line and receive data at approximately 3.5 mbps. The distance can be 100 kms or even more. This is much faster than the dial-up connection (56 kbps) and the ISDN connection (128 kbps). When a cable modem unit is installed next to your computer, a splitter is also placed. It separates the coaxial cable line serving the cable modem from the line that serves the TV set. A separate coaxial cable line is connected from the splitter to the cable modem.

A cable modem has two connections. One to the cable wall outlet, and the other to a PC or to set-top box for a TV. A set-top box is a device that enables a television set to become a user interface to the Internet and also enables the television set to receive, and decode digital television broadcast.

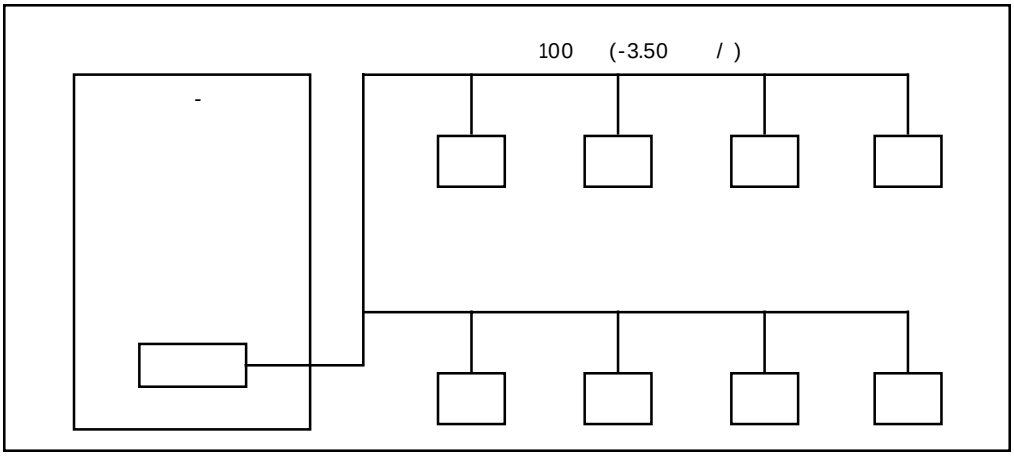


Figure 1.3 Cable Modem System

Typically cable modem is connected to a standard 10BASE-T (10 refers to transmission speed of 10 mbps, BASE refers to baseband signalling and T refers to twisted pair cabling) with Ethernet (LAN) card in the computer. A wire called "Category 5 cabling" (Cat 5 cabling) with connects the cable modem to the Ethernet card. Data transmission rate between the cable modem and the computer is 10 mpbs. Basically you connect the cable modem to the TV outlet for your cable TV and the cable TV operator connects a Cable Modem Termination System (CMTS) at his end (Head-End) (Figure 1.2). CMTS is a central device for connecting the Cable TV Network to a data network like the Internet through the backbone of a cable Internet Service Provider. The CMTS can talk to all cable modems (CMs) but the cable modem can only talk to the CMTS. Now if two cable modems have to talk then the CMTS will have to relay the messages.

Data Over Cable Service Interface Specification (DOCSIS) is a cable modem standard that defines technical specifications for both cable modems and CMTS.

Student Activity 2

1. How do you pass data from forms to server scripts ?
2. What is a modem ?
3. Which factors are to be considered while buying a modem ?
4. Describe various types of modems.
5. What are cable modems ?

1.4 ISP AND INTERNET ACCOUNT

An Internet Service Provider (ISP) is an organization or business offering public access to the Internet. It is your gateway, to the Net. You have to subscribe to a provider for your Internet connection. You use your computer and modem to access the provider's system and the provider handles the rest of the details of connecting you to the Internet.

There are many types of Internet providers. You can, for instance, choose one of the big commercial on-line service providers. The primary business of an ISP is hooking people to the Internet by giving an Internet account to subscribers, and providing them with two different kinds of access: shell access and SLIP/PPP access. Most ISPs offer both kinds of access, some offer both with a single account, and others require that you choose one or the other. Once you register, your provider will give you a user name (called a userid), a password, and a phone number to dial. To establish the Internet connection, you have your communications program dial the number. You then log in using your particular userID and password.

At present it is VSNL (Videsh Sanchar Nigam Limited) which is dominating the Internet scene in India through its GIAS (Gateway Internet Access Service). The other service providers in India are MTNL (Mahanagar Telephone Nigam Limited), Mantraon-line, and Satyam online.

Most probably it would be present as a service provider in your city, if you are lucky enough to dwell in an Indian city like Delhi, Mumbai, Chennai, Calcutta, Pune, Ahmedabad, Lucknow, Bangalore, etc. It is also available to you in various other cities through RABMN (Remote Area Business Management Network) and GPSS (Gateway Packet Switching Service), our older Indian networks, where you will really have to inquire yourself from VSNL or DOT for availability and type of connectivity. For many institutions and organizations, Internet access is available through ERNET, our old, but rich and reliable Educational and Research Network. NIC also provides Internet facility through their NIC-NET network.

Choosing an ISP

The privatization of Internet Service Providers (ISPs) is set to give a further fillip to the Internet boom. Central to the success of any service is the price criterion. You will be amazed to find out how a service offered at a premium could in effect be cheaper, considering the add-on facilities that are offered along with the core service. Do not forget that apart from the Internet connection, the ISP gives you an international contact address, that is, your E-Mail address. It is because of this E-Mail address that you must be discerning while choosing your ISP. The E-Mail address provided by the ISP would be all over your business, and it will not be easy for you to change your service provider if you wish to change your address. You will have to live with the ISP as well as the E-Mail address. So, be judicious while making the choice, just as you would while choosing your life partner.

User-to-Telephone Ratio

The first thing you must keep in mind while zeroing in on your ISP is the user-to-line ratio it commands. That is, how many users are using or are expected to use one single telephone line. Ascertaining this, however, is not easy as the number of subscribers are growing every day. Nevertheless, even the current user-to-line ratio will give you an idea about the standards the ISP has set for itself. This factor is very critical because it determines the ease of usage whether you would be able to connect to your ISP or not. Another way of finding this is to check out with some of the existing users as to how much time it normally takes to dial into a given ISP. If it takes more than 10 minutes to get through, that particular ISP should be avoided.

Interface Simplicity

Very few organizations take into account the simplicity of the interface while opting for an ISP. This occurs to them only when they begin to use the Internet service across their organizations. The right kind of interface can lead to tremendous savings in cost. There are other problems too. How many users in an organization know about dial-up networking under Windows? How many can remember and use passwords correctly? To how many people would you like to give the password? Does terms like TCP/IP sound friendly to them? Questions like these determine the success of the Internet enabled organizations. There are some ISPs to whom these questions do not apply. They provide an easy-to-use interface that once installed works by simply pressing a button.

Roaming Facility

The roaming facility is particularly relevant for those who travel a lot. Though most ISPs advertize this particular facility, there are not many who pay heed to it. Its benefits are realized only when one reaches another city and wants to access an urgent E-Mail or the Internet. How does one connect to the Internet when one is not an ISP subscriber in that particular city? To overcome this problem, either you will have to use a facility like Hotmail to access your mail from around the world or use the roaming facility provided by your ISP. The roaming facility allows you to dial-in into the local node of your ISP or of the regional ISP that your service provider has a tie-up with. Then all you have to do is to plug in your computer to a telephone line, find out the numbers for dial-up access, and then using your password, access your original Internet account. A crucial point here is the number of cities that your ISP has presence in or has tie-ups for the same.

Multiple Login Facility

Very few users know about this facility, mainly because it is hardly advertized. However, it can prove to be a life-saver and a great help for small and medium business houses. If an organization has only one Internet connection, but more than one employees want to access the Net

simultaneously then this would be possible only if the ISP offers to the organization the multiple login facility. In fact, this facility can even be availed of while being away from the organization. For instance, one user may be in New Delhi and the other user in Mumbai. But, with the E-Mail ID it would be possible for the man away in Mumbai to simultaneously access the Internet. Some ISPs offer multiple E-Mail IDs that allow you to segregate E-Mail individually. But you have to pay extra for this.

ISDN Facility a boon for Corporates

ISDN services are a boon for corporates that have multiple users who need simultaneous Internet access. Mantraon-line is the first private ISP to offer the same, and at special rates. If you believe in planning for the future, and are optimistic about the utility of the Net in your organization, then maybe you should give this option a serious thought.

Special Packages

The private ISPs are putting out some unique usage packages. One of them is by Mantraon-line. It has launched a special package for night users. For those who access the Net at night, Mantra offers a dial-up account which costs almost half compared to the regular connection. This account cannot be used during day time. This is only the beginning as far as special packages are concerned. Soon you will find ISPs (especially the regional ones) coming out with packages that will fit your needs better than your cotton trousers. Already, VXL has launched India's first cable modem ISP service in Bangalore. So do not forget to check out each and every player before deciding on your Internet partner.

Support

This is a very crucial topic and an area of service where most of the players have been found wanting. Try getting any help from the service provider and the beautifully programmed EPABX system will take you around each and every option, only to disconnect your call at the end stating: "Sorry, the person handling your call is busy at the moment." In case, you happen to be using a pulse-dialling equipment, you can forget using the telephone, and may as well go to their office and clear out the matter there and then.

Ideally, new users should subscribe to an ISP where they can be hand-held through the initial process, as Bill Gate's Windows operating system does try its best to support you in the exercise. An installation guide, the help desk's phone number, Windows 95 installation CD are part of the necessary survival kit that a new user must have while undergoing this not-so-holy procedure.

Discounts on Renewal

Last but not the least, you must find out whether your ISP will renew your account at the same rate or whether there are any discounts to retain its old customers? This is a factor that can upset those lining for their first-buy. VSNL has been very successful in playing this card. It offers slashed rates to those subscribers who renew their accounts.

Brochure-speak

If you can have more than a hundred different versions of the holy Ramayana, just think what the crafty marketing people can do to simple terms of the Internet. Hence, one must see through the exotic looking tariff cards of most ISPs. You must have the ability to judge beyond the gloss and the glitter. To summarize, here is what you want from an Internet Service Provider:

- Access via a local phone call
- a flat monthly fee
- An ISDN or fast (28.8 kbps) connection
- A PPP account
- A shell account at no extra charge
- The ability to use whichever Internet clients you want
- Full Internet access to all resources

- The capability of having your own web home page
- Software support, through which you can use to connect to and use the Internet
- Technical support should be open 24 hours a day, 7 days a week

Internet was available for some time through ERNET and it was made available for commercial use by VSNL since August 1995. Presently VSNL operates in the following six metres:

– New Delhi

– Chennai

– Calcutta

– Mumbai

– Bangalore

– Pune

The membership and networking rules are controlled by VSNL. VSNL has its own leased lines to USA where the Indian network is connected to the main Internet network. VSNL offers three types of membership to any user to be connected to Internet. The three types of Internet accounts are:

- Shell account
- TCP/IP account
- Web-site or Homepage

Shell Account

It is usually subscribed by students and other members who want to access technical information in any branch of knowledge. The information is available only in text mode and no graphics and pictures are available. One can send E-mail to others on Internet, obtain information about latest research papers of various universities, search for product information, having on-line discussion, float questions to be answered by others, find out the latest news on any topic, find out tourist information like train, air schedule, interested places, etc.

Shell account is more useful and cheaper. It requires very small hardware set-up and the membership is economical. The membership fee are Rs. 500 for students and Rs. 5000 for a professional per year for 500 hours or whichever is earlier and Rs. 25,000 for a company.

TCP/IP Account

TCP/IP account provides the same advantages to that of shell account plus and it provides unlimited access to Internet. It requires UNIX-Based machine, 1-2 GB of Hard-disc, 16-32 MB of RAM and SVGA/VGA monitors and TCP/IP software for better results.

It provides an access to graphics, games, movies and other multimedia products along with text chapters, The membership fee for this account to an educational institution is Rs. 15,000 and Rs 500 for registration.

Homepage or Website

Website makes a user as a node of Internet. This means that if a university decides to float its own Website, then all information regarding admissions, scholarships, rules and regulations, courses it offers, department, staff and their profiles, workshops, conferences, seminars and symposia to be conducted, etc., can be kept at University's computer centre. This computer can be accessed by any member on Internet for any particular course; it will be automatically listed with all its details. This Website is important to a university or a company to offer 24-hours service and on-line help to international clients.

Developing a Website on Internet is a costly affair as far as hardware and other costs are concerned. There are no set of rules and regulations for granting Website by VSNL. Homepages can have

text, graphics, pictures and animation. Some homepages can add sound and give multimedia capabilities. Any access to a homepage is called a "hit". The more popular the homepage, the more hits it will receive. Homepages can be grouped together in any order and be hyperlinked.

Student Activity 3

1. What is an ISP ? Name some ISPs available in India.
2. What are the factors to be considered for choosing an ISP ?
3. Define shell account.
4. Define TCP / IP account.
5. Define home page.
6. Define a Website.

1.5 HTTP AND URLs

If you have ever noticed an advertisement that says "check out our Web site", you must have seen the way they describe how to find it, giving you the address, or *URL*, which almost begins with the letters *http*:. (URL stands for Uniform Resource Locator). Within this system, there is a unique URL for any hypertext item on the Net. Moreover, there are also unique URLs for non-hypertext items from other services, such as gophers, anonymous ftp sites, Usenet newsgroups and wais databases. You see lots of URLs in newspapers and magazines these days. These URLs are basically "pointers" to documents, movies, photos, and so on, located on computer somewhere. You may want to visit those sites using URLs. If it's out there, you can get to it.

Let's look at an example:

http:// www.uptecnet.com/news.htm#yaksha

In this example, the URL has the following elements:

- The first part of URL specifies the *transfer protocol*, the method that a computer uses to access this file. Most Web pages are accessed with the HTTP which is why Web addresses typically begin with http. The http:// at the beginning of a Web page's URL is so common that it often goes without saying; if you simply type *uptecnet.com* into the address window of Internet Explorer or Navigator, the browser fills in the http:// for itself. In common usage, the http: // at the beginning of a URL often is left out. Since the Web is not the only Internet service, you may occasionally see addresses that start with ftp:, gopher:, telnet:, news:, or some other name. The protocol is followed by a colon.
- The *www.uptecnet.com* part of the address — called a domain name — is preceded by two slashes. This is the name assigned to a computer located somewhere in the world that is permanently connected to the host name of the Web server. In the name, *www* stands for World Wide Web, *uptecnet* is the name of the company or organization hosting the site, *.com* is the identification of the type of organization hosting the site — called the topmost domain. The Web server is invisible from the URL that means URL does not tell you where the Web server is actually located. It is the job of DNS (Domain Name System) to route your Web page request to the Web server regardless of its physical location.
- The *news.htm* is the path to a file — it is saying that the document *news.htm* is stored in folder named / (root).
- The *# Yaksha* part of the name, called a fragment ID, is the name of a specific part of the document.

1.6 WEB BROWSERS

To access the World Wide Web, you use what is called a Web browser (already discussed in the previous chapter). Browsers are sometimes also called Web clients, since they get information from a server. When you start a WWW browser or follow a hyperlink, the browser (acting like a client) sends a request to a site on the Internet. That site (acting like a server) returns a file which

the browser then has to display. In order for you to see or hear what's in the file, the browser should be capable of interrupting its contents. This differs depending on the type of file, text, graphics and/or images that may be displayed. If the file is written using HyperText Markup Language (HTML), the browser interprets the file so that graphics and images are displayed along with the text.

Basic Features of Browsers

Before we get involved in all the details, let is discuss some important browser features.

1. The Web browser should be able to look at the Web pages throughout the Internet or to connect to various sites to access information, explore resources, and have fun.
2. The Web browser must enable you to follow the hyperlinks on a Web page and also to type in a URL for it to follow.
3. Another feature of browser is to have a number of other commands readily available through menus, icons, and buttons.
4. Your browser ought to include an easy way to get on-line help as well as built-in links to other resources on the Web that can give you help or answers to your questions.
5. You will definitely want a way to save links to the sites you have visited on the WWW so that you can get back to them during other sessions. Web browsers take care of those in two ways, through a history list, which keeps a record of some of the Web pages you've come across in the current session, and a bookmark list, which you use to keep a list of WWW pages you want to access any time you use your browser. The name of the site and its URL are kept in these lists. The bookmark list is particularly important and the browser will contain tools to manage and arrange it.
6. One of the main feature of a browser is to search the information on the current page as well as search the WWW itself.
7. Browsers give you the facility to save a Web page in a file on your computer, print a Web page on your computer, and send the contents of a Web page by e-mail to others on the Internet.
8. Few Web browsers (like Netscape Communicator) are complete Internet package, means they come with components like e-mail client, newsgroup client, an HTML composer, telnet client, ftp client, etc.
9. Web browser should be able to handle text, images of the World Wide Web, as well as the hyperlinks to digital video, or other types of information.
10. To take advantage of some of the most exciting things on the World Wide Web, your browser needs to properly display and handle Web pages that contain animated or interactive items. Netscape Navigator can incorporate these features through its ability to interpret programs written in Java and Java Script.
11. Web browsers interact not just with the Web, but also with your computer's operating system and with other programs, called plug-ins, that gives the browser enhanced features.
12. Another important feature to insist on in your browser is caching. A browser that caches keeps copies of the pages you visit so that it does not have to download them again if you want to return to them. Reloading a page from the cache is much quicker than downloading it again from the original source.
13. The most important feature of any browser is ease of use. While all Web browsers are fundamentally simple to use, the one you settle on should be very easy to work with; it should function as a transparent window onto the Web.
14. If you will be browsing the Web from within a secured network, you may have to configure your browser to work through a special computer on your network called a proxy server. Most popular browsers let you configure them to work with a proxy server, but some don’t, so find out if you will be working through a proxy before deciding on your browser. If you are, your ISP or system administrator will tell you if you need to do anything special to use your browser.

1.7 URL

If you have ever noticed an advertisement that says "check out our Web site", you must have seen the way they describe how to find it, giving you the address, or *URL*, which almost begins with the letters *http*:. (URL stands for Uniform Resource Locator.) Within this system, there is a unique URL for any hypertext item on the Net. Moreover, there are also unique URLs for non-hypertext items from other services, such as gophers, anonymous ftp sites, Usenet newsgroups and wais databases. You see lots of URLs in newspapers and magazines these days. These URLs are basically "pointers" to documents, movies, photos, and so on, located on computer somewhere. You may want to visit those sites using URLs. If it's out there, you can get to it.

Let's look at an example:

http:// www.uptecnet.com/news.htm#yaksha

In this example, the URL has the following elements:

- The first part of URL specifies the *transfer protocol*, the method that a computer uses to access this file. Most Web pages are accessed with the HTTP which is why Web addresses typically begin with http. The http:// at the beginning of a Web page's URL is so common that it often goes without saying; if you simply type *uptecnet.com* into the address window of Internet Explorer or Navigator, the browser fills in the http:// for itself. In common usage, the http: // at the beginning of a URL often is left out. Since the Web is not the only Internet service, you may occasionally see addresses that start with ftp:, gopher:, telnet:, news:, or some other name. The protocol is followed by a colon.

- The *www.uptecnet.com* part of the address — called a domain name — is preceded by two slashes. This is the name assigned to a computer located somewhere in the world that is permanently connected to the host name of the Web server. In the name, *www* stands for World Wide Web, *uptecnet* is the name of the company or organization hosting the site, *.com* is the identification of the type of organization hosting the site — called the topmost domain. The Web server is invisible from the URL that means URL does not tell you where the Web server is actually located. It is the job of DNS (Domain Name System) to route your Web page request to the Web server regardless of its physical location.

- The *news.htm* is the path to a file — it is saying that the document *news.htm* is stored in folder named / (root).

- The *# Yaksha* part of the name, called a fragment ID, is the name of a specific part of the document.

URL Anatomy

URL (pronounced U-R-ell) stands for Uniform Resource Locator, which is simply an address of a document on the Web or, more accurately, on the Internet. Although a URL can look complex and long, it is made up of four basic parts — protocols, hostname, folder name, and file name — each of which has a specific function.

Types of URLs

URLs vary depending on the location of the document to which you are linking. For example, a URL will be longer and include more information if the file is on the World Wide Web. A URL will be shorter and include less information if the file is on your local computer or server. Basically, URLs fall into two categories:

- Absolute

- Relative

An absolute URL contains all the information necessary to identify files on the Internet. A relative URL points to files in the same folder or on the same server. In other words, the file linked to is relative to the originating document.

Absolute URLs

An absolute URL contains the protocol indicator, hostname, folder name, and file name. Absolute URLs are similar to addresses used by the Indian Postal Service, which include a name, street address, apartment number (if applicable), city, state, and pin code. If some of the information is missing — say, the street number or house number — the carrier can not deliver the mail to the right person.

Here are some absolute URLs:

```
http://www.xmission.com/services/index.html

http://www.altavista.digital.com/

ftp:/ftp.raycomm.com/download/readme.txt
```

Relative URLs

A relative URL usually contains only the folder name and file name or even just the file name. You can use these partial URLs when you are pointing to a file that is logged within the same folder or on the same server as the originating file. In these cases, a browser does not need the server name or protocol indicator because it assumes that the files are located in a folder or on a server that is relative to the originating document.

Document-Relative URLs

You will often use a document-relative URL when you are developing or testing a set of HTML documents. When linking from index.html, use the following URLs:

```
services/consulting.html

services/other/tips.html
```

When linking from consulting.html to tips.html, use the following URL: other/tips.html

Server-Relative URLs

A server-relative URL is relative to the server root — that is, relative to the hostname part of the URL. Server-relative URLs have a forward slash (/) at the beginning of the file name, which indicates that you interpret the path of the document from the top of the current server (the server root), rather than from the current document location. For example, from anywhere in our site, we could use a server-relative URL to display out home page with a link to /index.html, which would display the index.html file right under the top of the server. Some server-relative URLs include:

```
/index.html

/contacts/names.html
```

Server-relative URLs are useful when you are linking to a specific location on the server (such as contact information) that is not likely to change and that is not clearly relative to the current document.

Adding Special Characters

HTML is not limited to standard ASCII characters, which do not include special characters such as trademark symbols. You can use character entities to create many other characters. Each of these extra characters has both a name and a number. To include a character entity in the text of your Web page, type an ampersand (&), the name or number of the character, and a semicolon (;). If you use the character number, precede it by#. For example, to include a copyright symbol, you can type & copy; or © in your text. Not all characters have names; some only have numbers.

The table below lists of some useful character entities; you can use the version in either the second or third column of the table (unless only one version is available). For a complete listing of all the standard character entities, see one of the following sites:

```
http://www.natural-innovations.com/boo/doc-charset.html

http://www.owlnet.rice.edu/-jwmitch/iso8859-1.html
```

When displaying text, browsers ignore repeated whitespace characters. For example, while writing an HTML program, if the Enter key is pressed several times, the browsers display only one vertical space. Similarly, if ten spaces are typed, the browsers generate only one space.

Character	Character Entity (Name)	Character Entity (Number)
Less than (<)	<	<
Less than (>)	>	>
Bullet (·)		•
Em dash (—)		–
En dash (-)		—
Trademark (ä)		™
Nonbreaking space		
Inverted exclamation point (!)	¡	¡
Copyright(Ó)	©	©
Registered trademark (Ò)	®	®
Paragraph sign (¶)	¶	¶
One-half (½)	½	½
Inverted question mark (?)	¿	¿

1.8 INTERNET SECURITY

Commercial and government enterprises are reluctant to use the Internet because of security concerns. During the past several years, attacks on routers have become frequent, with attackers realising that they can create many more problems by targeting the routing infrastructure than attacking any single system. The Internet currently uses BGP (border gateway protocol) for inter domain routing. Also, because BGP sessions use TCP to transmit data between routers, the recent increase in TCP based attacks is an additional threat to BGP security.

In the past, the Internet community used SNMP (simple network management protocol) to monitor the health of the network, and to debug operational problems. But now it is pretty easy to eavesdrop on SNMP traffic and discover the appropriate community name, and then access the SNMP database on the management network device. This is especially true when the management network device uses no authentication. There are also programs that guess passwords, trying millions of combinations of letters to find a match. Once an attacker gains access to the SNMP database, he can attack in many ways. If security relevant information, such as a password, is stored in the database, the attacker can learn the password.

The need to augment Intranet security is being alarmingly realized with the emergence of E-commerce. Presently, E-commerce operations are marked by fear of loss of money and privacy. One recent survey, undertaken by Equifex and Harris Associates, determined that over two-thirds of Internet consumers considered privacy concerns to be very important.

E-Mail Threats

E-Mail when sent across the Internet is more like a post card. It can be intercepted at any stage and read by anybody who can lay his hands on it. To ensure the secrecy of the message, the sender as well as the receiver should agree on a secret key. There starts the problem. If your intended recipient is in a far away country, then you have to distribute the key first to him before you can send him the message. And this presents a logistical problem. Public key cryptography was designed to overcome this problem through what is known as public key private key pair. Another way of ensuring the secrecy of E-Mail messages is through the use of a technique called signing a message.

Firewall – A Safe Bet

Are you planning to connect your organization to the lntemet? Do your employees dial into your computers from remote places? Do you have multiple branches connected to each other? If your

answer is yes, then Firewall is what you need to protect your lntranet. Firewall is typically defined as a system or group of systems that enforces an access control policy between two networks. It may also be defined as a mechanism used to protect a trusted network from an untrusted network. Firewall is a collection of components or a system placed between two networks.

Firewall acts as a gatekeeper between a company's internal network and the outside world. It acts as an electronic barrier to stop unauthorized entry. A firewall basically performs two important functions: gatekeeping and monitoring.

Gatekeeping by Firewall : Firewall acts as a gatekeeper between the company's internal network and the outside network. It examines the location from which the data enters your system and then decides, based on your instructions, whether or not to allow that information.

Monitoring by Firewall : In addition to gatekeeping, the firewall also monitors information. Monitoring is one of the most important aspects of firewall design. Monitoring functions include logging of all system activities and generation of reports for system administration. Monitoring can be active or passive. In active monitoring, a firewall notifies a manager whenever an incident occurs. The firewall product, Smart wall, alerts the administrator via E-Mail or a pager about suspicious on-line activity. In passive monitoring, a firewall logs a record of each incident in a file or a disk. Then a manager can analyze the log periodically to determine whether attempts to access the organization have increased or decreased over time.

Firewall Architecture

An organization that connects to the Internet over a serial line might choose to implement a firewall. Router R2 implements the outer barrier; it filters all traffic except datagrams destined for the bastion host. Router RI implements the inner barrier that isolates the rest of the corporate Internet from outsiders; it blocks all incoming datagrams except those that originate on the bastion host. Figure 1.4 contains a superfluous network that connects the two routers and the bastion host. Such a network is often called a stab network because it consists of a stubby wire to which only three computers are connected.

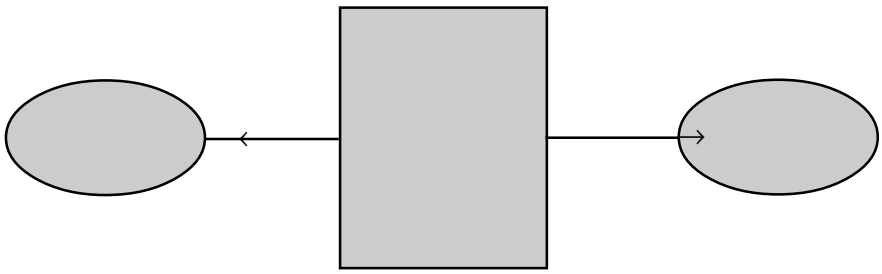


Figure 1.4 Firewall with one domain

Student Activity 4

1. What is a Web Browser?
2. Describe the basic features of a Web Browser.
3. Define URL.
4. Describe various types of URLs.
5. Why is Internet Security required?
6. What is a firewall? Describe the firewall architecture.

1.9 TELNET

Telnet is a standard Internet protocol for remote terminal connection service. By remote terminal we mean the other computer whose data/programs have to be accessed. This computer may be in the near vicinity or seven seas away. The computer on which the user works is called the local computer. In other words, Telnet is a facility by which a user can sit on a local computer and access remote computers provided he has a valid user id (user identity) and password of the remote computer or it could be that the remote computer allows public access. Telnet works on the client/server principle where the remote computer acts as a server by accepting the requests of the client (local computer), processing it and then sending the results back to the client.

Generally Telnet is used in order to retrieve:

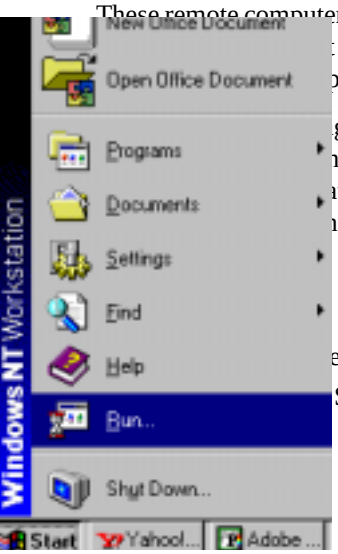
- Special information from a remote database
- Weather reports
- Library catalogs
- Departmental stores catalogs
- E-mail while touring
- Information from other Internet tools like FTP, Gopher, etc.

Getting Connected To Telnet

Generally there are two ways by which a local computer can be connected to a remote computer.

The most common way is to log on to any Internet host on which you have an account. For example, you can make a connection to a remote UNIX computer. The next thing the host would ask you to do is to enter your name and password. If the above data is correctly entered then you can access the UNIX software. In fact your computer emulates the remote terminal.

Another way to access a remote computer would be if that remote computer provides public Telnet access i.e. any netizen can access these types of remote computers from his own desktop. These remote computers do not require any password to be entered and can be directly accessed.



to find a particular book in the library catalog; you can do so provided that puter provides public access.

g 'domain access' your computer has an IP address as you are directly net, then the client application executes on the local computer. If Telnet is ation your computer is not directly on the Internet. Infact it is connected to nternet access provider's host computer. The Telnet client application runs

establish a connection to the Internet i.e. login on the Internet.

Start and then select Run (Figure 1.5). The Run dialog box appears.

Figure 1.5 The Start Menu

Step 3: Now enter telnet in the Run dialog box, and then click the OK button (Figure 1.6).

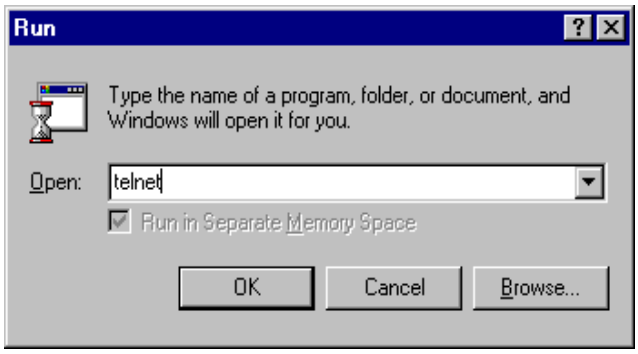


Figure 1.6 The Telnet Window

The Telnet window appears.

Figure 1.7 The Telnet Window

Step 4: Select the Connect option from the above window and the resultant is a drop down menu.

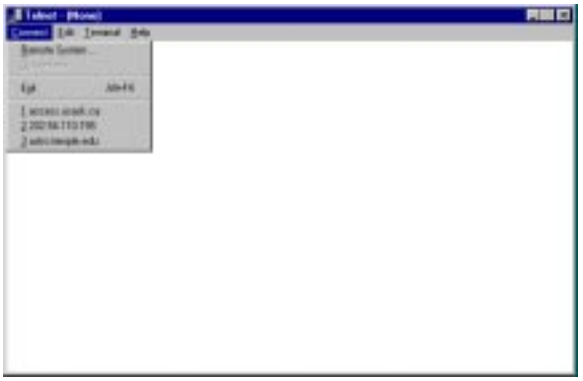


Figure 1.8 The Connect Pull down Menu

Step 5: From the Connect drop down menu, click the option Remote System. Now you will get the connect dialog box.

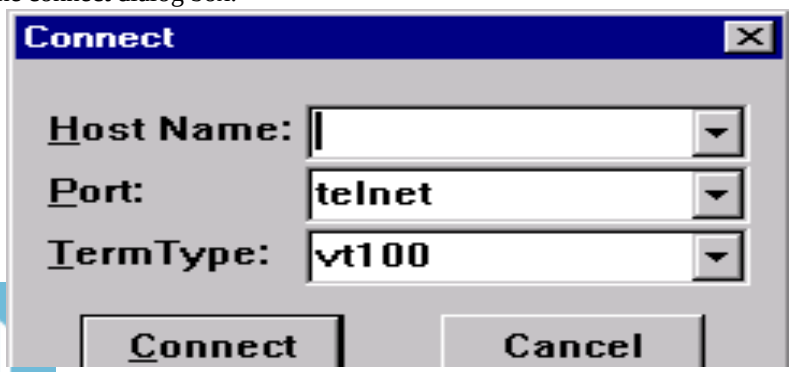


Figure 1.9 The Connect Dialog Box

a. In the Host Name box of the Connect dialog box, type the URL or the IP address of the remote system you want to connect to, e.g., fedworld.gov. This site provides a huge number of US federal documents of all sorts.

b. In the Port box, specify a port number or service to be used. In normal computer terminology port refers to a connection between two devices. On the Internet port numbers are used to identify different types of connection for different services. Normally, for Telnet, the default port number is 23 but for certain programs you have to specify the port numbers. Therefore, for the Cookie server the port which deals with quotable quotes is 12345 and so its complete address is:

astro.temple.edu.12345 or

129.32.1.100 12345

i.e., the host waits to receive commands over port 12345. The reason for stating the host port number is that different port numbers on the same host may receive instructions of different types and perform accordingly.

c. In the Term Type box, specify terminal type, i.e., the type of terminal you want to emulate. The most commonly used terminal type is vt100. You can also use tn3270 if you want your computer to emulate an IBM 3270 computer. But in our example (fedworld.gov) we use the default terminal type vt100 and the default port. Now click the Connect button.

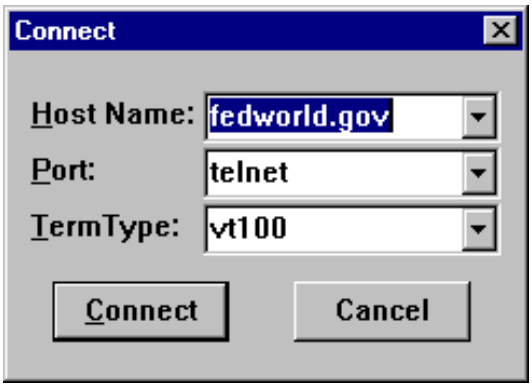


Figure 1.10 The Connect Dialog Box for the Site Fedworld

Step 7: If the above operation has been successfully performed the remote computer will ask you to enter the login name and password. In the above case log on as NEW. No password is required as it allows public access. There are many Telnet sites which allow public access. For these sites there are no passwords and generally the login names are displayed on the screen so that the user is not perturbed. If nothing is mentioned then try using 'guest' as login.

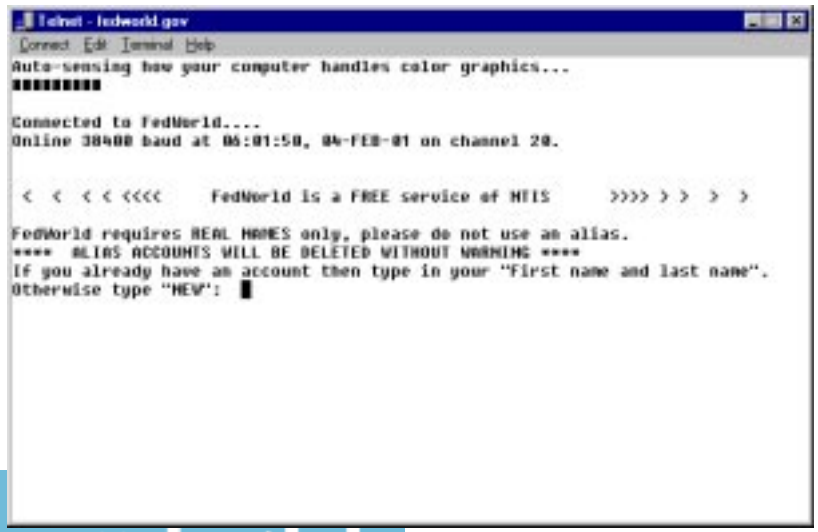


Figure 1.11 The Telnet Site-Fedworld

Step 8: In case you are able to establish a successful connection you get further information.

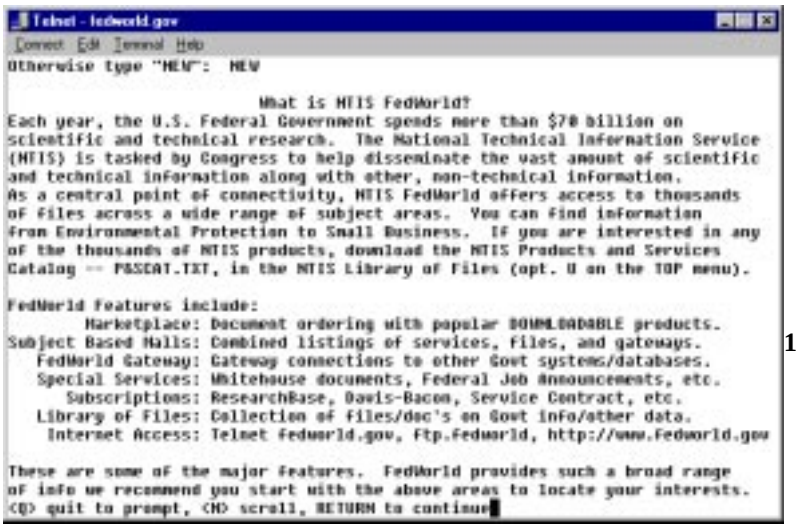


Figure 1.12 A Successful Connection to the Site Fedworld

Step 9: If you are not able to establish a connection, then you will get the message Connection Failed or Connection to host lost.

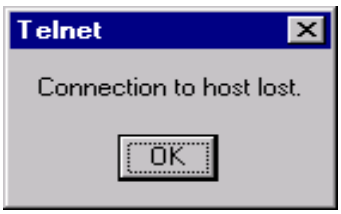


Figure 1.13 Unsuccessful Connection

Step 10: Once all the information required is extracted from the telnet site, then it is always better to log out and then terminate the Telnet session. A Telnet session can be ended by highlighting and clicking the Disconnect option under the tab Connect.

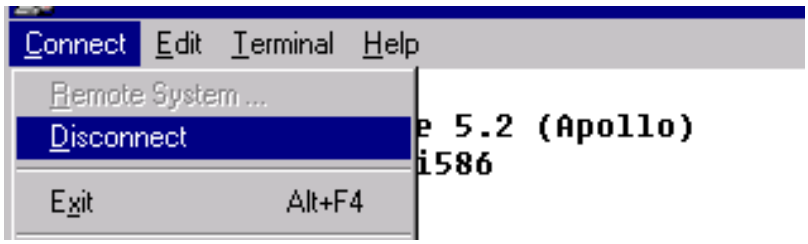


Figure 1.14 Disconnecting from Telnet

Student Activity 5

1. Define Telnet.
2. Describe the usage of Telnet.
3. Write a step-wise procedure to establish a Telnet session.

1.10 E-MAIL

Its long since you have sent a letter and you are still waiting for a response. Well continue waiting as normal postal techniques are slow. But if you are a smart user of the Internet then the entire

process of sending mail and getting a response is reduced to a few minutes if the recipient is also logged on to the Internet. So how is this process followed? Well, just subscribe to any e-mail software online. This process takes a few minutes and then you can send mail to anyone on the Internet provided you know his/her e-mail address. E-mail or electronic mail is one of the most popular applications of the Internet. With e-mail a user can send/receive messages from other users on different networks provided he has access to the Internet, thus simplifying the communication system. Before proceeding in depth let us compare e-mail with snail mail or postal mail.

Table 1.2: Comparison between E-mail and Postal mail

E-MAIL	POSTAL MAIL
1. **Fast:** E-mail takes minutes to deliver messages all over the world.	1. **Slow:** Snail mail takes longer time to deliver messages – from four to five days to weeks depending on the destination.
2. **Cheap:** E-mail costs just the cost of a local phone call.	2. **Expensive:** Snail mail costs as per the National and International tariff rates.
3. **Convenient:** In e-mail, for sending the similar message to many people, only the address book (on the computer) needs to be ticked and the send button of the mail is to be clicked.	3. **Time Consuming:** In postal Mail, for sending the same message to many people, the same number of envelopes with addressses will have to be used each along with a copy of the message.
4. **Mode of Travel:** E-mail is passed on from one computer to another over the network. Each computer on the path routes the message further until it reaches the right destination.	4. **Mode of Travel:** Postal mail makes stops at different postal stations which come on the way while travelling by water/road, etc., till it reaches the destination.
5. **Rate independent of quality and data type:** All sorts of graphics, sound, video, text image and multimedia files of any length can be sent over the Internet.	5. **Rate Depends** on the weight of mailing document and the destination.
6. **Legal:** With the advent of new cyber laws, e-mail documents are considered legal.	6. **Legal:** Postal mail documents are valid proofs and are legal.
7. **Safety:** It is debatable but more or less it is as secure as the postal mail but if you can encrypt messages then it becomes a little more safer.	7. **Safety:** It is safe till the time it falls in wrong hands.
8. **View of the Document:** There is no guarantee that the e-mail received by the recipient is in the same form as that sent by the sender but more or less the message is readable.	8. **View of the Document:** There is no change in the look of the document unless the mail is damaged physically.
9. **Unmanageable:** When you subscribe to mailing lists, within no time you tend to receive mail, some of which may be unwanted and from unknown sources.	9. **Manageable:** You get mail from people whom you generally know though mischief mongers and advertisers may increase the volume of your mail a little.

Applications of E-mail

1. E-mail is generally used for sending proposals and quoting rates for tenders on the Internet.
2. E-mail can be used for personal communication among friends and family members, especially if they are situated at the other end of the world.
3. E-mails are being popularly used for sending greetings, not only between business houses and customers but to individuals also.
4. E-mail on the Internet is very popular among office goers and jobs seekers for sending resumes for various types of jobs.

Working of E-mail

1. Open any e-mail client and key in the message and the address of the recipient.
2. Click the Send button.
3. This message travels in the form of packets which contains the address of the recipient. These packets are then routed to the service provider's server.
4. When an e-mail message reaches the service provider's server, it checks for the validity of the address. If the address is valid then it sends it forward on the Internet otherwise it bounces/returns the mail back to the sender.

5. The forwarded mail is now received by the recipient's provider and sends it to the mail server.

6. The forwarded mail is now stored in the recipient's mail box.1

7. Once the mail is stored in the mail box, whenever the recipient logs on the Internet and checks his e-mail account he can read the mail, send a reply, store the e-mail or even delete it depending on his choice.

Basic Functions provided by all e-mail client software

1. Compose and send mail.

2. View attachments including all sorts of multimedia files by downloading them.

3. Send copies and blind copies of the message to different people.

4. Store different messages in different folders as per the requirement.

5. Subscribe to different newsletters.

6. Add signatures and smileys.

7. Maintain address books.

8. Customise the mail.

9. Inform the sender of the undelivered mail.

10. Communicate directly with other online services like America Online, Compuserve, etc.

Before discussing the various parts of an e-mail client software, let us first subscribe to an e-mail client.

Subscribing to an E-mail Account

There are various means of getting registered to an e-mail software. Some sites provide free e-mail accounts while for others a minimal amount needs to be paid.

Some free e-mail sites are:

1. Microsoft Hotmail or http://www.hotmail.com

2. Yahoo! Mail or http://mail.yahoo.com

3. Rediff mail or http://rediffmail.com

4. Usanet or http://www.netaddress.com

5. Technologist mail or http://www.technologist.com

These e-mail services are free as they are supported by advertisers banners flashing across the screen.

Though you can subscribe to any of the above e-mail clients, in this chapter we will limit our discussion to the e-mail client Hotmail. Let us now see the actual subscription process in order to open an account in the e-mail software namely Hotmail.

How to Create an E-mail Account

Step 1: Connect to the Internet.

Step 2: In the address bar of the browser window type www.hotmail.com.

Step 3: When hotmail gets loaded it asks you to sign in. Signing in means entering your login name/sign-in Name and Password if you already have an account with Hotmail. If you are registering for the first time then click the button

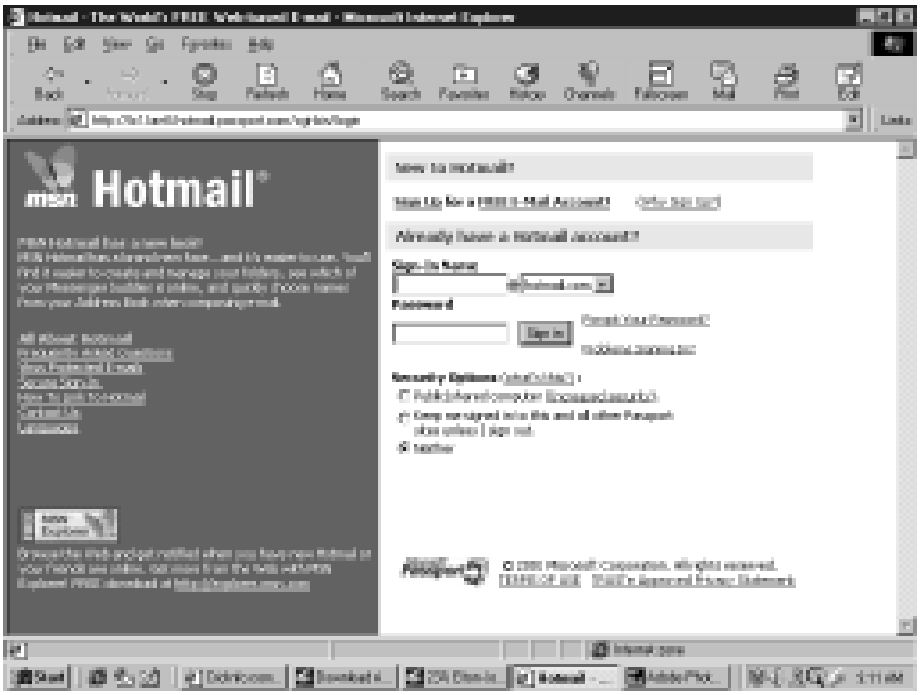


Figure 1.15 Mail Client - Hotmail

Step 4: A form for registering into Hotmail appears. Now enter the required information about your personal profile, namely:

1. First Name
2. Last Name
3. Language
4. Country/Region
5. State/ Territory
6. Time Zone
7. Gender
8. Birthday
9. Occupation

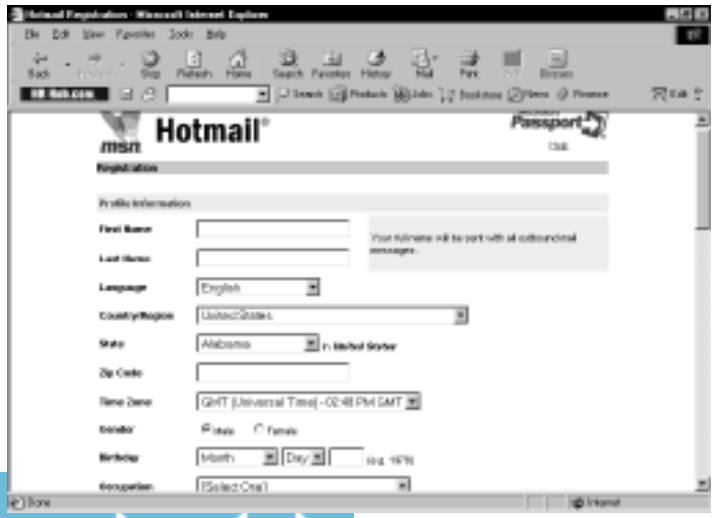


Figure 1.16 The Hotmail registration form

Apart from personal information, also enter information regarding the account namely:

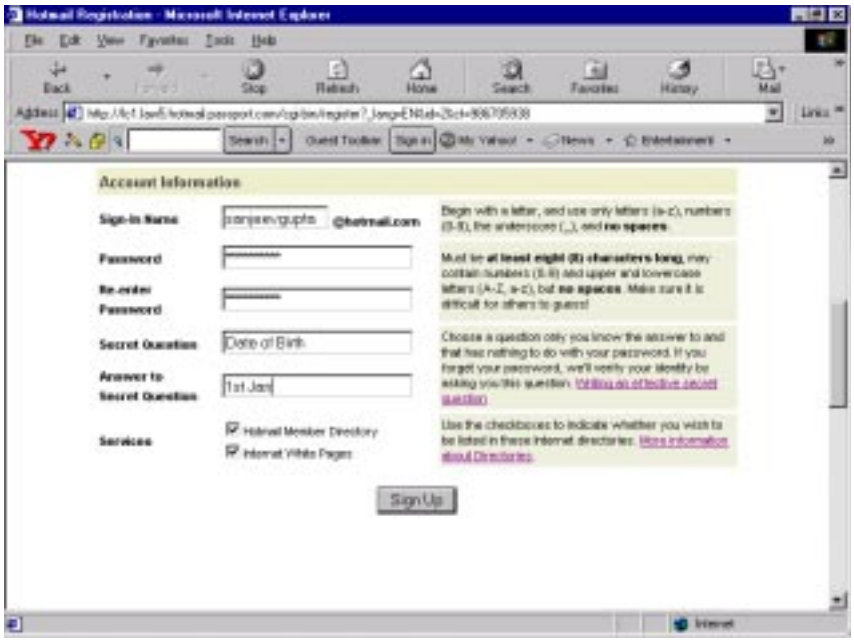


Figure 1.17 Account Information Registration Form

1 Sign -in Name

2. Password

3. Secret Question and an answer to it. This is required for the reason if you forget you password then in order to access your account the system. Prompts you with the secret question and if you give the right answer then your account can be recovered.

Some of these fields are optional while others are essential.

Step 5: Next, click on the Sign up button.

Step 6: This form is then sent to the Hotmail server via the Internet where it is authenticated. During authentication it checks if the login name entered is unique (i.e., no other person has registered by the same login name) and if all the relevant information is complete. If there is some omission then the system prompts the user to enter it again. If the server finds that the login name already exists then the system flashes a message. For example, you enter a login name as sanjeevgupta. The system flashes then it sends a message Registration Error. We're sorry someone has already selected sanjeevgupta. Hotmail now provides a list of similar names for login name. It is now upto the user to select one of the options provided or try another name. After selecting the name click Submit New Sign-In Name. Let us now sign in with the name profgupta.

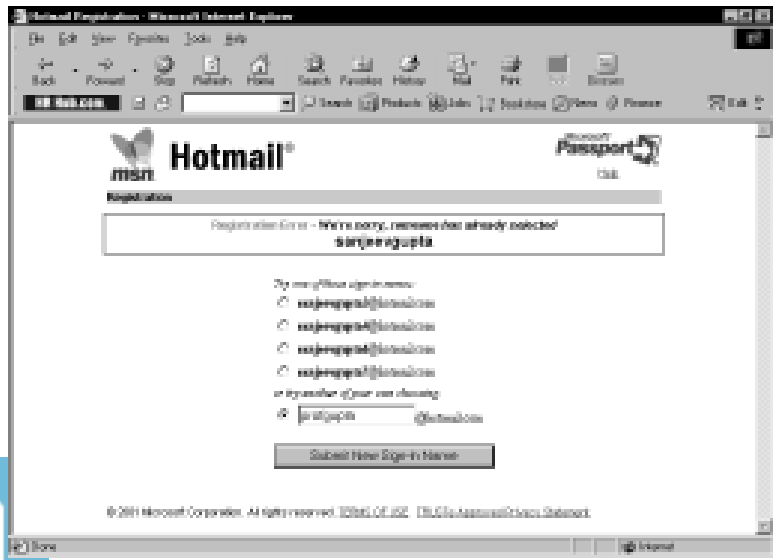


Figure 1.18 Login Name Already Exists

Step 7: If the formalities are complete and the loginnume is unique, then the sytem sends a confirmation message by displaying Sign Up Successful. Congratulations. In this way an e-mail Account is created with Hotmail. Remember your e-mail identity and password. As now onwards, whenever you want to access your e-mail account you will require them. As you click Continue at Hotmail button, the e-mail Client Interface opens up.

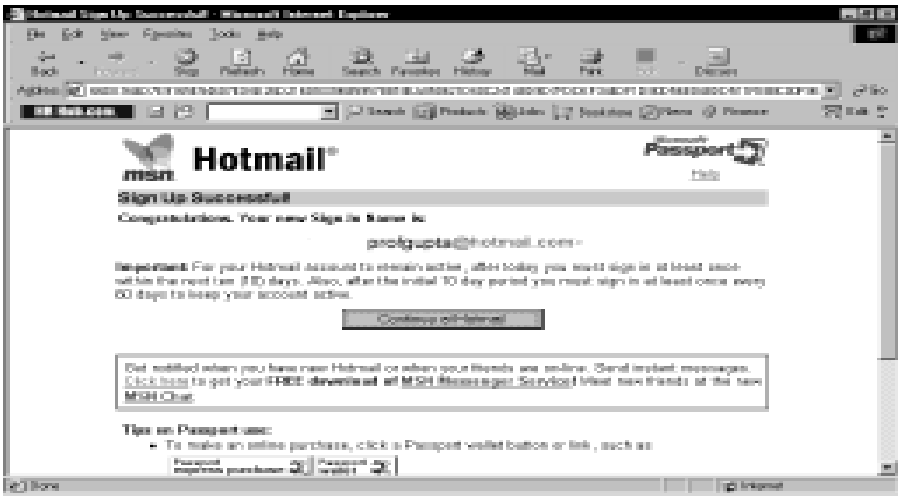


Figure 1.19 Confirmation of Registration

Step 8: If at any time you want to log out then click the sign out option in the Hotmail Interface. On doing so you get a confirmation message.

Figure 1.20 Signing Out From Hotmail

Student Activity 6

1. What is E-mail?
2. Differentiate between e-mail and postal mail.
3. Describe various applications of e-mail.
4. What are the basic functions provided by all e-mail client software?
5. How will you subscribe to an e-mail account?
6. How will you create an E-mail Account?

1.11 GOPHER

Gopher is a protocol designed at the University of Minnesota to search, retrieve, and display documents from remote sites on the Internet. Users interact with Gopher via a hierarchy menus and

can use full-text searching capabilities of Gopher to identify desired documents. It allows one to move around the globe looking for information in various information centres or libraries or servers. Gopher actually goes out, gets the information desired, and puts the information on the screen.

A Gopher offers access to the libraries with library catalogues in different countries. The Internet gives access to the bibliographic records of millions of books, and details on the holdings of academic and research libraries around the world. One can also check on new and interesting titles, and even order them from a number of libraries. Gopher was created as a piece of software to utilise some of the services that were becoming available on the Internet. The integration of many services into Gopher has made the Internet an easier medium to navigate.

When Gopher first appeared, we could access it only through a Gopher client program. Nowadays, users can access the Gopher sites using the web browsers. The main advantage of using a Gopher client is that it comes with a built-in list of Gopher addresses.

1.12 SEARCH ENGINE

Traversing the Internet is simple if the address of the website is known but what happens if the website's address is not known? The best thing about the Internet is that it has a vast pool of resources and the worst thing is that if an Internet user does not know efficient searching techniques then he will be wasting a lot of time and energy. To overcome this, the Internet provides surfers with many search tools to locate information on the Internet. For example, a web surfer may want some information regarding leather bags. What should he do? Whom should he ask? A simple solution would be to get connected to the Internet and type in the URL of any search tool and then perform a search on the relevant keyword/phrase. In our case the search phrase is leather bags. The search tool then matches the keyword and gives a list of websites containing the relevant information. Now the user can open any website on that list just by clicking the hypertext link associated with it. Easy isn't it?

Types of Search Tools

There are basically five ways by which you can perform a search on the Internet, namely:

1. Search Engines.
2. Meta Search Engines.
3. Web Directories.
4. Virtual Libraries.
5. Certain useful web services.

A search engine is a collection of programs that gathers information from the Web, indexes it, and puts it in a database so that it can be searched. The search engine takes the keywords or phrases you enter in the search form and then searches the database attached to it with the search phrase. If it does not find any match then it returns a message No Match Found, otherwise it returns hyperlinks to sources which contain the search phrase along with a brief description of the site containing the search phrase. Search engines contain a computer program called 'spider' or 'crawler'or 'bot' or 'robot' which is responsible for building up the database attached with each search engine. Different spiders collect information using different techniques and from different resources. Some robots are intelligent as they can also find synonyms of the keywords and so results matching those synonyms will also be displayed. Others can automatically remove articles and prepositions from the key phrase and then perform the search. Still others can find the meaning of the key phrase and display results which in essence mean the same.

How to use a Search Engine

We can begin with an example of a simple search with the help of a search engine keeping in mind the example discussed in the introduction of this chapter.

Step 1: To begin with enter the URL of a search engine like www.khoj.com and then press the <Enter> key.

Step 2: Type in the keywords leather bags in the search box next to Search button.

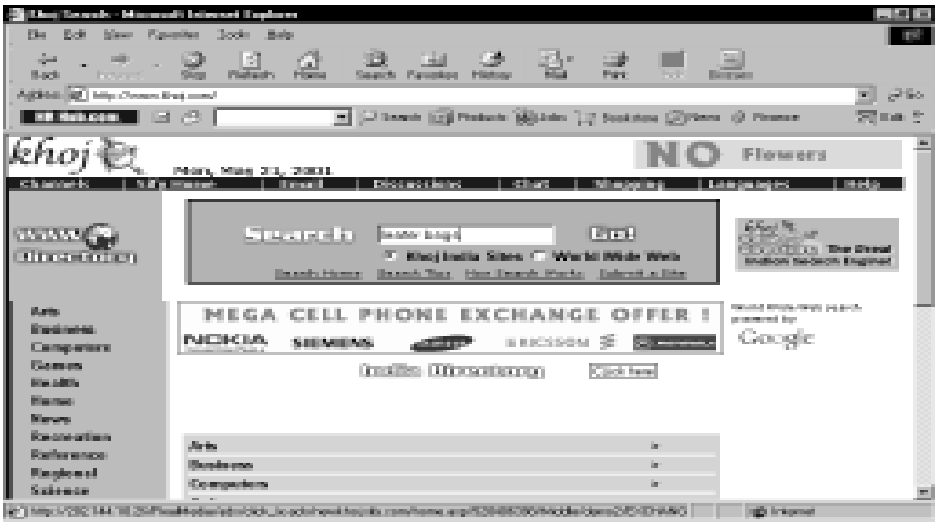


Figure 1.21 Searching the Web for Information on Leather Bags

Step 3: Clicking the Search button a large number of hyperlinks are retrieved. These hypertext links are listed in accordance to the relevance of the search query from most to least relevant.

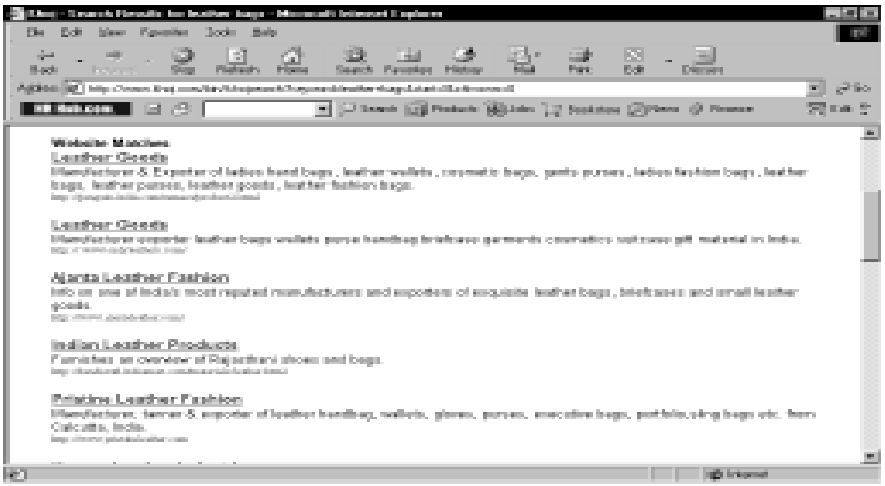


Figure 1.22 Search Results on Leather Bags

Step 4: Click any hyperlink which looks relevant to you and that particular site will open. Let us try the hyperlink leather bags (option 4). If the site contains the information you want then its fine, but what if the information contained is not exactly what you want. Do not fret! Just click on the Back button which takes you back to the search results. Click another hyperlink to open another site. Now wasn't that simple!

Figure 1.23 Clicking on the Hyperlink Leather bags

Some facts associated with Search Engines

1. **Spider Class:** Some search engines are able to traverse down the entire length of the website, no matter how many the levels of directories. These are called Deep Spider Class search engines. A shallow spider can only go to the given URL or can go at the most one step further by entering the URLs in a single level of directories. Therefore, depending on the spider class associated with each search engine the results will be displayed.

2. **Search Results:** Different search engines use different searching techniques. Some provide features like phrase search, for example, “boolean operators”. Some search engines will search for the word ‘boolean’ and another for the word ‘operators’ in websites whereas others will look only in those websites where the words boolean and operators are written together. Therefore by phrase search, we mean that search should be performed on the phrase as a single entity.

3. **Boolean Operators:** Certain search engines support boolean operators like AND, OR, NOT, etc. to narrow down the search. Sometimes AND is implied as + and NOT is implied as– for example, a search phrase like animal and cats will result in sites which contain information about cats which is an animal. In the search phrase if only the key word cats had been used then not only information about the animal cat would be returned but a list of all websites containing the word cat like Cats Computers, Cats Accessories, Cats and Dogs Boutique will also be listed.

4. **Proximity Searching:** There are certain operators called proximity operators like NEAR and WITHIN which further help to get relevant results. For example, if you want to know how earthquakes affect life, then type in the search phrase as Earthquake NEAR life. By giving this query what the search engine actually does is display those results where these words appear in the same document close to each other rather than pages apart.

5. **Wild Cards:** Some search engines allow you to use wild card characters like `*`. Suppose you want to find out information regarding both 'MAN' and 'MEN', you could use in your search query the word M*N. This would give you results related to both.

6. **Media/Field Search:** There are certain search engines which allow you to limit your search only to a particular type of a media/field, e.g., if you want the image of the National Flag of India then you could specify this to the search engine so that only images or video clips could be searched for the right information.

7. **Case Sensitivity:** Certain search engines are case sensitive. In other words, they differentiate capital letters from lower case letters. This is particularly useful if you are dealing with proper names, e.g., if you are trying to locate the website of a 'Hotel' 'called' 'The Park' and if the search engine is not able to distinguish between upper and lower cases then information regarding gardens and parks will also be listed.

8. **Sorted and Relevant Results:** Search engines generally return those results in which the contents closely match the users request. Therefore the most relevant sites are displayed first and then the less relevant ones. Again some search engines allow you to sort your result date-wise.

9. **Summaries:** Generally, when search results are displayed, a summary of sites are also displayed along with hypertext links. Web search engines support hypertext linking, hence users are able to link directly to the resources and the results get displayed.

10. **Other Facilities:** Some search engines can detect duplicate instances of the same page and then remove them. Thus with certain search engines you can specify the number of search results to be displayed in a page. Again, most of the search engines provide the facility to further refine the search or modify it.

There are many search engines. A few of them are:

1. Alta Vista (http://www.altavista.com)
2. Excite (http://www.excite.com)
3. Google (http://www.google.com)

4. Hot Bot (http://www.hotbot.com)
5. Infoseek (http://www.infoseek.com)
6. Khoj (http://www.khoj.com)
7. Lycos (http://www.lycos.com)
8. Megellan (http://www.mckinley.com)
9. Northern Light (http://www.northernlight.com)
10 Rediff (http://www.rediff.com)
11. Web Crawler (http://www.webcrawler.com)
12. Yahoo! (http://www.yahoo.com)

1.13 PLUG INS AND HELPER PROGRAMS

Web browsers are limited in their capabilities as far as interprating those documents which are not in HTML format. However, a browser can be extended in its capability by adding small applications programs.

Such programs that run within the browser are known as plugins. A number of plugins are available from various vendors. Some such plugins are:

- Media Player
- Acrobat Reader
- Shock Wave etc.

Another group of programs which also extends the capability of a browser independently, are know as helper applications or helper programes. Some of such programes are :

- MP3 player
- Download Accelerator

Student Activity 7

1. Define Gopher.
2. Define Search Engine.
3. Name various type of search tools.
4. How will you use a search engine ?
5. Give some facts associate with search engine.
6. Name any five common search engine.
7. What are plug-ins and helper programs ?

1.14 FILE TRANSFER PROTOCOL

FTP stands for file transfer protocol. It is a means of transferring files between computers of different types across a network. A protocol is a language that enables computers to speak to one another. FTP is used to make files and folders publicly available for transfer over the Internet. Standard commands are used to connect to computers, which hold files. These computers are called FTP servers. In some cases you may need to get permission from the network computer's administrator to log on and gain access to files on the computer. But often you will find that you can use FTP to gain access to certain networks or servers without having an account, or being an official password holder, with that computer. These "anonymous" FTP servers can contain a broad range of data that is publicly available through FTP. Most FTP software are shareware but some are available free of cost. A collection of files available at any anonymous FTP site is called FTP archive. It may be difficult to search for FTP sites. Therefore, we use Archie. Archie is a

database of anonymous FTP sites. Archie keeps a record as to what file is stored in which database so that if the user knows a filename then, with the help of Archie, he can find out where that resource is located. With FTP you can upload and download files. File uploading is the procees of moving a file from your computer to some remote computer. File downloading is the process of moving a file from a remote computer to your own.

The general format for ftp is

```
ftp://name_of_ftp_site/directory_name/file_name
```

For example,

1. Microsoft has an "anonymous" FTP server at ftp://ftp.microsoft.com, where you can download files ranging from product fixes, updated drivers, and utilities to Microsoft knowledge base articles and other documentation.

2. ftp://ftp.jpl.nasa.gov/pub/images/browse/mars1.gif

   The FTP site jpl.nasa.gov holds an image of planet Mars in a file called mars1.gif whose directory path is pub/images/browse.

3. ftp://ftp.eff.org/pub/privacy/medical/aclu_drug_testing_workplace.faq

   This FTP site discusses the topic drug testing in the workplace and is paper number 5 of the organisation ACLU.

FEATURES OF FTP

1. FTP also works on the client server technology. Therefore, you can load, client FTP programs like CuteFTP, eFTP, and WS_FTP. FTP daemon, which is a server programme, runs on the FTP server and handles all FTP transactions.

2. FTP servers provide a storage place for useful files and programs.

3. You can log on to these servers and download the files to the local computer. For this you have to know the server name and where the file is located (sub-directory and filename).

4. On some FTP sites, you can only view or download files. Only the people who run or own the site can rename, delete, or upload files. For renaming, deleting or uploading files you may require another different set of logins and passwords.

5. You cannot move files within or between FTP sites. You can move files from an FTP site to a temporary location on your computer or a network drive and then upload them to another FTP site or a different folder on the same site.

6. Some programs may support opening and saving files from FTP servers by typing an FTP address in the File Open or File Save dialog box.

7. It is command driven, therefore, commands like *open*, *get*, *etc.*, are used.

WORKING OF FTP

When you log on to the FTP server a connection called a command link is established between the PC and the server. All commands like changing the directory, downloading files, etc., are sent from you to the server via this command link. Now when the actual downloading process begins a second link called the data link gets established. It is over this data link that the actual download takes place. The connection can open in two modes namely, binary and ASCII. Generally, we use the binary mode as the file's format does not get changed unlike the ASCII mode. After the file gets downloaded, the data connection closes automatically. It is only when you log out of the FTP that the command link gets broken.

Nature of FTP downloads

Types of software that can be downloaded by FTP are:

1. ***Freeware:*** These are software that do not involve any costs.

2. ***Shareware:*** These are complete software which are enabled only for a fixed period of time after which they automatically get disabled. Therefore, if the user is satisfied with the product then he can pay a registration fee and become a licensed user of that software.

3. ***Demoware:*** This is a minature or demo version of the original software. It is just enough to show the functionality of the package but many values are limited. This type of software is generally provided to the users so that they can test the software before purchasing it.

4. ***Patches/Upgrades:*** This includes only a programme or a few programmes which when applied to the main software may add a new module or may remove a bug from the main software.

Though there are many FTP clients we will discuss one of the most popular FTP programs– CuteFTP.

FTP Client CuteFTP

CuteFTP is one of the fastest and easiest way to transfer files across the Internet. Whether you are publishing a web page, downloading the latest digital music and software, or transferring high-volume files between branch offices, CuteFTP provides the tools you need to make your life on the Internet more enjoyable and productive.

Characteristics

1. CuteFTP has a user-friendly interface.
2. It is fast and reliable.
3. It is browser integrated, i.e., CuteFTP automatically launches when you click on an FTP URL.
4. It is a shareware with 30-days trial period free of cost.
5. It has a site manager which is password protected, login dependent and can be encrypted.
6. CuteFTP is compatible with Firewalls, which are used to provide security to your computer system/network against viruses and illegal access.
7. It helps in uploading and downloading files and simply draging and droping files between local and remote computers.
8. CuteFTP includes CuteHTML which is an HTML editor that enables you build a web page.
9. CuteFTP also has a scheduler so that you can easily automate daily uploads, downloads, renames, etc., with CuteFTP's built-in scheduler.
10. It has a Transfer Queue to select multiple files from various sites and queue them for later transfer. The queue transactions may also be edited, saved, and added to the scheduler.
11. Persistent File Transfers facility in CuteFTP allows to auto-reconnect and resume a transfer until the file requested has been uploaded or downloaded, ensuring you get the file transfer completed.

An FTP Session with cuteFtp

Step 1: In order to connect to your FTP site, you must be connected to your ISP. Unless you are connected to your ISP before you launch CuteFTP, you will not be able connect to any FTP site.

Step 2: To start CuteFTP, click the CuteFTP icon that was created during installation. **The FTP Site Manager** window will be displayed (Figure 1.24). This is where you will enter the FTP server information to connect to the remote computer where your files will reside. To connect to any of these sites, follow the next step.

Step 3: Click the **Connect** button.

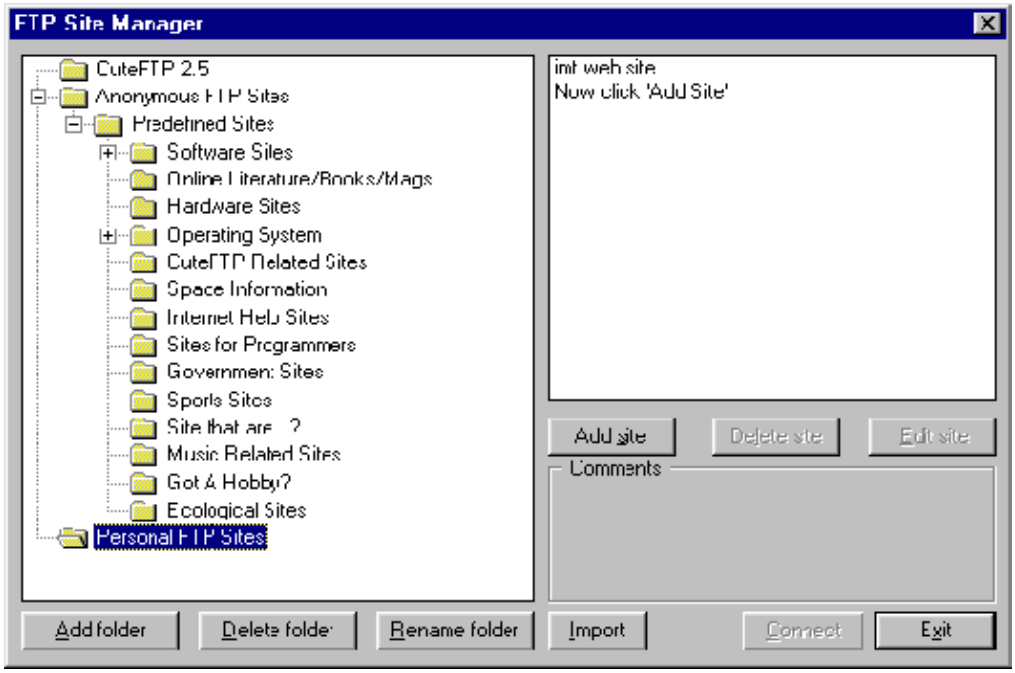


Figure 1.24 FTP Site Manager

Step 4: Configure your FTP connection. In the **FTP Site Manager** window, select a folder from the left window for your site to reside or create a new folder by clicking the **Add Folder** button. Enter a folder name and click the **OK** button. Next, highlight the newly created or selected folder, and click the **Add site** button. This will bring up the **Edit Host** where your server information will be stored (Figure 1.25). The four entries that will be covered here are given below:

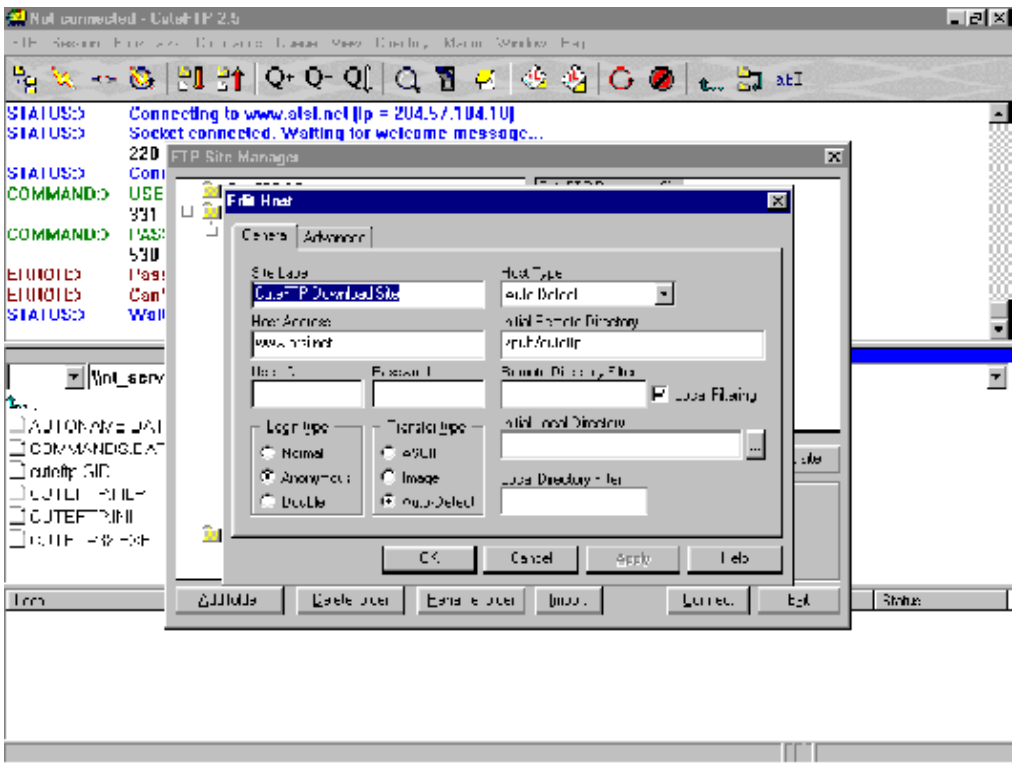


Figure 1.25 Host Screen Form

a. Site Label box

b. Host Address box

c. User ID box

d. Password box

a. ***Site Label Box:*** This box will identify your site so that you can easily access your FTP server from the Site Manager. You can label it whatever you want.

b. ***Host Address Box:*** This is where you enter the FTP server address (i.e., host address, FTP address, server address, IP address, etc.). You must obtain this FTP server address from the service provider who will be hosting your web pages.

(c) ***User ID Box:*** Your server administrator provides this unique ID. It is case sensitive so be careful to enter capital and lower case letters correctly. If you do not know what your user ID is, you can obtain this information from your ISP or service administrator.

(d) ***Passwords Box:*** Your server administrator provides this unique password. Like the user ID, your password is case sensitive, hence ensure capital and lower case letters are typed correctly. If you do not know what your password is, you can obtain this information from your ISP or service administrator. You will not be able to see what you are typing in to the password field, so be careful not to make any mistakes.

Once you have entered this information, click the **OK** button, which returns you to the **FTP Site Manager** window, where your site will be displayed in the right window.

Step 5: In order to connect to the server, highlight the site you have created and click the **Connect** button (Figure 1.24). After connecting, you should see a message from your ISP informing you of rules and regulations of the server. Click the **OK** button (Figure 1.27) at the bottom of that screen and the message will be cleared. To modify your server information, highlight your site in the **FTP Site Manager** window and click the **Edit site** button (Figure 1.24). Make your modifications and click the **OK** button. You can also rename and delete sites and folders from the **FTP Site Manager** window with the corresponding buttons (Figure 1.24).

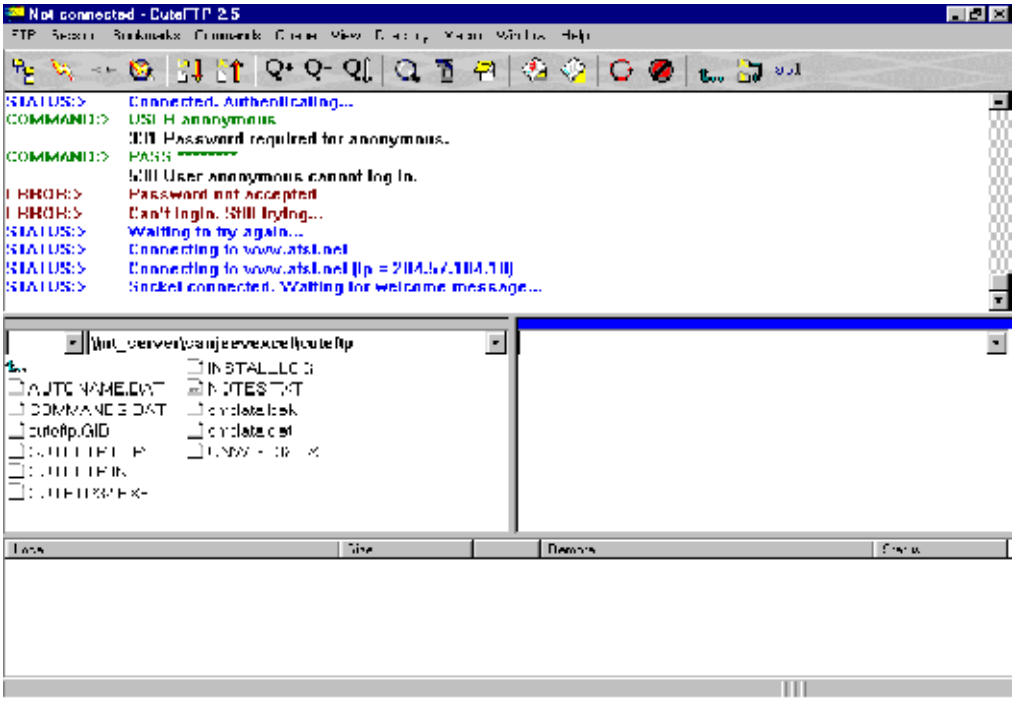


Figure 1.26 FTP Window

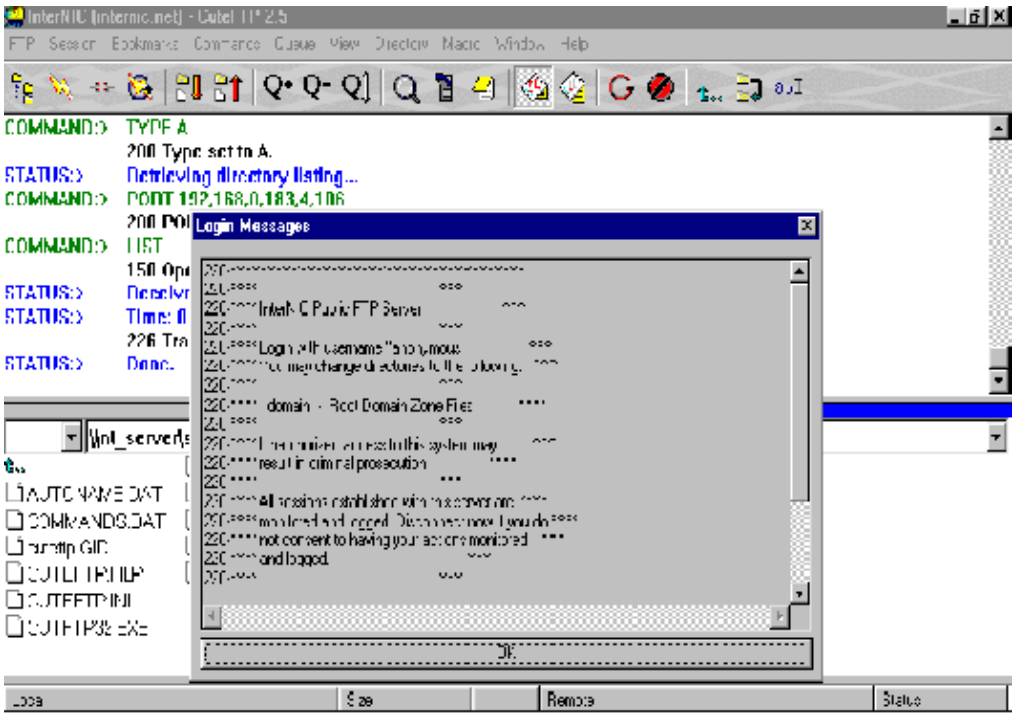


Figure 1.27 Login Message

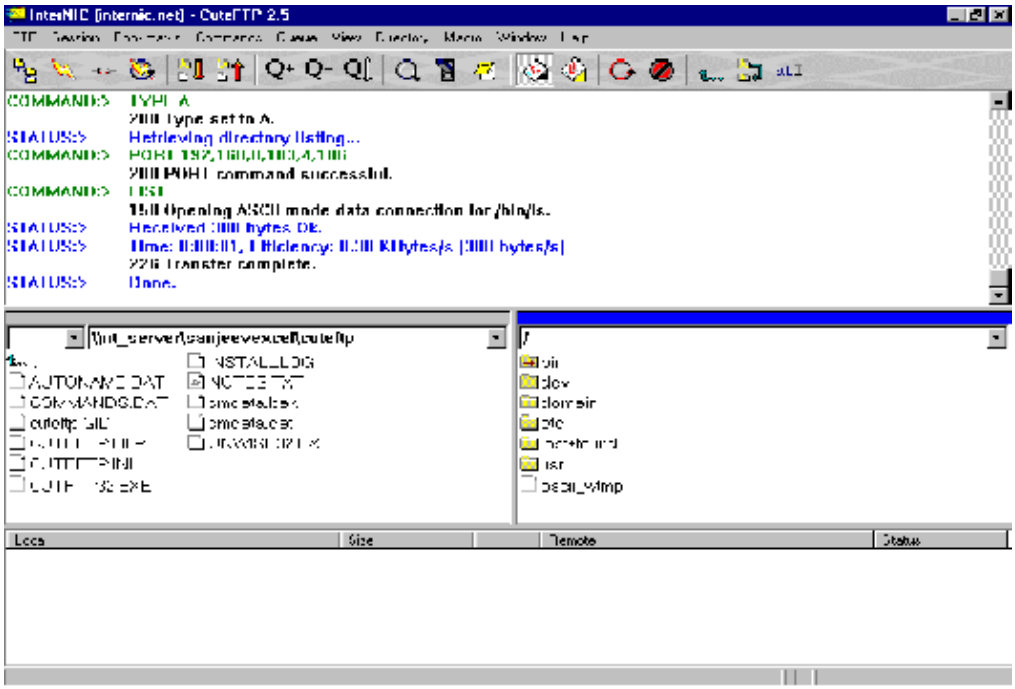


Figure 1.28 FTP Window after connect

Step 6: Use CuteFTP Main Screen to move your files to the server. This screen is divided into four windows: an upper horizontal window, two lower vertical windows, and a bottom horizontal window (Figure 1.28).

(a) ***Upper Window (Log):*** This is where the command CuteFTP gives the server as well as the server's responses are displayed (Figure 1.28).

(b) ***Lower Left Window (Local):*** The lower left window displays the selected directory from your hard drive (Figure 1.28). By default, it should display the contents of your CuteFTP directory. Locate the directory beneath the root where your HTML files (or whatever files you want to upload) are located. Once you are in the correct directory you will see your files displayed in that left window.

(c) ***Lower Right Window (Remote):*** The lower right window displays the files and directories on your server (Figure 1.28). You are now viewing the hard disk drive contents of a remote computer. Although there are probably many directories above this, the server limits you to your directory and to any sub directories you create beneath your virtual root.

(d) ***Bottom Window (Queue):*** The queue window is used for temporary storage of files you wish to transfer at a later time. You can browse a remote site, and drag and drop files from different directories into the queue window. Once you have chosen your files, you can batch transfer them by clicking on Queue/Transfer option. All preliminary actions have now been accomplished and the files can be moved to the server.

There are five methods of moving the files to the server:

(a) ***Method One:*** This is the drag and drop method. Highlight the file(s) on your local drive that you want to upload. Use your left mouse button to "grab" the highlighted file(s) and drag it (them) to the right window and drop it (them) anywhere in that window. By default, you will receive a prompt to "upload selected file?". Click the **OK** button after you have verified the file name(s). The upper window will display commands, status, and the server's response to your commands.

(b) ***Method Two:*** Highlight the files you want to upload in the left screen and then click the Upload button located at the top of the CuteFTP screen below the Menu Bar. It looks like an arrow pointing up. If that button is not highlighted, you have failed to highlight the files you want to upload. When you click the Upload button, the Confirm Upload screen, as in the previous method, will prompt you. You may accomplish the same thing by selecting commands/upload from the Menu Bar.

(c) ***Method Three:*** Double click the file you want to upload. By default CuteFTP takes that action as a Transfer File request.

(d) ***Method Four:*** Highlight the file you want to upload and Right click it. Left click the upload option in the Menu that appears. Continue as in the previous methods.

(e) ***Method Five:*** If you have transferred the files to the queue, and you are ready to upload them to the server, click Queue/Transfer Queue. As CuteFTP auto-refreshes the remote directory, you should now see your files on the right side (remote side) of the screen. If the files seem to be uploading (the log is showing lots of commands), but you do not see the files displayed on the server, then there can be a problems. Scroll up the log in the upper window using the scroll bar. Any entries in red such as a "Permission Denied" is an indication that you are not being allowed to upload the file(s) and you will need to contact your ISP or service administrator. When you see your files on the right hand side, you have completed the file transfer. You can now type in your URL or web page address in your browser and you should see the web page corresponding to the HTML file(s) you uploaded.

FTP Commands

There are basically six FTP commnds which are :

1. ***OPEN:*** This opens a communication channel between your computer and the FTP server.
2. ***CLOSE:*** This informs the remote computer that the job is over.
3. ***CD:*** This helps to change directory on the FTP server since the required file may be on another directory.

4. ***DIR :*** It gives the directory listing of the current directory.

5. ***GET:*** With this command you retrieve the file from the remote computer to your own. Here you have to specify the entire path of the file.

6. ***PUT:*** This lets you send one or more files from your computer to the remote computer.

1.15 SUMMARY

- Web pages run your server programs through a server feature called common Gateway Interface.
- The programs that run on the server under the feature are called CGI Scripts for historical reasons.
- The eariest way to envision a CGI program is to imagine that it is part of Web Server.
- There is no difference between a URL that corresponds to a static document and one that corresponds to a CGI programs.
- CGI program must be written to follow the server's guidelines.
- Unlike a conventional computer program, a CGI program does not write output on a user's screen.
- A CGI program does not receive input from a keyboard or a mouse – the browser handles all interaction with the user.
- CGI programs can also keep a record of which corporate web pages a user visits and choose advertisements that suits the users.
- An active document is a computer program that points an image on a screen.
- To pass data from forms to server scripts, you use special attributes within the various form tags.
- A modem is a device, which is used to convert digital signals to analog signals so that they can be transferred over the standard telephone line.
- A cable modem enables you to attach your PC to a local cable TV line and receive data at approx. 3.5 mbps.
- An Internet Service Provider (ISP) is an organization or business offering public access to the Internet.
- Browsers are sometimes also called web clients, since they get information from a server.
- URL stands for Uniform Resource Locator, which is simply an address of a document on the web or, more accurately, on the Internet. A URL will be shorter and include less information if the file is on your local computer or server. Basically URLs fall into two categories – absolute and Relative.
- The need to augment internet security is being alarmingly realized with the emergence of E-commerce.
- Firewall is a collection of components or a system placed between two networks. A firewall basically performs two important functions : gatekeeping and monitoring.
- Telnet is a Standard Internet protocol for remote terminal Connection Service. Generally Telnet is used in order to retrieve : Special information from a remote database, weather reports, Library catalogs, Departmental stores catalogs, E-mail while touring, Information from other Internet tools like FTP, Gopher, etc.
- Gopher is a protocol designed at the University of Minnesota to search, retrieve and display documents from remote sites on the Internet. A Gopher offers access to the libraries with library catalogues in different countries.

- There are basically five ways by which you can perform a search on the Internet, namely : Search Engines, Meta Search Engines, Web Directories, Virtual Libraries, Certain useful web services.

1.16 KEYWORDS

(CGI) Common Gateway Interface : Server feature through which web pages run the server programs.

Forms : The webpage with blank areas in which the user enter information.

Modem (Modulator-Demodulator) : A device used to convert digital signals to analog signals and vice versa.

Cable Modem : A device that enables to attach the PC to a local cable TV line and receive data at approximately 3.5 mbps.

(ISP) Internet Service Provider : An organization or business offering public access to the Internet.

HomePage : The first page of a website.

Website : A collection of webpages.

Web Browser : A *www* client which sends the request to the web server.

(URL) Uniform Resource Locator : Address of a document on the web or Internet.

Firewall : A collection of components or a system placed between two networks which acts as an electronic barrier to stop unauthorized entry.

Telnet : A Standard Internet protocol for remote terminal connection service.

E-mail (Electronic mail) : An application of Internet used to send / service messages to / from other users on different networks provided he has access to the Internet.

Gopher : An Internet protocol used to search, retrieve, and display documents from remote sites on the Internet.

Search Engine : A collection of programs that gathers information from the web, indexes it, and puts it in a database so that it can be searched.

Plug-ins : Programs that run within the browser.

FTP (File Transfer Protocol) : FTP is a set of rules for transferring files on the Internet.

1.17 REVIEW QUESTIONS

1. Fill in the blanks :
   a. Web pages run server programs through a server feature called the .....
   b. ..... make it possible to enter data directly.
   c. The two methods of passing data that a script may use are ..... and .....
   d. Browsers are also called .....
   e. A URL can be ..... or .....
   f. ..... and ..... are important aspects of firewall design.
   g. ..... and ..... are helper programs.
2. How is CGI helpful in advertising ?
3. List some facilities provided by an ISP.

4. Describe the functioning of Gopher.
5. What can you send through E-mail?
6. What is the advantage of Telnet ?
7. Compare shell account and TCP / IP account.
8. Describe various elements of a URL.

Answers to Review Questions

1. (a) CGI (b) Forms (c) GET, POST (d) Web clients
   (e) absolute, relative (f) Gate keeping, monitoring
   (g) MP3 player, Download Accelerator

1.18 FURTHER READINGS

Ed Titled ; *Foundations of Worldwide Web Programming with HTML and CGI* ; 1995, IDG Books Worldwide.

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT 2

HTML

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe HTML and its various versions.
- Create an HTML document.
- Understand the Basic Structure of an HTML document.
- Use various presentation controls.
- Insert various types of list in the web page.
- Add pictures and images in the web page.
- Display wall paper in the background or set a background colour.
- Create forms.
- Link web pages and publish documents.

U N I T S T R U C T U R E

2.1 INTRODUCTION

Hypertext Markup Language (HTML) is the language of World Wide Web pages. The formats that apply to text, headings, and graphics on Web pages are controlled by HTML. HTML's popularity has brought hypertext technology — the technology that lets you jump from topic to topic, rather than finding and reading information linearly — to the fingertips of people worldwide, particularly in the context of the World Wide Web.

2.2 HTML - AN INTRODUCTION

HTML stands for Hyper Text Markup Language. HTML represents a way to take ordinary text and turn it into hypertext by just adding special elements – called markup tags – that tell Web browsers how to display a Web page's contents.

The main goal of HTML is to be a universal language for classifying the function of different sections of a document. It is designed to work on a wide variety of platforms. HTML and the Web were first conceived in 1989 by a researcher named Tim Berners-Lee who worked for CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. The Web pioneers, led by Berners-Lee, regrouped into the Word Wide Web Consortium (W3C) in December 1994. The W3C is now responsible for the standards of HTTP, HTML, and other Web technologies.

HTML is one member of a family of markup languages called SGML. SGML was developed by the International Organization for Standards in 1986 to define markup languages designed for various different purposes. Every language in the SGML family conforms to certain requirements, the main one being that all of the symbols are strictly defined using a DTD, or Document Type Definition. The DTD for HTML defines what tags are available and how they can be used. HTML shares important characteristics with its SGML parent:

- A character based method for describing and expressing content.
- A desire to deliver that content equally to multiple platforms.
- A method for linking document components together to compose compound documents.

HTML : The first version of HTML was just called HTML. The basic tags have not changed much. HTML has tried to stay backward – compatible so that new versions of HTML do not contradict old versions.

HTML + : Dave Raggett worked on a successor to HTML called HTML + in 1993. Although it was never an official specification, many of its ideas were incorporated into HTML 2.0.

HTML 2.0 : A specification for HTML 2.0 was created in July 1994, and after editing by Dan Cannolly, HTML 2.0 became a standard approved by IETF in November 1994. HTML 2.0 was the first popular version of HTML and the massive explosion of Web popularity took place from late 1994 to 1995 using Web pages written in HTML 2.0. HTML 2.0 was criticized for not containing more formatting commands.

HTML 3.2 : HTML 3.2, originally code-named "Wilbur", became finalized in January 1997. HTML 3.2 was immediately popular, mostly because it supported existing practices in a logical way and was more compatible with HTML 2.0. It added support for tables. IE3 and Netscape 3 supported HTML 3.2 almost completely.

HTML 4.0 : Code-named "Cougar", HTML 4.0 is the latest version of HTML. The largest difference between HTML 4.0 and previous versions of HTML is the character set. HTML 4.0 uses a character set called "Unicode" that allows thousands of different characters. HTML 4.0's specification strongly encourages the use of style sheets and discourages tags and attributes that are solely for formatting, visual appearance, and layout. HTML 4.0 also introduces the <OBJECT> tag, which is used to present multimedia.

Document Overview

HTML documents are essentially plain text files. They contains no images no sounds, no video, and no animations; however, they can include "pointers", or links, to these other file types, which is how Web pages end up looking as if they contain non-text elements.

HTML itself is a system of codes made up of tags and attributes that serve to identify parts and characteristics of HTML documents. Some tags provide document structure; others reference other files. Attributes provide additional information within tags. HTML tags identify logical document parts — that is, the major structural components are part of the HTML documents

such as headings, lists, and paragraphs. For example, if you want to include a heading, a paragraph, and a list in your document, you type the text and apply the appropriate tags to the text.

How structural components exactly appear on your computer screen depends on the browser. Some tags work in conjunction with attributes, which provide additional information about an element, such as how elements should align (left, centre, or right), what other file should be accessed, or even the colour of an element. For example, an attribute might indicate that a heading should appear centred in the browser window, that the browser should load an image file from the Web, or that the Web page background should appear sky blue.

HTML is made up of tags and attributes, which work together to identify document parts and tell browsers how to display them. Tags identify document parts by specifying, for example, that a chunk of information be displayed as a heading. Attributes are optional parts of tags and modify or specify information in tags, such as colour, alignment, height, or width.

Adding Tags

Tags are easy to read and use once you become familiar with their components. First, all tags are composed of *elements* that are contained within *angle brackets* (< >). The angle brackets simply tell browsers that the text between them is a HTML command. A tag with its angle brackets looks like this:

```
<TAG>
```

Second, most tags are paired, with an opening tag (<TAG>) and a closing tag (</TAG>). Both tags look alike, except the closing tag also includes a forward slash (/). To apply tags to information in your document, place the opening tag before the information, and place the closing tag after the information, like this:

```
<TAG> information that the tag apply to </TAG>)
```

Including Attributes

Attributes provide extra information about a tag. For example, if you apply a heading tag (<H1>) to a line of text, like this:

```
<H1> A heading goes here </H1>
```

You can further specify that it should be centred by using an attribute, like this:

```
<H1 ALIGN = "Centre"> A centred heading goes here </H2>
```

As a general rule, most attributes — those that include only letters, digits, hypens, or periods — work fine without quotes. For example, you can type ALIGN = CENTRE or ALIGN = "CENTRE"; all browsers should display these in the same way.

Attributes that have other characters, such as spaces, or # signs, however, do require, quotes. For example, if you use the WIDTH = attribute to indicate percentage of the document window, type WIDTH = "75%".

Applying Structure Tags

Structure tags provide browsers with information about document characteristics, such as the version of HTML used, introductory information about the document, and the title. Most structure tags, although part of the HTML document, do not appear in the browser window. Instead, structure tags work "behind the scenes" and essentially tell the browser which elements to include and how to display them.

All HTML documents should include five structure tags, nested and ordered.

2.3 CREATING A WEB PAGE

HTML syntax describes how a Web browser recognizes and interprets the instructions contained in the markup tags. The special control characters that separate HTML markup from ordinary text are the left and right angle brackets: < >. Inside the browser software, a parser reads and

constructs display information. It instructs the browser to take appropriate action based on the markup it finds. In HTML, left and right angle brackets can enclose all kinds of special instructions called tags.

A tag takes a generic form that looks something like this:

```
<Tag Name {Attribute {="Value"} ...} > Text {</Tag Name>}
```

(Text within tags is case insensitive)

<TAG NAME> : All HTML tags have names. For example, H1 is a level-one header; OL is an ordered list. Tags are surrounded by angle brackets to mark their contents for special attention from the parser.

{ATTRIBUTE{="VALUE"}...} : Some HTML tags require or permit named attributes to be associated with them. Some attributes require that value be associated with them.

For example, in an <IMG> tag (used to point to a graphics file), a required attribute is SRC (Source), to provide a pointer to the file where the graphic resides, as in IMG SRC = "../gifs/redball.gif."> (...) indicates that some HTML tags may include multiple attributes, each of which may or may not take a value.

Text : This is content that is modified by a tag. For example, if the tag were a document title, the HTML string <TITLE> UPTEC Computer Consultancy Ltd. </TITLE> would display the text in the title bar at the top of a graphical browser's window.

{</TAG NAME>} : A closing tag name is denoted by a left angle bracket (<), followed by a forward slash (/), then the tag name, and finally a closing right angle bracket (>); curly braces indicate that this element does not always occur.

The ampersand (&) is another special HTML control character. It is used to denote a special character for HTML content that may not belong to a 7-bit ASCII character set. Such tagged items are called character entities. Some HTML elements have no content. For example, the horizontal rule element (which uses the <HR> start tag) has no content; its only role is to create a line. Elements with no content are called "empty elements", and they never have end tags.

First HTML Example

To create this HTML example, just start your text editor and type in the following HTML code and save it as S11.htm.

```
<HTML>

<HEAD>

<TITLE> A Hello World Example in HTML </TITLE>

</HEAD>

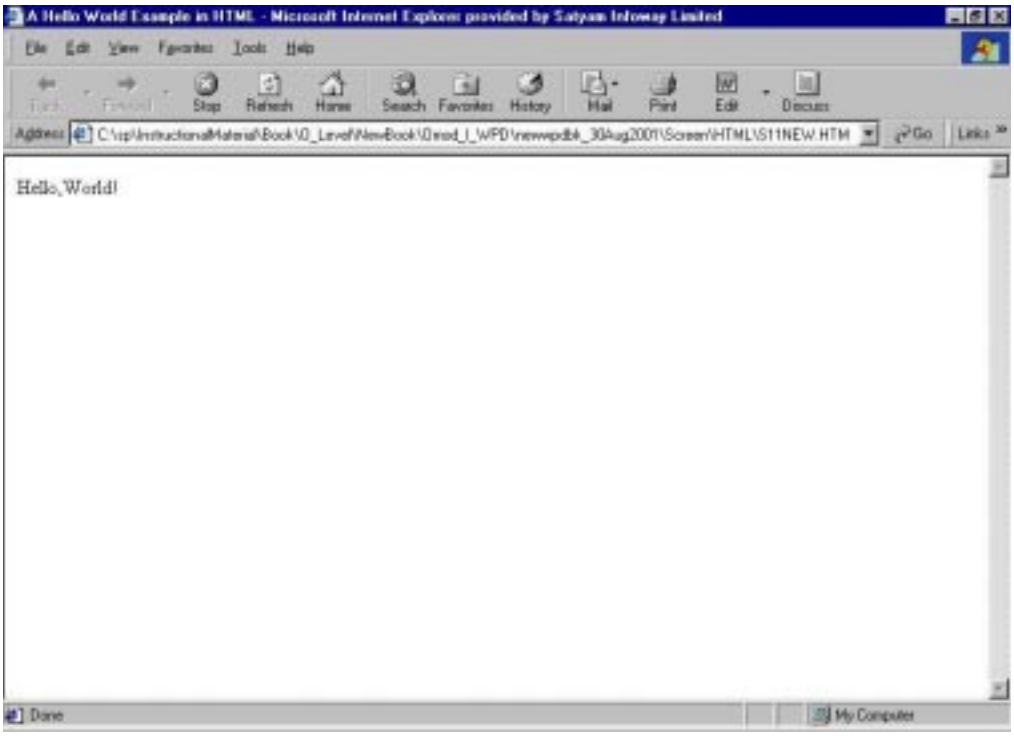
<BODY>

Hello, World !

</BODY>

</HTML
```

Netscape Navigator will display this code as a simple web page.



Student Activity 1

1. Define HTML.
2. Describe various version of HTML.
3. What is a tag ?
4. What are attributes ?
5. Describe various control characters of HTML.

2.4 BASIC STRUCTURE OF AN HTML DOCUMENT

Well structured HTML documents come in these three parts:

1. *A head* that identifies a document as HTML and establishes its title.
2. *A body* that contains the content for a Web page. This part holds all displayed text on a page, as well as most links to graphics, multimedia, locations inside the same file, and to other Web documents.
3. *A footer* that labels a page by identifying its author, date of creation, and version number.

Defining HTML Documents with the HTML Element

One should bracket an entire HTML document by the identification tags <HTML>, to open the document, and </HTML>, to close it. These tags identify the DTD for the document to an SGML sensitive program, to allow the program to interpret a document's contents properly. An optional line may sometimes precede a document head. It is called a document type prolog and describes, in SGML, that the HTML document complies to the indicated level of the HTML DTD.

```
e.g.: <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2// EN">
```

It indicates that the HTML document confirms to the HTML 3.2 DTD distributed by the Internet Engineering Task Force (IETF). You can also tell that the DTD is PUBLIC and is not system dependent. Finally, you can tell that the HTML tag set is defined by the English language (the EN in the DOCTYPE statement).

Describing Documents with the Head Element

The head element is used to mark the position of the head section. The head section contains elements that define certain information about an HTML document, such as what its title is, who the author is, and reference information about the document. To create a head element, start with <HEAD> tag, then include all of the elements you want in your head section, then end the head element with a </HEAD> tag.

Naming Documents with the Title Element

Titles are displayed by browsers on the top of the page, usually in the title bar. Every HTML document must have a title contained in a <TITLE> start tag and a </TITLE> end tag. For example:

```
<HEAD>

<TITLE> A Hello World Example in HTML </TITLE>

</HEAD>
```

Wrapping Your Content with the Body Element

The real content for any HTML document occurs in the body section, which is enclosed between <BODY> and </BODY> tags.

Two Categories of Body Elements

There are two basic categories of HTML elements used in the body section:

- Block-Level Elements
- Text-Level Elements

Block-Level Elements

Block-level elements are used to define groups of text for a specific role. They include tags that position text on the page, begin new paragraphs, set heading levels and create lists. Some commonly used block-level elements and their tags are:

```
Paragraph: <P> and </P>

Heading, level one: <H1> and </H1>

Heading, level two: <H2> and </H2>

Horizontal rule: <HR>

Centreing: <CENTER>
```

Text-Level Elements

Text-level elements are for mark up bits of text, including creating links, inserting things like images or sounds, and changing the appearance of text. Some commonly used text-level elements are:

Bold: <B> and </B>

Italic: <I> and </I>

Line-break:

Link anchor: <A HREF = "URL"> and </A>

Image: <IMG SRC = "URL">

Footer

Technically speaking, HTML does not include a separate tag to denote a page footer. But it is recommended because a good footer helps to identify a document's vintage and contents and let interested readers contact the author if they spot errors or want to provide feedback.

e.g.: Type the following HTML code, and save it as S12.htm in your web folder.

```
<HTML>
<HEAD>
<TITLE> Rupert's Fabulous T-shirt Company </TITLE>
</HEAD>
<BODY>
<HI> Welcome to Rupert's Fabulous T-shirts! </H1>
<H2> Fabulous T-shirts Since 1752 </H2>
```

Our Company, <B> Rupert's Fabulous T-shirt Company </B>, is your <B> <I> second-best choice </I> </B> for T-shirts. (The best choice is <A HREF = "http://WWW. inkyfingers.com/"> Inky Fingers </A>.)

```
Write us at: <BR>
555 Garment Way <BR>
Alameda, (A 94412
<P>
(all us at (510) 555-9912. <I> <B> We're here to help: </B> </I>
<IMG SRC = "http://www.emf.net/~estephen/images/turle shirts.jpg")
<HR>
<CENTER> Why not visit <A
HREF = "http: //www.yahoo.com/"> Yahoo</A>?
</CENTER>
</BODY>
</HTML>
```

Student Activity 2

1. Describe the basic structure of an HTML document.
2. What is the function of <TITLE> tag ?
3. What are block-level elements ?
4. What are text-level elements ?
5. What is the advantage of inserting a footer in an HTML document ?

Presentation Controls / Text Presentation Tags

Presentation tags alter the display of content by affecting properties such as font styles and horizontal rules.

Using Font-Style Elements

Font-Style elements change the appearance of text. These font-style elements are known as physical markup. All font-style elements are a subcategory of text-level elements, and they all require both start and end tags. They can all be nested according to the normal rules of nesting text-level elements. The seven font-style elements are: bold (<B>), italic (<I>), underline (<U>), strikeout (<STRIKE> or <S>), big (<BIG>), small (<SMALL>), and teletype (<TT>).

```
Tag: <B> ... </B>

Tag Name:     Boldface
```

Explanation:

- It indicates that the enclosed text is in bold face type. The bold element does not indicate strong emphasis when read by some text-only or text-to-speech browsers.

```
Tag: <I> ... </I>

Tag Name:     Italic
```

Explanation:

- This element marks up text in italics (text slanted diagonally upward to the right).
- It is appropriate to use the italics element to indicate text in a foreign language.

```
e.g.:   <I> corpe diem </I>
```

NOTE: Remember that its effectiveness fades quickly with overuse. If you are including a citation or bibliographic resource in your document, use <CITE> instead of <I>.

```
Tag: <U> ... </U>

Tag Name:     Underlined text
```

Explanation:

- The Underline element underlines text.

```
e.g.:   <U> Hello, World ! </U>

Tag: <STRIKE> ... </STRIKE>

Tag Name:     Strikethrough
```

Explanation:

- This element indicates that the enclosed text should have a line drawn through the middle of the text.
- Strikethrough text is difficult to read onscreen, so use this tag sparingly. Use <STRIKE> to show text removed from earlier versions of a document, the old text will appear with a line through it.

```
e.g.:   <STRIKE> I'm struck ! </STRIKE>

Tag: <BIG> ... </BIG>

Tag Name:     Big text
```

Explanation:

- It makes text font one size larger than the <BASEFONT> size.

- Nesting <BIG> tags can produce text in a larger font than using only one <BIG> tag.

```
e.g.:  <BIG> The Big and Tall Company </BIG>

       <BIG> <BIG> The Very Big and Tall Company </BIG> </BIG>

Tag: <SMALL> ... </SMALL>

Tag Name:     Small text
```

Explanation:

- This element makes text one size smaller than the basefont size.
- Nesting <SMALL> tags can produce text in a smaller font than using only one <SMALL> tag.

```
e.g.:  <SMALL> The Small and Short Company </SMALL>

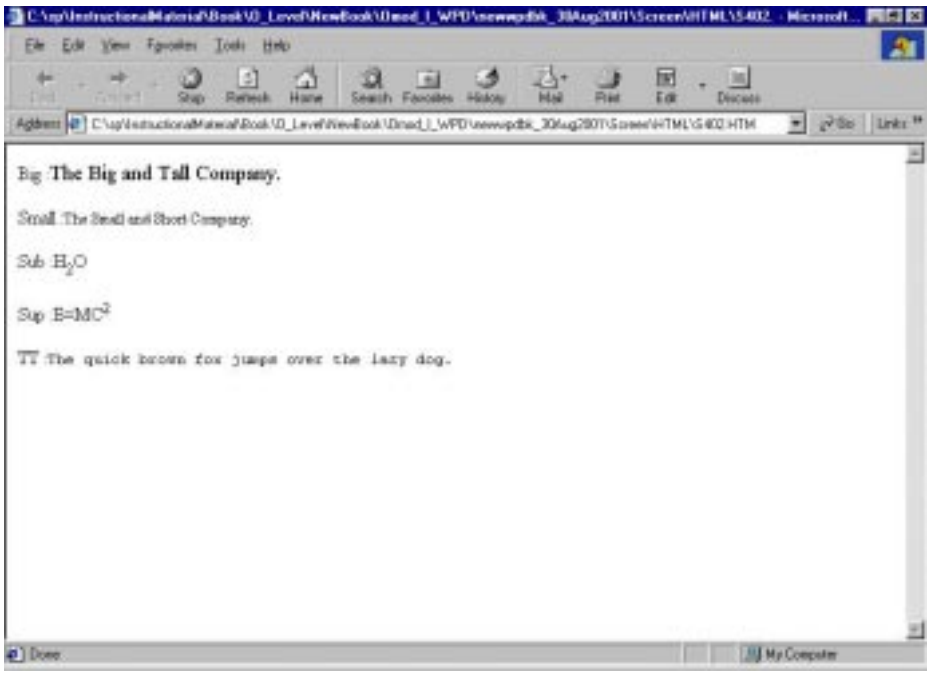
       Tag: <TT> ... </TT>

Tag Name:     Teletype text
```

Explanation:

- The Teletype element renders the enclosed text in teletype font. This means that the text will be monospaced to look like a typewriter font (browsers will often use Courier font by default).

  e.g.: <P> All the vowels on my typewriter are broken. I keep typing in a standard phrase and it comes out like this: <TT> Th qck brwn fx jmps vr th lz dg </TT>. I think I need a type writer repairman. </P>



2.5 LISTS

Lists provide methods to lay out item or element sequences in document content.
There are three main types of lists:

- Unordered lists
- Ordered lists
- Definition lists

Ordered lists are numbered in some fashion, while unordered lists are bulleted. Definition lists consist of a term followed by its definition. Both ordered and unordered lists require start and end

tags as well as the use of a special element to indicate where each list item begins (the /LI tag). In addition to these three types of lists, HTML also allows two other types of lists that are not very commonly used:

- Directory lists
- Menu lists

```
Tag: <LI>

Tag Name:     List Item
```

Explanation:

```
<LI> is a singleton tag.
```

- It is a child element that is used to create a list item in an ordered list, unordered list, menu list, or dir list.

Attributes: TYPE = (DISC ¦ SQUARE ¦ CIRCLE) or (1¦a¦A¦i¦I)

- When an ordered list <OL> is used, the <LI> element will be rendered with a number. One can control that number's appearance with the <TYPE> attribute.
- Similarly, inside an unordered list <UL>, one can control the type of bullet displayed with <TYPE>. VALUE = number changes the count of ordered lists as they progress.

```
Tag: <UL> ... </UL>

Tag Name: Unordered List (Block-level element)
```

Explanation:

- It creates an unordered list with bullets preceding each list item. Unordered lists can be preceded by any one of several bullet styles: a closed circle, an open circle, or a square.

Attributes:

COMPACT : Renders the list as compactly as possible by reducing line leading and spacing.

TYPE : (DISC ¦ SQUARE ¦ CIRCLE)

- The type of bullet can be suggested with the <TYPE> attribute. The CIRCLE attribute value is used for a hollow bullet, the DISC type creates a solid bullet, and the SQUARE attribute value renders a solid block.
- The default appearance for a list is with a disc.
- You can use an optional </LI> end tag at the end of each list item.
- The following example generates two separate lists:

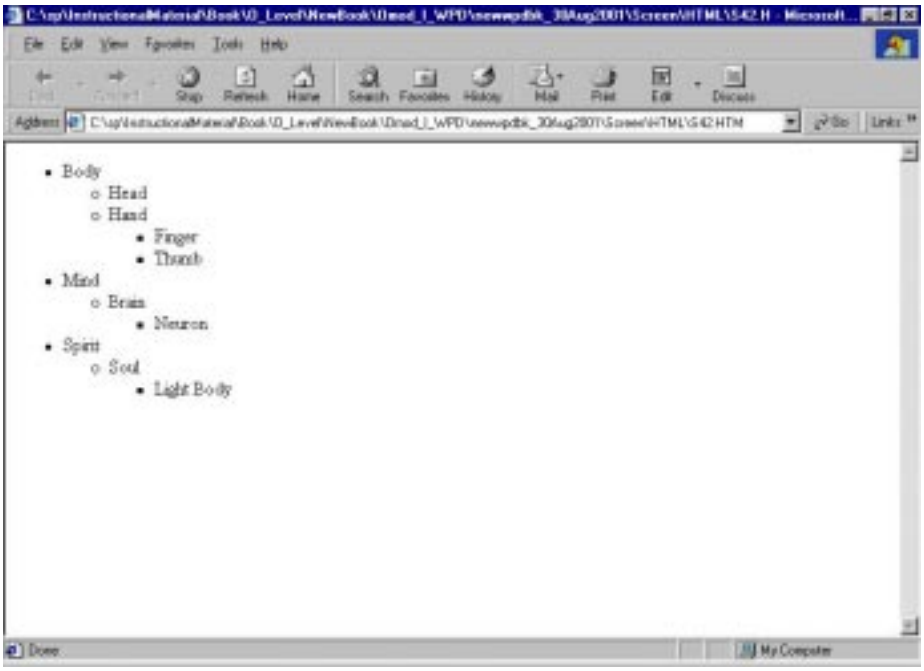
```
<TITLE> Two Shopping Lists </TITLE>
<BODY>
<UL>
<LI> Eggs
<LI> Milk
<LI> Apples
<LI> Razor Blades
</UL>
<UL TYPE = "SQUARE">
<LI> Hammer
<LI> Screwdriver
```

```
<LI  TYPE  =  "DISC">  Screws
<LI  TYPE  =  "CIRCLE">  Chainsaw
</UL>
```

NOTE Some browsers do not recognize the TYPE attribute at all, and most browsers do not recognize that the TYPE attribute can be used with the <LI> tag. In fact, even IE 4 does not recognize it.

- One important aspect of lists is that you can nest one list inside another to create a sublist. The default appearance of the sublists will vary from the main list, with the first sublist using circle bullets, and the next nested list using squares. For example:

```
<UL>
<LI>  Body
                <UL>
                <LI>  Head
                <LI>  Hand
                        <UL>
                        <LI>  Finger
                        <LI>  Thumb
                        </UL>
                </UL>
<LI>  Mind
                <UL>
                <LI>  Brain
                        <UL>
                        <LI>  Neuron
                        </UL>
                </UL>
<LI>  Spirit
                <UL>
                <LI>  Soul
                        <UL>
                        <LI>  Light  body
                        </UL>
                </UL>
</UL>
```




```
Tag: <OL> ... </OL>

Tag Name:     Ordered List
```

Explanation:

- These tags are used to create ordered lists. Ordered lists are identical in behaviour to unordered lists except they use numbers instead of bullets, and you can use an attribute to start numbering at a number other than one.

Attributes: `TYPE = (1| a | A | i | I)`

- Changes the style of the list

```
TYPE = "1" (Arabic numbers)

TYPE = "a" (Lowercase alphanumeric)

TYPE = "A" (Uppercase alphanumeric)

TYPE = "i" (Lowercase Roman numbers)

TYPE = "I" (Uppercase Roman numbers)
```

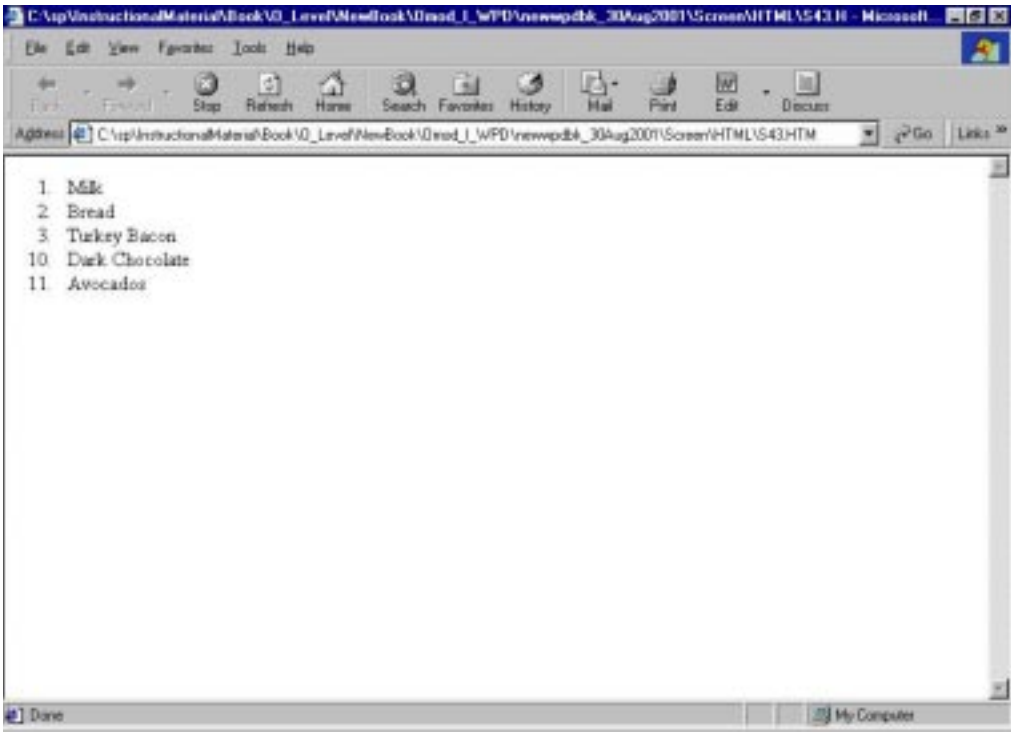
[Arabic Numbers are the default]

COMPACT : Renders the list as compactly as possible by reducing line leading and spacing.

```
START = "Value"
```

- Indicates where the list numbering or lettering should begin.
- In ordered lists, the <LI> tag can use the VALUE attribute to force to make a particular list item to have a certain number.

```
e.g.: <OL>
      <LI> Milk
      <LI> Bread
      <LI> Turkey Bacon
      <LI VALUE = "10"> Dark Chocolate
      <LI> Avocados
      </OL>
```




```
Tag:  <DT>

Tag  Name:     Definition  Term
```

Explanation:

- The <DT> tag is a singleton tag. It is a child element and can only be used in a definition list element. It creates a term that can be defined in a definition list.

```
Tag:  <DD>

Tag  Name:     Definition
```

Explanation:

- It is a singleton tag which marks the definition/description for a term in a glossary list. It is used for glossaries or other lists in which a single term or line needs to be associated with a block of indented text.

```
        Tag:  <DL>  ...  </DL>

Tag  Name:     Definition  List
```

Explanation:

- Definition lists require a start tag (<DL>) and end tag (</DL>) and two special elements: one for definition terms (the <DT> tag) and one for definition (the <DD> tag). The list is rendered without bullets.

Attributes:

COMPACT : Renders the list as compactly as possible by reducing line leading and spacing.

```
e.g.:  <DL>
       <DT>  Term  A
       <DD>  Definition  of  Term  A
       <DT>  Term  B
       <DD>  Definition  Term  B
       </DL>

Tag:  <DIR>  ...  </DIR>

Tag  Name:     Directory  List
```

Explanation:

- This block-level element marks unbulleted list of short elements, such as filenames.

```
e.g.:  <DIR>
       <LI> Item 1
       <LI> Item 2
       <LI> Item 3
       </DIR>
```

```
Tag: <MENU> ... </MENU>
Tag Name:    Menu List
```

Explanation:

- It encloses a menu list in which each element is typically a word or short phrase that fits on a single line.

  This list is rendered more compactly than most other list types.

Attributes: `COMPACT`

- Renders the list as compactly as possible by reducing line leading and spacing.

```
e.g.:  <MENU>
       <LI> Sourdough
       <LI> Butter milk
       <LI> Rolls
       </MENU>
```

- For both directory and menu lists, the only item that should be contained is a list item element (the <LI> tag).

```
<STYLE>
```

Student Activity 3

1. What are presentation tags ?
2. Describe various presentation tags.
3. Name various types of lists that can be entered in an HTML document.
4. Differentiate between <UL> and <OL> tags.
5. Which three tags let you create the definition lists ?

2.6 ADDING PICTURES

A page of all text and no graphics can be a little boring. To spice up your Web page, consider adding some pictures. Where do you get pictures to add to your Web page? Use clip art, create your own pictures with a drawing program or graphics program, load pictures from a digital camera, or use a scanner to import a drawing or photograph. There are also many Web sites that offer images free for downloading.

To add a picture, you use the <IMG> (image) tag, entering the tag at the place in the BODY section of the Web page where you want the graphic to appear. You use the SRC (source) attribute to specify the name of the file that contains the picture that you want to be displayed on the Web page, like this:

```
<IMG SRC="picture.gif">
```

Before you add a picture to your page, you need to determine where the picture will be stored. The filename is either an absolute pathname or a relative pathname.

2.7 ABSOLUTE VS RELATIVE PATHNAMES

An absolute pathname includes the full pathname of the file. This means that if you move your files or if you change your directory (folder) names, you have to edit every <IMG> tag in every HTML file that contains the absolute pathname. For that reason, this naming convention is not recommended.

A relative pathname indicates the pathname of the image file relative to the pathname of the HTML file. This is the recommended naming convention for graphics files. For example, if your image file is stored in the same directory as your HTML file, you can use just the image filename in your <IMG> tag, with no pathname, as in the example in the previous section. If the image file is in the same directory as your HTML file, but in a subdirectory, include the subdirectory name in the <IMG> tag, like this:

```
<IMG  SRC="images  /  picture.gif">
```

If the image file is stored one directory level up from your HTML file, use two dots (..) in the pathname to move up a directory level, as follows:

```
<IMG  SRC="../picture.gif">
```

Some Web authors like to store their frequently used graphics in a directory called images or pix, so that all the <IMG> tags in all the pages in the rest of the directories of the Web site can refer to one set of graphics files.

2.8 IMAGE ATTRIBUTES

You can add the following attributes to the <IMG> tag to adjust the picture and control how text flows around the picture:

Height and width control the size (in pixels) at which the graphic appears on the Web page. These attributes are options; use them only if you do not like the default size and need to resize the picture. The Web browser that displays the Web page adjusts the height and width of the graph to the sizes that you specify. When you use the HEIGHT and WIDTH attributes, make sure that you keep the same proportions as the original graphic; if you do not, the picture looks like you s- t-r-e-t-c-h-e-d it – either horizontally or vertically. Resizing a graphic to be larger than the original is rare. The larger the number of pixels, the bigger the picture. For example, the following <IMG> tag displays the picture in the file picturegif as 30 by 50 pixels, regardless of the size of the stored picture:

```
<IMG  SRT="picture.gif"HEIGHT="30"  WIDTH="50">
```

Using small (but legible) graphics on Web pages is best, because they land faster than large graphics. To make a graphic small, be sure to use a graphics program to reduce the size of the file. Do not just change HEIGHT and WIDTH attributes, which do not change the file size (or speed up downloading time), only the way it is displayed by the browser.

ALIGN controls how text flows around the graphic. Align has five possible values:

TOP places one line of text even with the top of the image.

MIDDLE places one line of text at the middle of the image.

BOTTOM places one line of text even with the bottom of image. Top, Middle, and Bottom are useful when you have a single line of text that you want placed next to a graphic.

LEFT places the graphic on the left side of the page, with your text paragraph wrapped around the left side of the graphic. LEFT and RIGHT are useful when you have paragraphs of text that you want to wrap around a graphic. For example, the following tag displays a picture on the left side of the Web page, with the surrounding text wrapped around its right side:

```
<IMG  SRC="picture.fig"  ALIGN="LEFT">
```

HSPACE and VSPACE control the amount of white space around the image. Both values are indicated in pixels. HSPACE sets the amount of space at the left and right of the image; you use this attribute to control the distance between the text that is wrapped around your graphic and the

graphic itself. VSPACE sets the amount of space above and below the graphic. The following tag inserts a picture with 25 blank pixels to either side, and 10 blank pixels above and below the picture:

```
<IMG SRC="picture.gif" HSPACE="25" VSPACE="10">
```

BORDER indicates that a border should be placed around the image, and controls the width of the border. The width is measured in pixels. The next tag inserts a picture with a border that is three pixels wide (a heavy line):

```
<IMG SRC="picture.gif" BORDER="3">
```

ALT contains the text that appears while the picture is loading, or if a user has a browser that does not display graphics, or if the user has opted not to load graphics when viewing Web pages. Always include ALT attributes in all image tags, to make your Web site accessible to vision impaired users, who use special software to read the text on the screen. For example,

```
<IMG SRC="picture.gif" ALT="Book cover of this week's selection">
```

Note that you can use as many or as few of the attributes as you need. Only the SRC attribute, which specifies the filename, is necessary. All attributes are placed in the same <IMG> tag, separated by spaces. For example, if you decided to use all the attributes listed here, your <IMG> tag might look like this:

```
<IMG SRC="picture.gif" JEOGJT="30" Width="50" ALIGN="LEFT" HSPACE="25"
VSPACE="10" BORDER="3" ALT="Book cover of this week's selection">
```

When readers click the "list of book club members" link, they jump to the Book Club Members location. You can add an anchor name to jump to an existing anchor in any Web page, like this:

```
<A HREF=*http://www.sample.com/index.html #members">list of book
club members</A>
```

2.9 TEXT TAGS

Text tags provide a logical structure for content.

Using Block-Level Elements to Structure the Document

Block-level elements contain blocks of text and can organize text into paragraphs. Block-level elements, according to the W3C standard for HTML 4.0, should have a line-break or paragraph break before and after the element. Some common block-level elements are:

```
headings (<H1> and </H1>, and <H2> and </H2>), paragraphs (<P> and
</P>), horizontal rules (<HR>), and centred text (<CENTER> and </
CENTER>).
```

Using Text-Level Elements

Text-level elements mark up bits of text, in order to change the appearance or function of that text. Text-level elements do not start new paragraphs; instead, text-level elements are usually used within a paragraph. So, text-level elements:

- can define character appearance and function.
- must be nested in the proper order.
- do not generally cause paragraph breaks.
- can contain other text-level elements but not block-level elements.

```
Tag: <ABBR> ... </ABBR> or <ACRONYM> ... </ACRONYM>

Tag Name:     Abbreviation
```

Explanation:

- It is a text-level element that indicates that the enclosed text is an acronym or abbreviation.

Attributes: TITLE = "text"

- Provides the expanded version of the abbreviation and appears in a pop-up box when the mouse is over the acronym.

```
e.g.: I spy for the <ACRONYM TITLE = "Federal Bureau of
Investigation"> FBI </ACRONYM>.

Tag: <BLOCKQUOTE> ... </BLOCKQUOTE>

Tag Name:     Blockquote
```

Explanation:

- This tag or block-level element is used for quoting one or more paragraphs from another source. Navigator and IE indent the entire block of quoted text.

```
Attributes:   CITE = "text"
```

- Provides information about the source of the quote.

```
e.g.:   <P> From the Bridges of the New York City, Queensboro
Ballads by Levi Asher (http: //www.levity.com/brooklyn/index.htm):
<BLOCKQUOTE>
```

It is not just that everybody hates the city, the more time I spend with these people the more I understand that they hate everything. Or at least they seem to, because it is the culture of Wall Street to never show joy.

```
</BLOCKQUOTE>

Tag: <BR>

Tag Name:     Line Break
```

Explanation:

- It is a text-level element (an empty element, consisting of the
 tag) which forces a line break in HTML text.

  Attributes: CLEAR = (LEFT¦ALL¦RIGHT¦NONE)

- LEFT inserts space that aligns the following text with the left margin directly below a left-aligned floating image.
- ALL places the following text past all floating images.
- RIGHT inserts space that aligns the following text with the right margin directly below a right-aligned floating image.
- NONE is the default, which does nothing.

```
e.g.:  Hello <BR>
       World
```

Line-breaks are useful for addresses and short items.

```
Tag: <CITE> ... </CITE>

Tag Name:     Short Citation
```

Explanation:

- It is a text-level element used to indicate that enclosed text is a citation (titles of excerpts, quotes, references) from another source.
- Text within <CITE> and </CITE> is usually rendered in italics.

```
e.g.:  <P> I have read and reread <CITE> Moby Dick
       </CITE> but I still can not make heads nor tails of it. </P>
       Tag: <CODE> ... </CODE>
       Tag Name:     Code
```

Explanation:

- It is a text-level element used to enclose programs or samples of code to offset them from normal text and is usually displayed in a monospaced font.

```
e.g.:  <P> To use the automatic date feature in Excel, just enter
<CODE> = Date ( )</CODE> into a cell </P>.

Tag: <DEL> ... </DEL>

Tag Name:     Delete Section
```

Explanation:

- It marks text that has been deleted from a previous version of the Web document.

  Attributes: CITE = "URL"

- Points to another document that describes why the text was deleted.

```
DATETIME = YYYY-MM-DDThh:mm:ssTZD
```

- Marks the time when you changed the document. The attribute's value confirms to the IS08601 time/date specification. The abbreviations shown above refer to the following date and time information:

YYYY	=	The year.
MM	=	The two-digit month – for example, "03" for March.
DD	=	The day.
T	=	The beginning of the time section.
hh	=	The hour, in military time (0-23 hours), without pm specifications.
mm	=	The minute.
ss	=	The second.
TZD	=	The time zone.
Z	=	The Coordinated Universal Time (CUT).
+hh:mm	=	The local time that is hours (hh) and minutes (mm) ahead of the CUT.
–hh:mm	=	The Local time that is hours (hh) and minutes (mm) behind the CUT.

Tag: <INS> ... </INS>

Tag Name: Inserted Section

Explanation:

- Identifies sections of a Web page that have been inserted in revision.

Attributes:

- Same as that of the <DEL> ... </DEL> tag.

  <INS CITE = "URL"> or <DEL CITE = "URL">

  <INS DATETIME = "DATE & TIME"> or <DEL DATETIME = "DATE & TIME">

- <DEL>... </DEL> and its companion tag <INS> ... </INS> were created to show revisions to web pages.

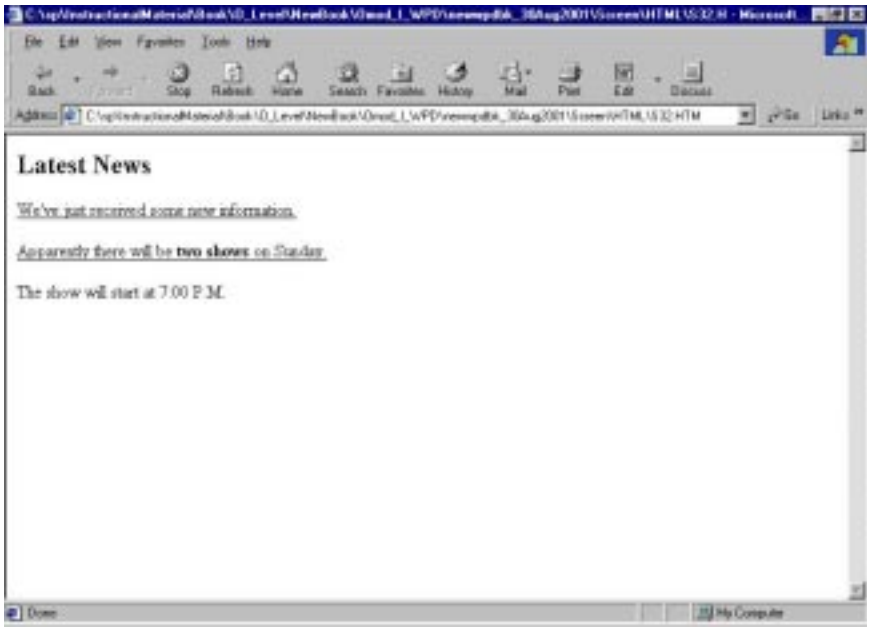
- Both these elements are special hybrid elements. They are unique in this respect - they are the only two elements used in the body of an HTML document that are not text-level or block-level elements.

```
Example:      <H2> Latest News </H2>

              <INS DATETIME = "1998-04-22TII:38:00 07:00"

              CITE = "http://www.tori.com/updatelog.htm">

              <P> We've just received some new information.
```




```
              <P> Apparently there will be <STRONG> two shows </<
              STRONG> on Sunday.

              </INS>

              <P> The show will Start at 7:00 P.M.
```

- This code marks the middle two paragraphs as new. They were changed on April 22, 1998 at 11:38 a.m.; information about the change can be found at the URL listed in the CITE attribute.

```
Tag: <DFN> ... </DFN>

Tag Name:     Defined Term
```

Explanation:

- It is a text-level element used to mark terms that appear for the first time in the Web document. These definitions are often in italics so the user can identify the first occurrence of the term.

```
e.g.: <P> It is not strange that <DFN>SGML</DFN> (Standard
Generalized Markup Language) is so similar to HTML.</P>.
```

  By marking your definitions, this way, special software programs can define an index or glossary for your document.

```
Tag: <EM> ... </EM>

Tag Name:     Emphasis
```

Explanation:

- A text-level element that adds emphasis to enclosed text.

```
Example: <P> I simply <EM> must </EM> get your recipe for chili,
karen Doason ! </P>

Tag: <KBD> ... </KBD>

Tag Name:     Keyboard Text
```

Explanation:

- It is a text-level element that marks text to be entered by the user at the keyboard.
- It changes the type style for all the text it contains, typically into a monospaced font like those used in character mode computer terminal displays.

```
e.g.:  <P> To start the program, hit the <KBD> S </KBD> key and
press the <KBD> Carriage Return </KBD>, then hold onto your hat!!</
P>

Tag: <P> ... </P>

Tag Name:     Paragraph
```

Explanation:

- The paragraph element's <P> tag marks the beginning of a paragraph. The end of a paragraph can be marked with </P>. It is a block-level element.

Attributes: `ALIGN = (LEFT|CENTER|RIGHT|JUSTIFY)`

```
Tag: <PRE> ... </PRE>

Tag Name:     Preformatted Text
```

Explanation:

- This block-level element forces the browser to display the exact formatting, indentation, and white space that the original text contains. This is valuable in reproducing formatted tables or other text, such as code listings.

Attributes: `WIDTH = number`

- This specifies the maximum number of characters for a line and allows the browser to select an appropriate font and indentation setting.

```
Example:      <P> morning wind
              my hair moves
              with the clouds and trees
                     <PRE> morning wind
              my hair moves
                     with the clouds and trees </PRE>
              morning wind
                     my hair moves with the clouds and trees.

Tag: <Q> ... </Q>

Tag Name:     Short Quotation
```

Explanation:

- This element is used to highlight short quotation from outside resources. The quote element is very similar to the blockquote element; the main difference is that since the quote element is not block-level, it does not start a new paragraph. Instead, it is used within a paragraph to mark a quotation.

Attributes: `CITE = "text"`

- Provides information about the source of the quote.

  Example: <P> Churchill said, <Q> "We have chosen shame and will get war," </Q> but he wasn't talking about 1066.</P>

```
Tag: <SAMP> ... </SAMP>

Tag Name:     Sample Text
```

Explanation:

- This text-level element uses the <SAMP> and </SAMP> tags to indicate sample output text from a computer program. Like the keyboard element, the sample element's text is often rendered in a monospaced font; and multiple spaces are collapsed. The keyboard element is used for text that a user must enter, whereas the sample element is used for text that a computer generates in response to a user's action.

```
e.g.: <P> Instead of giving me the expected results, the computer
kept printing <SAMP>  All work and no play makes Jack a dull boy </
SAMP> over and over again. I am not sure what it  means. </P>

Tag: <STRONG> ... </STRONG>

Tag Name:     Strong Emphasis
```

Explanation:

- It is a text-level element that provides strong emphasis for key words or phrases within normal body text, lists, and other block-level elements. Text within a strong element is usually rendered as bold or given a strident pronunciation by a text-to-speech reader.

```
e.g.:   <P> I swear, if they do not give me that raise <STRONG>
tomorrow </STRONG>, I       quit. </P>

Tag: <SUB> ... </SUB>

Tag Name:     Subscript
```

Explanation:

- This text-level element specifies that the enclosed text should be rendered in subscript, slightly lower than the surrounding text.

```
e.g.: This line of HTML Code contains the chemical formula for
water:

       We all need H<SUB>2</SUB>0

Tag: <SUP> ... </SUP>

Tag Name:     Superscript
```

Explanation:

- This element renders the enclosed text in superscript (a bit higher than regular text).
- This element is also useful for mathematical formulas.

Example: Here's Einstein's most famous equation:

```
       E = MC <SUP> 2 </SUP>

       Another good use of the <SUP> TM </SUP> today |

Tag: <VAR> ... </VAR>

Tag Name:     Variable text
```

Explanation:

- This text-level element marks up the variables used in computer programs, or the parts of a computer command chosen by the user.
- The text is usually rendered as monospaced, and like the keyboard element, multiple spaces are collapsed.

```
e.g.: <P> The formula for the <VAR> distance travelled </VAR> (in
miles) is <VAR> Speed </VAR> (in miles per hour) multiplied by <VAR>
time </VAR> (in hours). </P>.
```

Displaying Wallpaper in the Background

The background of your page can be either an image or a specific colour. To set an image as the background of your page, use the BACKGROUND attribute and place the name of your image file within quotes:

```
<BODY  BACKGROUND="image.gif">
```

Like tags, attribute names can appear in upper or lowercase. For the filename, be sure to use the same capitalization that is used in the actual filename. When a browser displays the page, the image in the file that you specify is tiled to fill the background of the Web page, that is, it is repeated across and down the page.

Choosing a Background Colour

Instead of using wallpaper, you can specify a solid background colour for your Web page. Use the BGCOLOR attribute in the <BODY> tag. For the colour, you can enter either a hexadecimal value that represents the colour, or one of 16 standard colour names. The colour names are the following: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. To indicate a standard colour, use this tag:

```
<BODY  BGCOLOR="BLUE">
```

Using a hexadecimal value is safer, if you want your colours to be read by a wide variety of browsers. The hexadecimal value is six characters that represent the amount of red, green, and blue in the colour. The first two characters indicate the amount of red, the next two characters the amount of green, and the final two characters the amount of blue. For example, this code specifies light aqua:

```
<BODY  BGCOLOR="#99FFFF">
```

Student Activity 4

1. How will you insert an image the named *elephant.jpg* at the very top of a web page ?
2. Name various types of alignments available for images.
3. How will you place a border around an Image ?
4. What can you do with text-level elements ?
5. Describe the function of following tags :

   (a) <CITE> (b) <BLOCKQUOTE>

   (c) <CODE> (d) <DEL>

   (e) <INS> (f) <DEN>

   (g) <EM> (h) <KBD>

   (i) <Q> (j) <STRONG>

   (k) <SUB> (l) <SUP>

6. What is the difference between BACKGROUND and BGCOLOR attributes of <BODY> tag ?

2.10 INTRODUCTION TO FORMS

Forms are the only method of two-way communication between Web pages and Web sites. Getting feedback is where HTML forms come into play. HTML supports a rich variety of input capabilities to let you solicit feedback.

Most new browsers – Netscape Navigator or Communicator 4.0 (and higher) and Microsoft Internet Explorer (Windows 95 version 3.0.3 and higher), plus NCSA Mosaic and its variants – already include HTML 4.0 level forms support, but other browsers do not.

Determining Form Content

The first step in developing a form is determining which information to include and how to present it, that is, how to break it down into manageable pieces. You then need to ensure that visitors can easily provide the information you want from them, which means that your form needs to be both functional and visually appealing.

Information Issues

When deciding which information to include and how to break it down, consider your purposes for creating the form. Begin by answering the following questions:

- What information do I want? Customer contact information?
- Why will visitors access the form? To order something on-line? To request information?
- What information can visitors readily provide? Contact information? Previous purchases?
- How much time will visitors be willing to spend filling out the form?

After determining what information you want and what information your visitors can provide, break the information into the smallest chunks possible.

Usability Issues

Usability refers to how easily your visitors can answer your questions. Some usability guidelines to consider when you are creating forms are as follows:

Group Similar Categories

When you group similar categories, the form appears less daunting, and visitors are more likely to fill it out and submit it.

Make the Form Easy

- Whenever possible, provide a list from which visitors can choose one or more items. Lists are easy to use, and they result in easy-to-process information.
- Ask visitors to fill in only a small amount of text. This takes minimal time, and it provides you with data that is fairly easy to process.

NOTE Many visitors are likely to ignore a request that requires them to enter lots of text.

Provide Incentives

- Provide visitors with incentives to fill out the form and submit it. Studies show that a penny or a stamp included in mailed surveys often significantly improves the response rate. Consider offering a chance in a drawing for a free product, an e-mailed list of tips and tricks, or a discount on services.

Design Issues

A well designed form, helps and encourages visitors to give you the information you want. A good form is visually appealing, graphically helpful, and consistent with the remainder of the site. Here are some guidelines:

- Use headings to announce each new group of information. This helps visitors move easily through the form.
- Be sure to visually separate groups. This makes the forms easier to use because sections become shorter and easier to read through.
- Use text emphases to draw the audience to important information.
- Specify how visitors are to move through the form. Do not make your visitors scroll horizontally to access information. Consider making a narrow, longer form rather than a wider, shorter form to accomodate those who have lower monitor resolution.

- Use arrows to direct visitors through the page. This can help visitors move through the page in a specified order.
- Be sure that it is clear which check boxes and fields go with the associated descriptive information. Use line breaks and spacing to clearly differentiate.
- Specify which fields are optional.
- Use a background image. Be sure, that the image does not outweigh the content and that the text adequately contrasts with the image.
- Make all the text entry fields the same width and put them on the left if you have a vertical column of name and address information; this way all the text will align vertically and look much better.

Creating Forms

Forms have two basic parts:

- The part you can see (that a visitor fills out).
- The part you can not see (that specifies how the server should process the information).

When adding forms support to a Web page, you must include special tags to solicit input from users. You also include tags to gather input and ship it to your Web server. Here is how this works:

- On a particular Web page, you include tags to set up a form and solicit input from users. This essentially amounts to filling out the form that you supply.
- After users fill out your form, they can then direct their input to the program running on the Web server that delivered the form.
- The information collected from a form can be:
    i. Written to a file.
    ii. Submitted to a database, such as Informix or Oracle.
    iii. E-mailed to someone in particular.

Setting the <Form>Environment

The two key attributes within the <FORM> tag are METHOD and ACTION. Together, these attributes control how your browser sends information to the web server and which input-handling program receives the form's contents.

ACTION = "..." Indicates the program on the HTTP server that will process the output from the form.

METHOD = "..." Tells the browser how to send the data to the server, with either the POST method or the GET method.

e.g.: <FORM METHOD = "POST" ACTION = "http://www.xmission.com/cgi-bin/cgiemail/~ejray/asr/ejray-asr-mail.text">

In this case, the http://www.xmission.com/cgi-bin/cgiemail part of the ACTION = line points to the program itself, and the following part (/~ejray/asr/ejray-asr-mail.txt) is the server-relative path to the file.

Understanding Widgets

Forms consist of several types of widgets (they are also called controls), which are fields you can use to collect data:

- Submit and Reset buttons send the form information to the server for processing and return the form to its original settings.
- Text fields are areas for brief text input. Use these for several word responses, such as names, search terms, or addresses.

- Select lists are lists from which visitors can choose one or more items. Use them to present a long but finite list of choices.
- Checkboxes allow visitors to select none, one, or several items from a list. Use them to elicit multiple answers.
- Radio buttons give visitors an opportunity to choose only one item.
- Textareas are areas for lengthy text input, as in open-ended comments.

NOTE In general, radio buttons, checkboxes, and select lists are all better choices for accepting input than textareas.

Submit and Reset Buttons

The first step in creating a form is to insert the <FORM> tags and add Submit and Reset buttons. Submit and Reset buttons allow visitors to submit information and clear selections.

Basic Form Tags

Tag/Attribute	Use
<FORM>	Marks a form within an HTML document.
<INPUT TYPE = "SUBMIT"	Provides a submit button for a form. The VALUE = "...">
	Value = attribute produces text on the button.
<INPUT TYPE = "IMAGE"	Provides a graphical submit button. The attribute indicates the image source file, and the BORDER =
Name = "POINT" SRC = "..."	attribute turns off the image border.
SRC = BORDER = 0>	
<INPUT TYPE = "RESET"	Provides a reset button for a form. The Value = attribute VALUE = "...">produces text on the button.

Traditionally, the Submit and Reset buttons go at the bottom of the form immediately above the closing </FORM> tag. The following example creates a Submit and Reset button by using the <INPUT> tag, the TYPE = attribute, and the VALUE = attribute.

```
<HR WIDTH = 80% SIZE = 8 NOSHADE>

<FORM>

<INPUT TYPE = "SUBMIT" VALUE = " Submit">

<INPUT TYPE = "RESET" VALUE = "Start Over">

</FORM>
```

If you want to use an image for your submit button, substitute the following code for the submit button.

```
<INPUT TYPE = "IMAGE" NAME = "POINT" SRC = "Submitbutton.gif"
BORDER = 0>
```

The TYPE = "IMAGE" attribute specifies that an image will be used to click on and submit the form. The NAME = "POINT" attribute specifies that the x, y coordinates where the mouse is located will be returned to the server when the image is clicked. Finally, the SRC = and BORDER = attributes work just as they do with regular images – they specify the URL of the image and turn off the border.

Text Fields

A text field is a blank area within a form and is the place for visitor supplied information.

Input Field Tag and Attributes

Tag/Attribute	Use
<INPUT>	Sets an area in a form for visitor input.
TYPE = "..."	Sets the type of input field. Possible values are TEXT, PASSWORD, CHECKBOX, RADIO, FILE, HIDDEN, IMAGE, SUBMIT, AND RESET.
NAME = "..."	Processes form results.
VALUE = "..."	Use this attribute with radio buttons and checkboxes because it does not accept any other input. You can also use this attribute with text fields to provide initial input.
SIZE = "n"	Sets the visible size for a field. Use this attribute with text input fields.
MAXLENGTH = "n"	Sets the longest set of characters that can be submitted.

An example to add a text field to an existing form:

```
<FORM>

<INPUT TYPE = "TEXT" NAME = "firstname" SIZE = "30" MAXLENGTH =
"30">

</FORM>
```

Guidelines for Including Multiple Text Fields

i. As a rule, forms are much more attractive if the fields are aligned.

ii. Here are some guidelines to follow when you include multiple text fields in your form:

- Place the fields at the left margin of your page, followed by the descriptive text.
- Set the text fields to the same size, when appropriate.
- As you add descriptive labels, remember to also add line breaks (
 or <P>) in appropriate places.
- Optionally, add a VALUE = attribute to the text input tag to "seed" the field with a value or to provide an example of the content you want.

Radio Buttons

A radio button is a type of input field that allows visitors to choose one option from a list. Each choice is mutually exclusive – choosing one excludes the remainder. For example:

```
<P>

Please choose the most appropriate statement.

<BR> <INPUT TYPE = "RADIO" NAME = "buying" VALUE = "regular">

I regularly purchase items on-line.

<BR> <INPUT TYPE = "RADIO" NAME = buying" VALUE = "sometimes">

I have on occasion purchased items on-line.

<BR> <INPUT TYPE = "RADIO" NAME = "buying" VALUE = "might"  CHECKED>

I have not purchased anything on-line, but I would consider it.

<BR> <INPUT TYPE = "RADIO" NAME = buying" VALUE = "will not">

I prefer to shop in real stores.
```

Use the same NAME = attribute for all radio buttons in a set. Browsers use the name attribute on radio buttons to specify which buttons are related and therefore which ones are set and unset as a group. Add the attribute CHECKED to one of the items to indicate the default selection.

Checkboxes

Each checkbox works independently from others; visitors can select or deselect any combination of checkboxes. Using checkboxes is appropriate for open questions or questions that have more than one " right" answer. In most browsers, checkboxes appear as little squares that contain a checkmark when selected. For example:

```
<P> I'm interested in (choose all that apply):

<BR> <INPUT TYPE = "CHECKBOX" NAME = "hiking"

VALUE = "hiking" > Hiking

<BR> <INPUT TYPE = "CHECKBOX" NAME = "mbiking"

VALUE ="mbiking"> Mountain Biking

<BR> <INPUT TYPE = "CHECKBOX" NAME = "camping"

VALUE="camping" > Camping

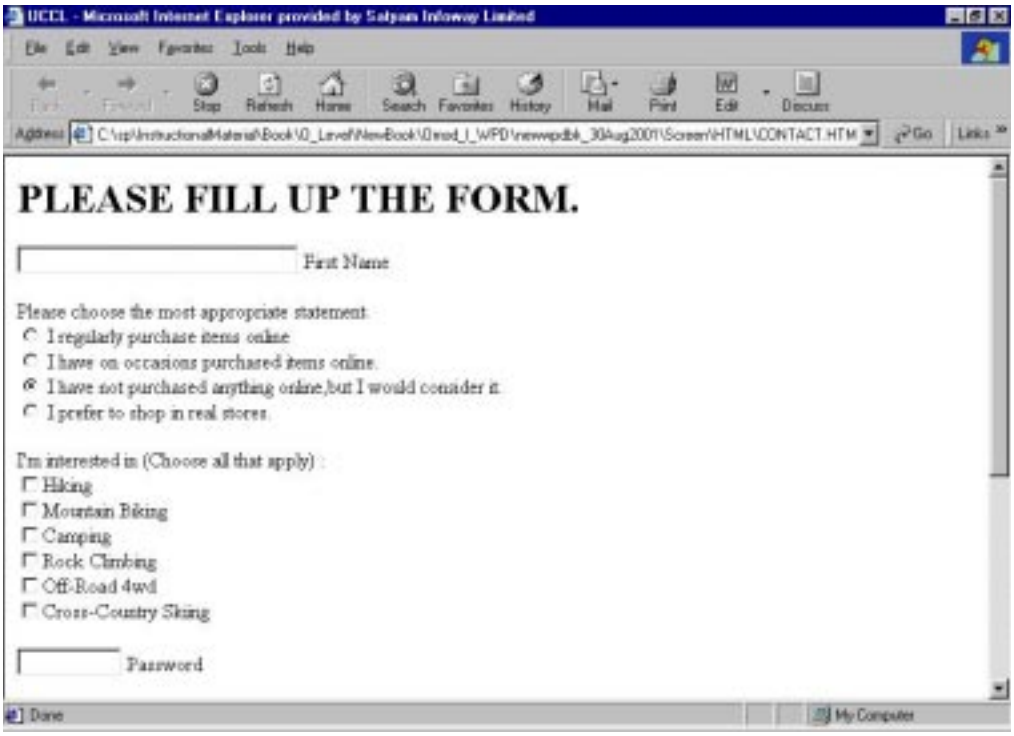
<BR> <INPUT TYPE = "CHECKBOX" NAME = "rock"

VALUE = "rock" > Rock Climbing

<BR> <INPUT TYPE = "CHECKBOX" NAME = "4wd" VALUE = "4wd" > Off-Road
        4wd

<BR> <INPUT TYPE = "CHECKBOX" NAME = "ccskiing" VALUE =
"ccskiing">Cross

Country Skiing
```



Password Fields

Password fields are similar to text fields, except the contents of the field are not visible on the screen. Password fields are appropriate whenever the content of the field might be confidential – as in passwords. Example to establish a password field:

```
<INPUT TYPE = "password" NAME = "newpass" SIZE = "10" MAXLENGTH =
"10">
```

When viewed in the browser, each typed character appears as an asterisk (*).

Hidden Fields

Hidden fields are not visible to the visitors. They are, recognized by the program receiving the input from the form and can provide useful additional information. Suppose a program cgiemail is used to process the form; it accepts a hidden field to reference a page shown after the customer completes and submits the form. The cgiemail program requires a hidden field such as the following:

```
<INPUT TYPE = "hidden" NAME = "success"

VALUE = "http://www.xmission.com/~ejray/asr/asrmaildone.html">
```

The TYPE = "HIDDEN" attribute keeps it from being shown, and the NAME = and VALUE = attributes provide the information that cgiemail expects. Hidden fields can go anywhere in the form, but it is usually best to place them at the top, immediately after the opening <FORM> tag.

File Fields

HTML also supports a special input field, a file field, to allow visitors to upload files. If you want visitors to submit information – say, a picture, a scanned document, a spreadsheet, or a word – processed document–they can use this field to simply upload the file without using FTP or e-mailing the file. After verifying that the server on which you will process your forms supports file uploads, you can implement this feature as follows:

Please post this photo I took in your gallery !

```
<INPUT TYPE = "FILE" NAME = "filenew" SIZE = "20" ACCEPT = "image/
*">
```

The values for the ACCEPT = attribute are MIME types. If you accept only a specific type, such as image/gif, you can specify that. If you take any image file, but no other files, you can use image/*. Finally, you can also provide a list of possible types you accept, separated by commas:

```
<INPUT TYPE = "FILE" NAME = "filenew" SIZE = "20" ACCEPT = "image/
gif, image/jpeg">
```

This code results in a text area plus a button that allows visitors to browse to a file, when rendered in most browsers.

Text Areas

Textareas are places within a form for extensive text input. One of the primary uses of textareas is to solicit comments or free form feedback from visitors.

Textarea Tags and Attributes

Tag/Attribute	Use
<TEXTAREA>	Sets an area in a form for lengthy visitor input. Initial content for the textarea goes between the opening and closing tags.
NAME = "..."	Establishes a label for an input field. The Name = attribute is used for form processing.
ROWS = "n"	Sets the number of rows for the visible field.
COLS = "n"	Sets the number of columns for the visible field.

Example:

```
<TEXTAREA NAME = "comments" COLS = "40" ROWS = "5"> Please type any
additional comments here.

</TEXTAREA>
```

Select Fields

Select fields are some of the most flexible fields used in developing forms because you can let visitors select single and multiple responses. Select fields, can either provide a long (visible) list of items or a highly compact listing.

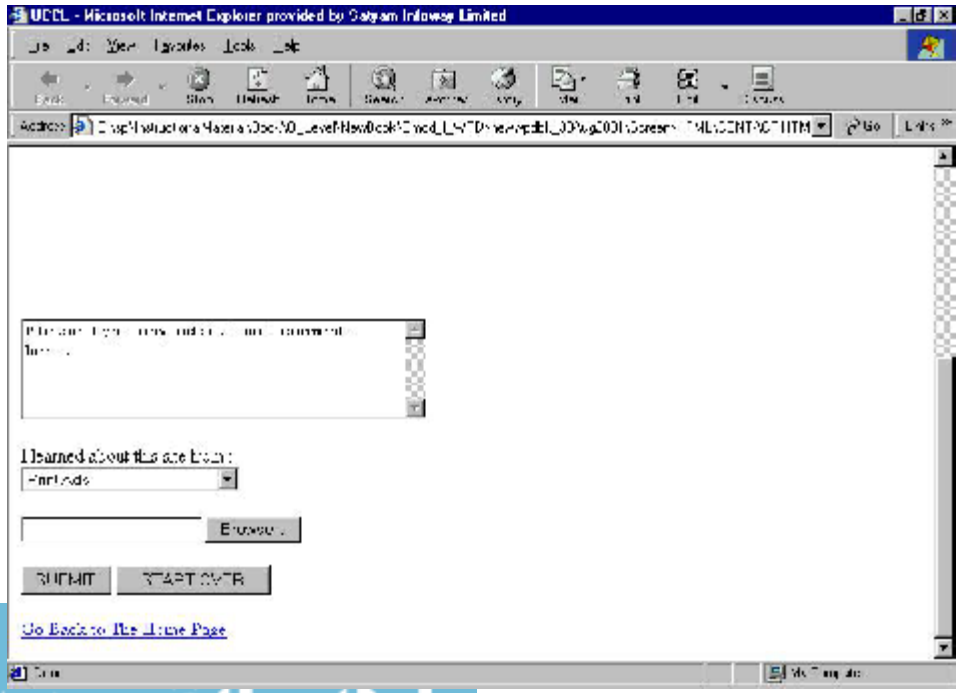
Select Field Tags and Attributes

Tag/Attribute	Use
<SELECT>	Sets an area in a form for a select field that can look like a drop-down list or a larger select field.
Name = "..."	Establishes a label for an input field. The NAME = attribute is used for form processing.
SIZE = "n"	Sets the visible size for the select field. The default (1) creates a drop-down list. You can change the default if you want more options to be visible.
MULTIPLE	Sets the select field to accept more than one selection.
<OPTION>	Marks the items included in the select field. You will have an <OPTION> tag for each item you include. The closing tag is optional.
VALUE = "..."	Provides the content associated with the NAME = attribute.
SELECTED	Lets you specify a default selection, which will appear when the form in loaded or reset.

For example:

I learned about this site from:

```
<SELECT NAME = "referral" MULTIPLE>
        <OPTION VALUE = "print" SELECTED> Print Ads
        <OPTION VALUE = "visit"> In-Store Visit
        <OPTION VALUE = "rec"> Friend's Recommendation
        <OPTION VALUE = "internet"> Sources on the Internet
        <OPTION VALUE = "other"> Other
</SELECT>
```



Student Activity 5

1. What are forms ?
2. Describe various points to be considered to develop a form.
3. How will you create a form ?
4. Describe <FORM> tag and its various attributes.
5. Describe <INPUT> tag and its various attributes.
6. How can you include multiple text fields in a form?

2.11 INTERACTIVE LAYOUT WITH FRAMES

A Frame is a rectangular region inside the browser window which can displays a web page. A web page can contain more than one frame. Thus, using frames one can display more than one web page in a single web page in the browser.

IN the early days of HTML technology only a single page could be seen at a time in a browser. Frames have been added to the HTML to overcome this limitation by dividing the browser window into number of HTML documents.

Frames allow the user to arrange text and graphics into rows and columns of text and graphics just like tables. Unlike a table cell, however, any frame can contain links that change the contents of other frames or itself. One frame could display an unchanging table of content while the other frames change based on which link are made. This feature provides tremendous flexibility in achieving dynamic character inside a web page.

Frames allow users to change the contents of on a portion of a web page being displayed while keeping other areas unchanged. The dynamic behavior offered by frames can be exploited to enhance a web page in variety of ways.

Frame capable of displaying a web page each can be grouped into what is called a frameset. A web page that displays framesets contains nothing but frameset tags and is aptly called a frameset document. Frames are introduced in a web using frame and frameset tags discussed below.

Creating a frameset document

Before you create a frameset document create the content of each frame as an ordinary HTML pages. Having done that a frameset document is created. A frameset document actually has no content. It only tells the browser which page to load, and how to arrange them in the browser window among different frames. For example, suppose we have the following three HTML pages:

```
One.html

<html>

<head>

<title>Vehicles</title>

</head>

<body>

<p>This is the first page to be displayed.</p>

<table border="1" cellpadding="0" cellspacing="0" style="border-
collapse: collapse" bordercolor="#111111" width="79%"
id="AutoNumber1">

  <tr>

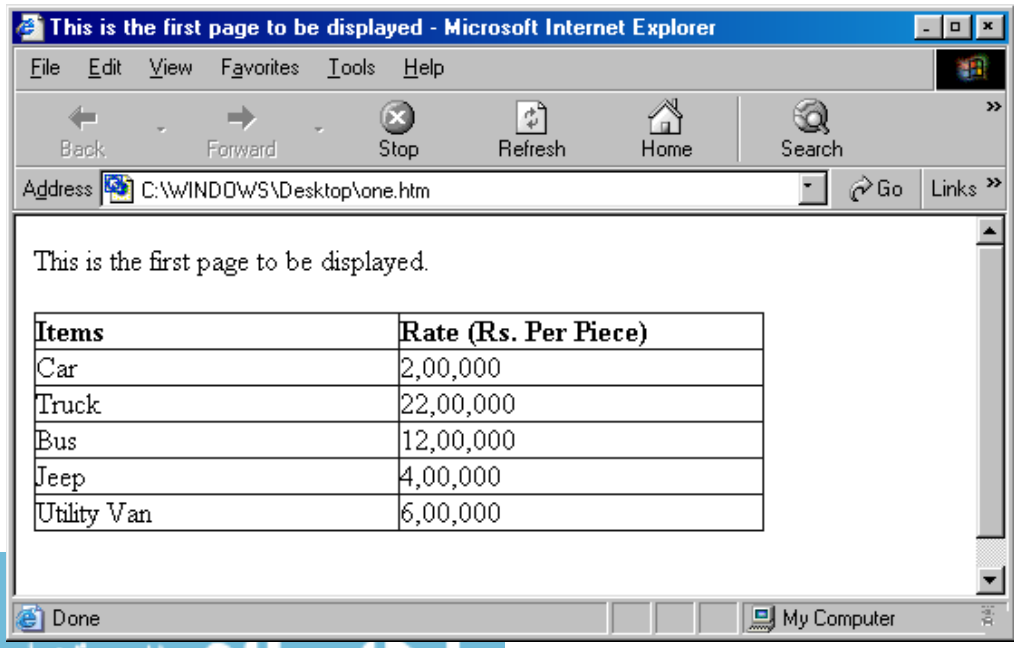
    <td width="50%"><b>Items </b> </td>

    <td width="50%"><b>Rate (Rs. Per Piece)</b></td>

  </tr>
```

```
<tr>
      <td  width="50%">Car</td>
      <td  width="50%">2,00,000</td>
   </tr>
   <tr>
      <td  width="50%">Truck</td>
      <td  width="50%">22,00,000</td>
   </tr>
   <tr>
      <td  width="50%">Bus</td>
      <td  width="50%">12,00,000</td>
   </tr>
   <tr>
      <td  width="50%">Jeep</td>
      <td  width="50%">4,00,000</td>
   </tr>
   <tr>
      <td  width="50%">Utility  Van</td>
      <td  width="50%">6,00,000</td>
   </tr>
</table>
<p> </p>
</body>
</html>
```

The page when opened in the browser looks like a shown below:



Two.html

```
<html>

<head>

<title>Second Page</title>

</head>

<body>

<p>This is the second page to be displayed.</p>

<table border="1" cellpadding="0" cellspacing="0" style="border-
collapse: collapse" bordercolor="#111111" width="79%"
id="AutoNumber1">

  <tr>

     <td width="50%"><b>Items </b> </td>

     <td width="50%"><b>Rate (Rs. Per Piece)</b></td>

  </tr>

  <tr>

     <td width="50%">Pen</td>

     <td width="50%">20</td>

  </tr>

  <tr>

     <td width="50%">Pencil</td>

     <td width="50%">10</td>

  </tr>

  <tr>

     <td width="50%">Sign pen</td>

     <td width="50%">32</td>

  </tr>

  <tr>

     <td width="50%">Marker pen</td>

     <td width="50%">40</td>

  </tr>

  <tr>

     <td width="50%">Paint brush</td>

     <td width="50%">60</td>

  </tr>

</table>

<p> </p>

</body>

</html>
```

The page when opened in the browser looks like a shown below:

```
Three.html

<html>

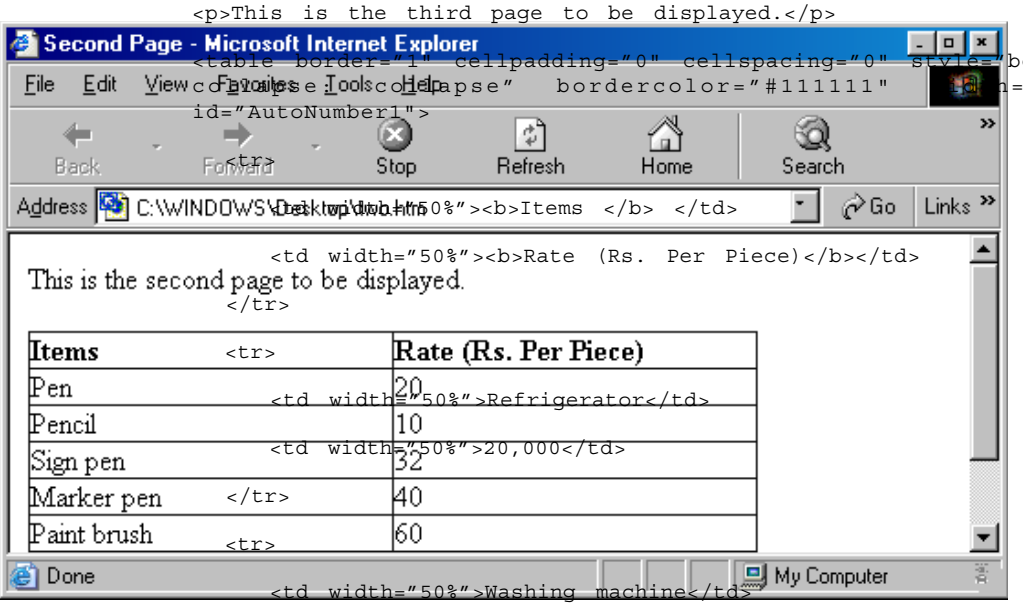
<head>

<title>Third  Page</title>

</head>

<body>

<p>This  is  the  third  page  to  be  displayed.</p>
<table  border="1"  cellpadding="0"  cellspacing="0"  style="border-
collapse"  bordercolor="#111111"  width="79%"
id="AutoNumber1">
    <tr>
        <td  width="50%"><b>Items  </b>  </td>
        <td  width="50%"><b>Rate  (Rs.  Per  Piece)</b></td>
    </tr>
    <tr>
        <td  width="50%">Refrigerator</td>
        <td  width="50%">20,000</td>
    </tr>
    <tr>
        <td  width="50%">Washing  machine</td>
        <td  width="50%">10,000</td>
    </tr>
    <tr>
        <td  width="50%">Air  conditioner</td>
        <td  width="50%">32,000</td>
    </tr>
    <tr>
        <td  width="50%">Inverter</td>
```



```
        <td width="50%">14,000</td>
    </tr>
    <tr>
        <td width="50%">Computer</td>
        <td width="50%">34,000</td>
    </tr>
</table>
<p> </p>
</body>
</html>
```

The page when opened in the browser looks like a shown below:

Having done this, create a frameset document to arrange these pages into a single web page as shown below:

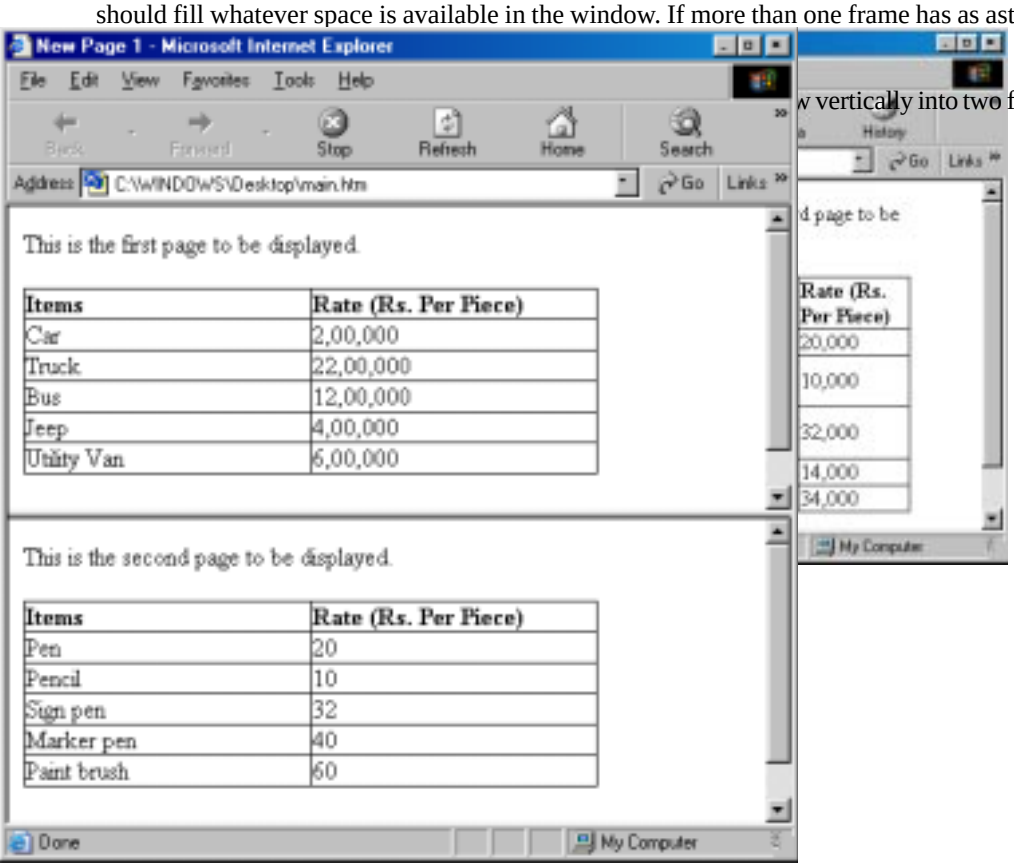
```
Main.html
<html>
<head>
<title>New Page 1</title>
</head>
<frameset cols="*,*,*">
    <frame name="left" src="one.htm">
    <frame name="center" src="two.htm">
    <frame name="right" src="three.htm">
    <noframes>
    <body>
    <p>This page uses frames, but your browser doesn't support them.</
p>
    </body>
    </noframes>
</frameset>
</html>
```

The <frameset> tag is what defines a group of frames. A frameset may have the following attributes:

Rows: This attribute indicates that the frames included into this frameset should be arranged in rows.

Cols: This attribute indicates that the frames included into this frameset should be arranged in columns.

The user should use Cols attribute instead of Rows if he desires frames to be side-by-side. One must specify the sizes of the Rows on Cols either as precise pixel values or as percentages of the total size of the browser window. The user can also use an astrick (*) to indicate that a frame should fill whatever space is available in the window. If more than one frame has as asterisk (*)



w vertically into two frames as

It is advised to safest approach to use % rather than exact pixel values to indicate the size of the rows and columns. For example to mate a left frame 35% of the width of browser window with a right frame taking up the remaining 65%. One should write as

< frameset cols = "35%, 65%" >

The <Frame>tag

Within the <Frameset> and </frameset> tags the user should have a <frame> tag indicating which HTML document to display in each frame. One need not to specify a closing </ Frame> corresponding to each of the frame> tags.

Include a SRC attribute in each < FRAME> tag with the address of the web page to load in that frame. The user can insert the address of an image file instead of a web page if the user just want a frame with a single image in it.

Link between frames and windows

When we give a frame a name with <FRAME NAME> attribute. We can made any link on the change the contents of that frame using the <A target> attribute. Consider the following modified example.

```
List.html

<html>

<head>

<title>Rates</title>

</head>

<body>

<p><a  target="bottom"  href="one.htm">Vehicle</a><br>

<a  target="bottom"  href="two.htm">Stationery</a><br>

<a  target="bottom"  href="three.htm">Household  equipment</a><br>

 </p>

</body>

</html>

Index.html

<html>

<head>

<title>New  Page  1</title>

</head>

<frameset  rows="23%,*">

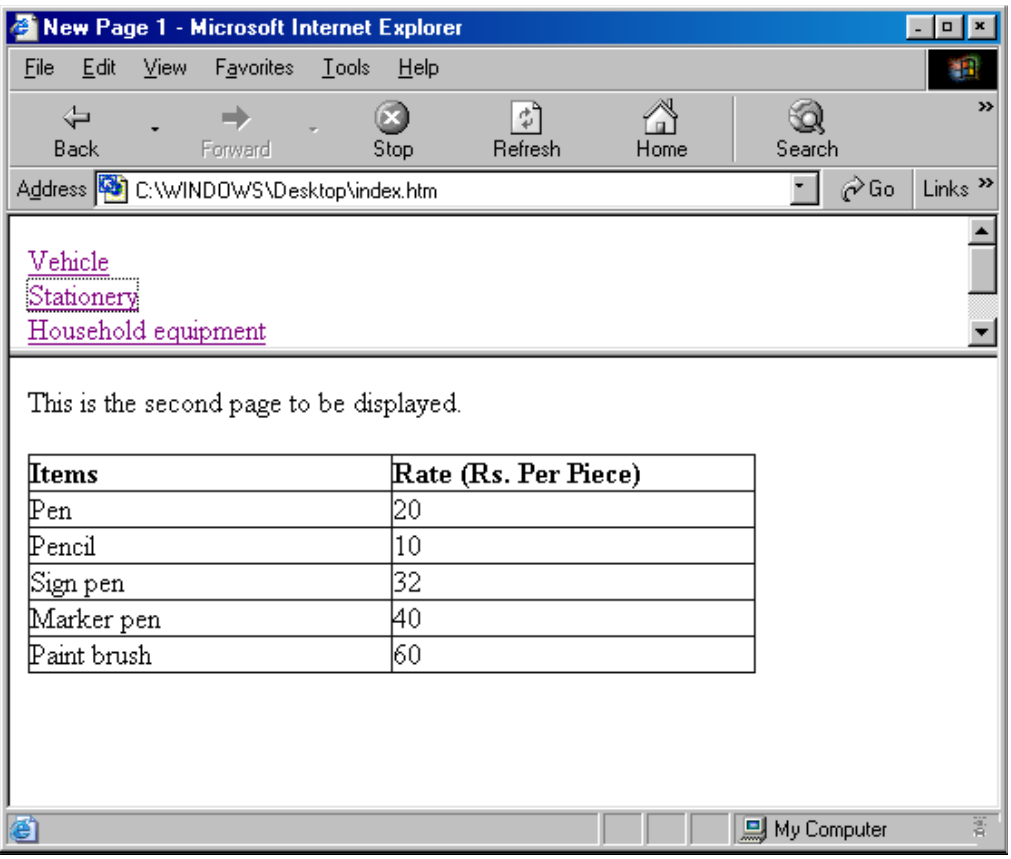
              <frame  name="top"  src="list.htm">

              <frame  name="bottom"  src="one.htm">

</frameset>

</html>
```

Now load the page Index.html into the web browser, it should like shown below:



Similarly, by clicking at Household equipment link you should see the following:

This way you can introduce a number of interactive frames in a single web page.

2.12 LINKING WEB PAGES AND PUBLISHING

Internal links behave identical to external links with one exception – you can use relative URLs for internal links. A relative URL simply drops the common part from the URL and lets the browsers automatically figure out the part that is missing. For example, instead of specifying <A HREF = "http://www.rupert.com/history.htm"> Rupert's History Page </A> just specify the part that's different from the current page's URL:

```
<A HREF = "history.htm"> Rupert's History Page </A>

Tag: <BASE> ... </BASE>

Tag Name:     Relative Addressing base
```

Explanation:

- It occurs within <HEAD> ... </HEAD> and establishes the URL basis for subsequent URL references in <LINK> or anchor statements in the document body.

Attributes:

- HREF = "URL"

  States the absolute or relative URL for the current document.

- TARGET = "window"

  There are four pre-defined names that can be targeted by an anchor. They all start with an underscore (_).

_blank For example:

```
<A HREF = "document.htm" Target = "_blank"> My document </A>
```

Clicking on My document would cause a new browser window to appear, containing the document.htm file.

_parent For example:

<A HREF = "document.htm" Target = "_parent"> My document </A>

Clicking on My document would cause the document.htm file to appear in the parent frameset.

_self For example, you have declared

<BASE TARGET = "worry"> to have links target a frame named "worry" by default. You can use:

<A HREF = "document.htm "Target =" _self ">My document </A>

When someone clicks on My document, the file document.htm will replace the current frame instead of the worry frame.

_top For example:

```
<A HREF = "document.htm" Target = "_top"> My document </A>
```

Clicking on My document would cancel all of the frames and replace the entire frameset with the document.htm file.

```
Tag: <Link> ... </Link>

Tag Name:     Link
```

Explanation:

- The link element indicates relationships between your documents and other documents or URLS.

Attributes:

- HREF = "URL"

  The address of the current link destination, accessible through normal web linkage mechanisms.

- MEDIA = SCREEN¦PRINT¦PROJECTION¦BRAILLE¦SPEECH¦ALL

  Identifies the ideal environment for the web page to be conveyed in. The default is ALL.

- REL = "text"

  It indicates a normal relationship to the document specified in the URL.

  ```
  e.g.: <LINK REL = "INDEX" HREF = "index.htm">
  ```

  This tag would let browsers know where they can find the main index for your Web site.

- REV = "text"

  It indicates a reverse relationship. The referenced document has the indicated relationship to the current document.

  e.g.: <LINK REV = "Index" HREF = "history.htm">

  This tag would let browsers know that there is a two-way relationship between history.htm and index.htm – that index. htm is the index of history.htm, and simultaneously, history.htm is indexed by index.htm.

- Target = "window"

  Specifies loading the link into the targeted window.

- Type = "text"

Specifies the Internet media type of linked resource. For example, the type for CSSI sheets is "TEXT/CSS".

Publishing Documents

Publishing means putting HTML documents on a Web server and telling people where to look for them. The exact process you will use for publishing documents depends on your situation. Some large organizations have well-defined publishing procedures; in such cases, you might simply fill out HTML forms and save your files in a specified folder. If your organization does not have procedures or if you are publishing on the Internet, you will need to upload your files, i.e., to copy files from your computer to a server.

Deciding where to store your pages is one of the final steps in creating your Web site plan. To make your site available to everyone on the Web, you need to publish the site on a Web server. You can either set up your own server or post your files to some one else's Web server.

Maintaining Your Own Server

Setting your own Web server is a very expensive option for publishing your site. It requires the following items:

- A computer capable of handling Web traffic, that is up and running 24 hours a day.
- Web server software.
- A dedicated, high-speed phone line (such as an ISDN or T1 line).
- An ISP that will set you up with a dedicated connection to the Internet.

Using Your ISP's Server

Most ISPs include a few megabytes of storage space free of charge with a dial-up account, and most offer additional space at reasonable prices. For most people storing Web files on their ISP's server is the most convenient and economical way to publish a site on the Web.

For each ISP, the file transfer procedures are different. Your ISP can give you the instructions that you need to transfer files to its Web server. As soon as the files are transferred, your site is available for viewing on the Web.

Using a Web Hosting Service

A Web hosting service is a company that rents space on their Web servers. Web hosting companies usually offer multiple Web servers, a fast connection to the Internet, domain hosting, frequent backups, unlimited access by the Web server (you) to update your pages, and use of standard CGI scripts, such as scripts that display counters that show how many visitors your page has had. Charges may be fixed, or they may depend on how much space your Web pages occupy and how many visitors you have.

To find a Web hosting service, start at Yahoo! *(http://www.yahoo.com)* and choose Business and Economy/companies / Internet Services / Web services / Hosting.

Publicizing Your Site

After your site is published on the Web and you have joined the global on-line community, how are you going to publicize the site? The first step is to register your site with some search engines, so that people doing on-line searches can find the site. Apart from advertizing your site by the word of mouth, here are some other ways to let other people know that your site is on the web. Use the method that is appropriate for your site:

- Get the URL of your site printed on your business cards, stationary, and in yellow pages.
- Add your URL to your signature block in e-mail and newsgroup messages.
- Include the URL in your return address whenever you send greeting cards or holiday cards.
- Have some bumper stickers printed showing your group or club's URL.

- Print the URL in your church bulletin each week.
- Post the URL on all school bulletin boards.

Testing Published Documents

Earlier in this chapter, we discussed testing documents on your local computer. Now it's time to test them in the real world, looking for the same issues addressed previously, as well as making sure that all the links work and that all the documents are transferred properly to the server. Additionally, at this stage, your goal is not only to look for layout, formatting, and proof reading errors, but also to get an accurate idea of what visitors will see when they access your pages. In particular, find out how fast pages load, how pages appear at various screen sizes and colour depths, and how different browsers display page elements.

Maintaining Documents

Maintaining HTML documents is the process of updating and revising existing pages, adding new pages, and deleting outdated pages. Regularly maintaining HTML documents is essential if you want visitors to keep returning to your site. Also regular maintenance helps make long-term maintenance less cumbersome. HTML documents contain two types of information:

- Static
- Dynamic

Static information remains constant. The company logo, most of the menus, and even product descriptions are examples of static information.

Dynamic information, on the other hand, must be changed or updated regularly. Prices, schedules, specific or timely information, and product lists are examples of dynamic information.

Publishing Tools

Web pages are coded using Hyper Text Markup Language (HTML) tags to specify the way text and graphics are displayed. When you create your own Web pages, you can either type the tags manually in a text editor or use a Web page editor to do the coding for you. These editors are also called Web publishing tools. There are number of Web page editors available for making Web pages. Web page editors are available as stand-alone applications or bundled into the Internet software package. Some popular editors are Netscape Composer (which is part of the Netscape Communicator suite), FrontPage and FrontPage Express (from Microsoft), PageMill (from Adobe), and HotDog Professional (from Sausage Software). All run under Windows 98 and some (including Netscape Composer) are also available for the Mac. This section describes FrontPage, in some detail, as an example of how a Web page editor works. The other editors work in a similar way.

Student Activity 6

1. Which attributes of <A> facilitates external linking ?
2. Which attributes of <A> facilitate internal linking ?
3. What do you mean by publishing documents ?
4. Give various options for publishing your website.
5. How will you test a published document ?
6. Why is it necessary to maintain HTML documents ?
8. What are publishing tools ? Give examples.

2.13 SUMMARY

- On world wide web, the documents are written in a special language called HTML (Hyper Text Markup Language).
- An HTML document must have <HTML>, <HEAD> and <BODY> tags, where <HEAD> and </HEAD> specify the header part and <BODY> and <//BODY> specify by the document body.

- Special characters <, > and & are printed using &lt ; > and & escape sequences.
- In HTML, lists can be of three types : unnumbered, numbered, and definition.
- The unnumbered lists are defined with <UL> tag, numbered lists are defined with <OL> tag. The items in these lists are defined with <LI> tag.
- The definition lists are defined with <DL> tag, definition terms are defined with <DT> tag and definition descriptions are defined with <DD> tag.
- Images can be inserted in text using <IMG> tag. Its source is specified with attribute, alternate text with **alt** attribute, alignment with **align** attribute and size with **height** and **width** attributes.
- Hyperlinks can be created using <A> tag. Linking to another document is called external linking. Linking within the same document is called internal linking.
- Various tags may be combined in HTML to have the desired effect.
- To make your site available to everyone on the web, you need to publish the site on a Web Server. This can be done by maintaining your own web server or posting your files to some one else's web server.

2.14 KEYWORDS

Tag : A coded HTML command.

Attribute : Special word, carrying special meaning, used inside an HTML tag.

Block-level Elements : Used to defined groups of text for a specific role.

Text-level Elements : Used for markup bits of text.

List : Provides methods to layout item or element sequences in document content.

Ordered Lists : Indented lists having numbers or letters in front of every list item.

Unordered Lists : Indented lists having a bullet symbol in front of every list item.

Definition Lists : Lists showing definition terms and definition descriptions.

External Linking : Linking to another web page.

Internal Linking : Linking to a section inside same web page.

Publishing : Putting HTML documents on a web server and telling people where to look for them.

Web hosting Service : A company that rents space on their web servers.

2.15 REVIEW QUESTIONS

1. Fill in the blanks :
   a. HTML is acronym for .....
   b. To define the basic font size, ... tag is used.
   c. To specify the background image for the document, background attribute is used with ..... tag.
   d. Two tags break the line flow. These are ..... and .....
   e. In <OL> and <UL>, the 8 type of bullet or number is defined by .....
   f. For linking to another web page, its URL is specified with ..... attribute of <A> tag.
2. Can you assign justified alignment to a paragraph in HTML ?
3. List and explain different attributes of body tag.
4. What is the difference between basefont and font tag ?

5. How would you indent a single word and put a square bullet in front of it ?

6. You plan to publish a CD-ROM disk containing HTML pages. How do you create a link from a page in the \guide folder to the \guide\mains\kolkata.htm page ?

Answers to Review Questions

1. (a) Hypertext Markup Language (b) BSEFONT (c) <BODY>
   (d) <HR>,
 (e) type attribute (f) href

2.16 FURTHER READINGS

Ed Titled ; *Foundations of Worldwide Web Programming with HTML and CGI* ; 1995, IDG Books Worldwide.

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java distributed Computing*.

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT

3

THE GENESIS OF JAVA

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Give a brief history of Java
- Define applets and applications
- Define bytecodes
- Understand various java buzzwords
- Define Java script
- Understand what's new in JDK 1.2

U N I T S T R U C T U R E

3.1 INTRODUCTION

This unit talks about the evolution of Java and its various features that have made it a popular programming language. Architecture Neutrality is one of the most importance characteristics of Java. Several other features of Java that have given it the status of an Internet language have also been discussed in this unit.

When the chronicle of computer language will be written, the following will be said: B led to C, C evolved into C++ and C++ set the stage for Java. Java is the first and foremost programming language. Creation of Java was driven by two elements in nearly equal measures viz.:

i. To adapt to the changing environments and uses.

ii. To implement refinements and improvements in the art of programming.

A Brief History of Java

In 1990, Sun Microsystems began a project called Green to develop software for consumer electronics. Gosling began writing software in C++ for embedding into such items as toasters, VCR's and Personal Digital Assistants (PDA's). The embedded software makes many appliances more intelligent. Gosling's solution to the problem of C++ was a new language called Oak. Finally in 1995, Oak was renamed Java. Since then, Java has been rising in popularity.

3.2 APPLETS AND APPLICATIONS

An applet is a Java program that appears embedded in a web document, just as graphics are. The Java applet runs when a Java enabled web browser, such as Netscape Navigator, loads it.

Application is a program that runs on your computer, under the operating system of that computer. This means that an application created by Java is more or less like one created in C or C++.

3.3 SECURITY

Security is probably the main problem facing Internet developers. Users are typically afraid of two things : confidential information being compromised and their computer systems being corrupted or destroyed by hackers. Java's built-in security addresses both these concerns.

When you use a Java compatible web browser, you can safely download Java applets without fear of virus infection.

3.4 BYTECODES

The key that allows Java to solve both the security and portability problems just described above is that the output of a Java compiler is not an executable code, rather it is a bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java run time system, which is called Java Virtual Machine (JVM). It is an interpreter for bytecodes.

Translating a Java program into bytecode makes it much easier to run a program in a wide variety of environments. Although the details of JVM will differ from platform to platform, all interpret the same Java bytecode.

When a program is interpreted, it generally runs substantially longer than it would run if compiled to executable code.

Sun has just completed its JIT (Just in Time) compiler for bytecodes which is included in Java2 release. When the JIT compiler is part of the JVM, it compiles bytecodes executable code in real time, on a piece by piece, demand basis.

3.5 JAVA BUZZWORDS

The fundamental forces that necessitated the extension of Java were portability and security. Java is

- Simple
- Secure
- Portable
- Object Oriented
- Robust
- Multi-threaded
- Architecture Neutral
- Interpreted

- Distributed
- Dynamic

Student Activity 1

1. Describe the history of Java
2. What are applets?
3. What is an Application?
4. What are bytecodes?
5. Give some features of Java.

3.6 SIMPLE

Java was designed to be easy for the professional programmer. It is easy to learn and can be used effectively. If you are an experienced C++ programmer, moving to Java will require very little effort.

3.7 MULTI-THREADED

A single threaded application has one thread of execution running at all times and such programs can do only one task a time.

A multi-threaded application can have several threads of execution running independently and simultaneously. These threads may communicate and cooperate and will appear to be a single program to the user.

3.8 ARCHITECTURE NEUTRAL

The central issue for Java designers was that of code longevity and portability. One of the main problems facing programmers is that no guarantee exists that if you write a program today, it will run tomorrow even on the same machine. Operating system upgrades, processor upgrades and changes in core system resources can all combine to make a program malfunction. It is basically 'write once; run anywhere, any time, forever'.

3.9 JAVA AND JAVASCRIPT

JavaScript is a separate programming language closely related to Java. It can be coded directly in an HTML document which makes the JavaScript source code part of the document itself. JavaScript is less powerful than Java but it gives the programmer a bit more control over the browser and it is used primarily to create dialog boxes and animations on web pages.

JavaScript does have a limited ability to call Java applet routines and alter Java applet variables. However, the core Java APD has no mechanism for calling JavaScript code or changing JavaScript variables.

3.10 NEW IN JDK 1.2

Java 1.2 adds even more enhancements, improvements or capabilities in Java 1.1. Some other completely new features that have been added are mentioned below:

Enhancements to Security, JavaBeans, Reflection and Performance

Numerous Java packages have received enhancements in Java 1.2. The security architecture incorporates policy based access control to enhance permission – management. JavaBeans add drag and drop support while reflection includes the ability to bypass security, specially when using object reflection.

Some performances have been improved, for instance, faster memory allocation, reduced memory usage for loaded classes and Just In Time compilers.

Java Foundation Classes

The Java Foundation Classes (JFC) encompass a broad range of enhancements. There is now support for advanced technologies with an Accessibility API, a Java 2-D API for enhanced Graphics and Imaging and Swing for a new set of GUI components in addition to the AWT components.

Collections

The Collections API makes working with groups of objects much easier. Prior to Java 1.2 you basically used the earlier Vector and Hashtable classes as well as the Enumeration interface. Here you can work with things like balanced trees, circular linked lists and simplified array sortings.

Student Activity 2

1. What is a multi-threaded application?
2. How can you say that Java is architecture neutral?
3. What is Java Script?
4. List some of the added features of JDK 1.2.

3.11 SUMMARY

- Java is an Object oriented language developed by Sun Microsystems. It is a simple, secure, portable, object-oriented, robust, multi-threaded, architecture neutral, interpreted, distributed and dynamic language.
- An applet is a Java program that appears embedded in a web document, just as graphics are and an application is a program that runs on your computer, under the operating system of that computer.
- Java Script is a separate programming language closely related to Java. It does have a limited ability to call Java applet routines and alter Java applet variables.
- Java 1.2 adds even more enhancements, improvements or capabilities in Java 1.1

3.12 KEYWORDS

Applet : A Java program that appears embedded in a web document, just as graphics are.

Application : A program that runs on your computer, under the operating system of that computer.

Bytecode : A highly optimized set of instructions designed to be executed by the Java run time system.

Java Virtual Machine (JVM) : An Interpreter for bytecode.

Multi-threaded Application : An application with several threads of execution running independently and simultaneously.

Java Script : A programming language closely related to Java with a limited ability to call Java applet routines and alter Java applet variables.

3.13 REVIEW QUESTIONS

1. Fill in the blanks :
   (a) _____ was renamed as Java.
   (b) _____ appears embedded in a web document.
   (c) JVM is an _____ for bytecode.

(d) Java is easy to _____and can be used _____.

(e) Java Script can be coded directly in an _____ document.

2. State true or false :

(a) Applet appears embedded in a web document.

(b) Translating a Java program into bytecode makes it much easier to run a program in a wide variety of environments.

(c) Java Script have a limited ability to call Java applet routines and alter Java applications.

(d) Java is neither secure nor portable programming language.

Answers to Review Questions

1. (a) Oak (b) applet (c) interpreter (d) learn, effectively (e) HTML

2. (a) true (b) true (c) true (d) false

3.14 FURTHER READINGS

E..Balaguruswami, *Programming with Java*, Tata McGraw Hill.

Herbert Schilelt, *The Complete Reference Java 2* –Tata McGraw Hill.

Herbert Schildt, *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

UNIT

4

AN OVERVIEW OF JAVA

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe an object
- Describe various features of object oriented programming
- Understand a simple Java program
- Understand how to compile and run Java Program

U N I T S T R U C T U R E

4.1 INTRODUCTION

As we start developing a software in the software systems model, the most important part in the above model is maintenance. As you know, all programs consist of two elements, code and data, i.e., some programs are written around 'what is happening' and others are written around 'who is being affected'.

To manage increasing complexity, the second approach called object oriented programming was conceived. This approach organizes a program around its data (that is, objects), and a set of well defined interfaces to that data. An object oriented program can be characterized as data controlling access to code. You will see that by switching the controlling entity to data, you can achieve several organizational benefits.

After studying this chapter, the students should be able to understand the concepts of Object Oriented Methodology Abstraction, encapsulation, Objects, Classes and Inheritance.

4.2 WHAT IS AN OBJECT

Objects are software bundles of data and related procedures. Software objects are often used to model real world objects that you find in everyday life. As the name implies, objects are the key concept in understanding object oriented technology. You can look around and see many examples of real world objects – your dog, your desk, your television set or your bicycle. These real world objects share two characteristics – they all have state and they all have behaviour. For example, dogs have state (name, colour, breed, hunger) and behavior (barking and fetching). Bicycles have state (current gear, current pedal etc.). The following illustration is a simple visual representation of a software object.

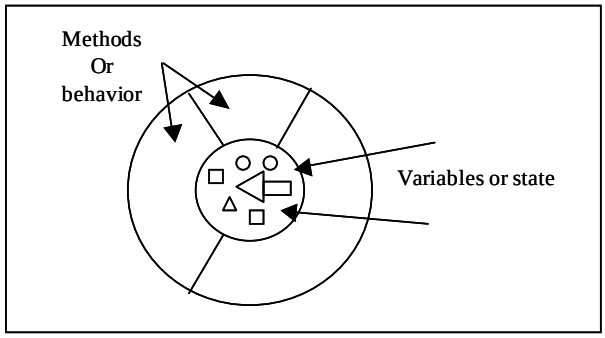


Figure 4.1 An Object

4.3 FEATURES OF OBJECT ORIENTED PROGRAMMING

Encapsulation

Everything that the software objects know (state) and can do (behavior) is expressed by the variables and methods within that object. A software object that models your real world bicycle will have variables that will include the bicycle's current state such as, its speed is 25 Kmph.

These variables and methods are formally known as instance variables and instance methods to distinguish them from class variables and class methods.

Figure 4.2 Your Bicycle

The software bicycle will also have methods to brake, change the pedal cadence and change gear. As you can see from figure 4.1 and 4.2, the object's variables make up the centre or nucleus of the object and the methods surround and hide the object's nucleus from other objects in the program. Packaging an object's variables within the protective custody of its method is called Encapsulation. Typically, encapsulation is used to hide important details from other objects. When you want to change gears on your bicycle, you don't need to know how the gear mechanism works, you just need to know which lever to move.

Thus, the implementation can change at any time without changing other parts of the program.

Inheritance

Classes inherit state and behaviour from their superclass. A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

Object oriented systems allow classes to be defined in terms of other classes. For example, mountain bikes, racing bikes and tandems are all subclasses of the bicycle class. Similarly, the bicycle class is the superclass of mountain bikes, racing bikes and tandems.

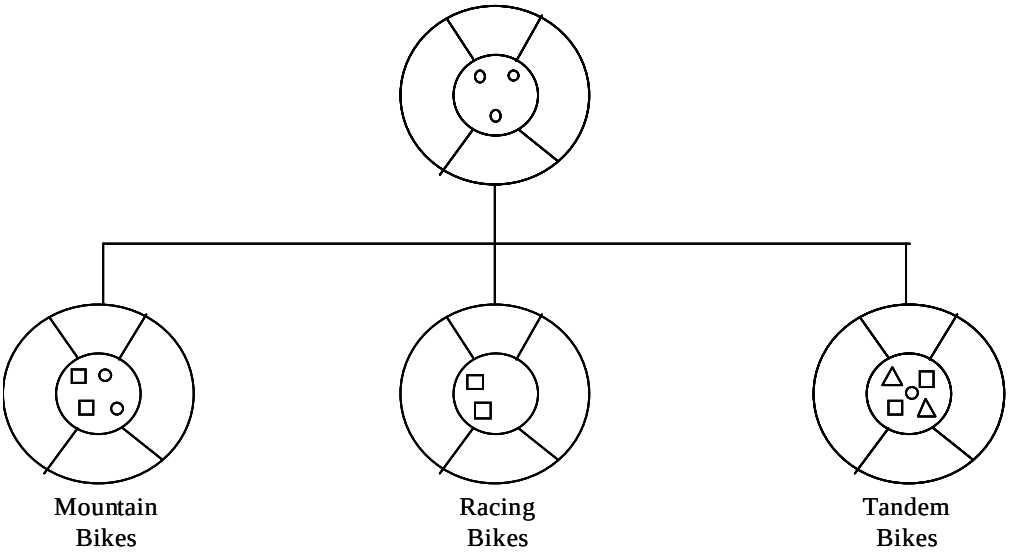


Figure 4.3 Hierarchy of Classes

Each subclass inherits state (in the form of variable declarations) from the superclass. Mountain bikes, racing bikes and tandems share some states : cadence, speed, and the like. Also each subclass inherits methods from the superclass.

Benefits of Inheritance

Abstraction

In Java, classes are used to categorize data in order to model real life systems. Abstraction is this process of categorizing data. Consider that a person is looking for a frame in an optician's shop. To be able to choose a frame from amongst the various types of frames available, he has to first identify the attributes he is looking for. Once he has identified the attributes, he has with him a category or class of frames. Similarly, to model any real life objects in OOPS, an 'object' has to be instantiated from a specific 'class'. This basic process of forming a class is known as 'Abstraction'.

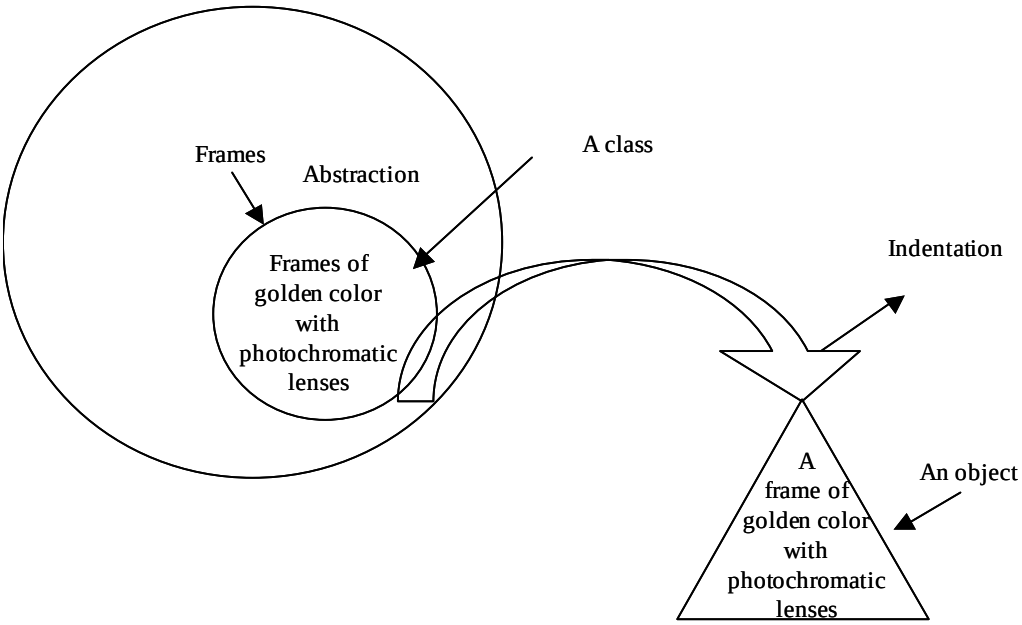


Figure 4.4 Abstraction

Student Activity 1

1. What is an Object?
2. What are the various features of object oriented programming?
3. Define Encapsulation.
4. Define Inheritance.
5 What are the benefits of Inheritance?

4.4 THE FIRST SIMPLE PROGRAM

Basic Java Application

```
1.  // This is my first program
2.  public class Myfirstprogram
3.  {
4.  public static void main (String args[ ])
5.  {
6.  System.out.println ("This is my first program");
7.  }
8.  }
```

We will explain this program step by step.

The first line is a single line comment which will be ignored by the compiler. Comments are used in programs so that you can understand what your program does. For multiple line comments use /* */.

In Java, everything should be in a class which is declared by the keyword class. Public is a keyword which makes this class available and accessible to any one.

Myfirstprogram is the name of the program and should also be the name of the Java file. The case of the file name should be the same i.e., Myfirstprogram.java.

Static keyword is used which signifies that this method will be called without creating an object of Myfirstprogram and this method will be invoked first and only once by JVM. Void is the return type of the main method, which means that this method is not going to return any value. The parameter that the main method takes is an array of string which is used to show command line parameters.

System.out.println method is used to print anything on the console.

Pair of { } braces define the scope of any method or class.

In the Java programming language, a statement is a single code terminated with the semicolon (;).

4.5 COMPILING

To compile and run the program, type the following instructions at the command prompt.

javac Myfirstprogram.java

java Myfirstprogram

Result / Output

This is my first program

javac keyword is used to compile the program and Java to run the program.

Student Activity 2

1. Write a simple Java program.
2. How will you compile and run a Java program?

4.6 SUMMARY

- Java is an Object-oriented language. Objects are software bundles of data and related procedures. Basic concepts of object oriented programming include encapsulation, inheritance and abstraction.
- A class represents a category of related entities and is called an object or instance of that class.
- A Java program can be compiled with javac and the output can be displayed through Java.

4.7 KEYWORDS

Paradigm: Organizing principle of a program. It is an approach to programming.

Class: A class represents a set of related objects, or a template for building objects.

Object: An instantiation of a class.

Abstraction: Act of representing essential features without including the background details or explanations.

Encapsulation: The wrapping up of data and functions into a single unit (class).

Inheritance: The process by which objects of one class acquire the properties of objects of another class.

Polymorphism: The ability to make more than one form.

Dynamic Binding: The linking of a procedure call to the code to be executed in response to the call.

Message: A request for execution of a procedure.

Operator Overloading: The process of making an operator to exhibit different behavior at different instance.

Function Overloading: The process of making a function to perform various different types of tasks.

4.8 REVIEW QUESTIONS

1. Fill in the blanks :
   a. _____ are the basic real life entities in an object oriented system.
   b. A _____ names a category of related entities or objects.
   c. _____ is a keyword which makes the class available and accessible to any one.
   d. To get an idea about what your program is trying to do use _____ .
2. State whether the following are true or false:
   a. Encapsulation is the capability to represent, denote and handle information at a higher level that is inherent to a computer or base language.
   b. Inheritance is a means for creating a new, more specific type from an existing more general type.
   c. Wrapping up of data and functions into a single unit (called class) is known as encapsulation.
   d. The Java compiler compiles the source file into bytecodes which are then interpreted by the JVM.

Answers to Review Questions

1.	(a) object	(b) class	(c) public	(d) comments
2.	(a) false	(b) true	(c) true	(d) true

4.9 FURTHER READINGS

E..Balaguruswami, *Programming with Java,* Tata McGraw Hill.

Herbert Schilelt, *The Complete Reference Java 2* –Tata McGraw Hill.

Herbert Schildt, *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

UNIT

5

DATA TYPES, VARIABLES AND ARRAYS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe various data types available in Java
- Understand how to declare literals and variables in Java.
- Describe how to type cast various data types
- Define arrays and its various types
- Define vectors
- Declare array in java.

U N I T S T R U C T U R E

5.1 INTRODUCTION

When working with computers, either for something as simple as writing a college paper or as complex as solving quantum theory equations, the single most important thing for the computer to do is diether programming language, Java supports its own set of data types.

5.2 DATA TYPES IN JAVA

Primitive Data Types

Java has eight primitive data types.

Reserved Word	Data Type	Size	Range of Values
byte	Byte length integer	1 byte	-2^8 to 2^7-1
short	Short integer	2 bytes	-2^{16} to $2^{16}-1$
int	Integer	4 bytes	-2^{32} to $2^{31}-1$
long	long integers	8 bytes	-2^{64} to $2^{63}-1$
float	Single precision	4 bytes	-2^{32} to $2^{31}-1$
double	Real number with double	8 bytes	-2^{64} to $2^{62}-1$
char	character (16 bit unicode)	2 bytes	0 to 216 -1
boolean	Has value true or false	A boolean value	true or false

Non Primitive

Datatypes which are built with the help of primitive data types but are not actually primitive are known as non primitive datatypes.

5.3 LITERALS

Literals are pieces of Java source code that indicate explicit values. For instance "Hello World !" is a string literal and its meaning is the words Hello World. Java has four different types of literals

- String
- Character
- Number
- Boolean.

String Literal

The string literal is always enclosed in double quotes. Java uses a String Class to implement strings, whereas C and C++ use an array of characters. For example

```
"Hello World".
```

Character Literal

Character literals are similar to string literals except that they are enclosed in single quotes and must have exactly one character. For example 'C' is a character literal that means the letter C.

Boolean Literal

Boolean literal can have either of the values: true or false. They do not correspond to the numeric values 1 and 0 in C and C++.

Numeric Literals

There are two types of numeric literals : integers and floating point numbers. For example 34 is an integer literal and it means the number thirty four. 1.5 is a floating point literal.

Student Activity 1

1. List primitive data types available in Java.
2. What are non primitive data types?
3. What are literals? Name its various types.
4. What is the difference between string literals and character literals?
5. Name various types of numeric literals available in Java.

5.4 CHARACTER

Characters in Java are a special set. They can be treated as either a 16 bit unsigned integer with a value from 0-655 35 or as a unicode character. The unicode standard makes room for the use of alphabets of many different languages. The syntax is

```
Char ch = 'b';
```

Boolean

A Boolean variable is not a number. It can have one of two values : true or false. There are two ways to set it.

For example

mybool is a variable To change the variable to false type

```
mybool = false;
```

The second way is to assign the same value as in another variable

```
mybool = mybool1;
```

You can also make the variable have a value based on the equality of other numbers

```
mybool = 6 > 7;
```

5.5 VARIABLE DECLARATION

Identifiers are the names of variables. They must be composed of letters, numbers, the underscore and the dollar sign ($). They cannot contain white spaces, identifiers may only begin with a letter, the underscore or a dollar sign. You cannot begin a variable name with a number. All variable names are case sensitive, for example myvar is not equal to MyVar.

Initialization

Variables can be assigned values through the syntax

```
myvar = 10; myvar = "Hello"; myvar = true;
```

Define Scope

There are three kinds of variables in Java– instance variables, class variables, local variables. Java does not have global variables (which can be used in all parts of a program). Instance variables and class variables are used to communicate information from one object to another and these replace the need for global variables. Local variables are used inside method definitions or even smaller blocks of statements within a method. The modifiers define the scope of variables. A variable declared without any access control modifier is available to any other class in the same package. To completely hide a variable from being used by another class use PRIVATE.

In some cases, you may want a variable in a class to be completely available to any other class that wants to use it, then it should be declared as public.

5.6 SYMBOLIC CONSTANTS

Variables are useful when you need to store information that can be changed as a program runs. If the value should never change during a program's run time, you can use a special type of variable called a constant. It is used when the value need not to be changed. Variables are also useful in defining shared values for all methods of an object. To declare a constant use the final keyword before the variable declaration. For example

```
final float Pi = 3.141592
```

5.7 TYPE CASTING

An int divided by an int is still an int and a double divided by a double is still a double. But what about an int divided by double or a double divided by an int? When doing arithmetic on unlike types, Java tends to widen the types involved so as to avoid loosing information.

The basic rule is that if any of the variables on the right hand side of an equal sign is double then Java treats all the values on the right hand side as doubles. If none of those values are double but some are floats, then Java treats all the values as floats. If there are no floats or doubles but there are longs, then Java treats all the values as longs. Finally, if there are no doubles, floats or longs, then Java treats everything as int.

However, if the right hand side may not be able to fit into the left hand side, then a series of operations takes place to chop the right hand side down to size. For a conversion between a floating point number and an int or a long, the fractional part of the floating point number is truncated. For example

```
int i = (int) 9.0/4.0
```

The result will be converted to integer.

Converting Strings to Numbers

```
int i = Integer. valueOf ("22").intValue( );

long l = Long.valueOf ("22").longValue( );

double x = Double.valueOf ("22").doubleValue( );
```

Automatic Conversion

When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:

- The two types are compatible
- The destination type is larger than the source type.

When these two conditions are met, a widening conversion takes place. For example an int type is always large enough to hold all valid byte values, so no explicit statement is required.

For widening conversions, the numeric types including integer and floating point types are compatible with each other. However, the numeric types are not compatible with char or boolean. Also char and boolean are not compatible with each other. Java also performs an automatic type conversion when storing a literal Integer Constant into variables of type byte, short or long.

Student Activity 2

1. What is a Boolean variable?
2. Give the syntax for variable initialization.
3. Name different types of variables available in Java.
4. What are symbolic constants?
5. What do you mean by type casting? Why does it required?
6. When does an automatic type conversion take place?

5.8 ARRAYS

An array is a group of like typed variables that are referred to by a common name. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index.

One Dimensional Array

A one dimensional array is essentially a list of like typed variables. To create an array you must first create an array variable of the desired type. The general form of a one-dimensional array declaration is

```
type var-name [ ];
int month-days [ ];
```

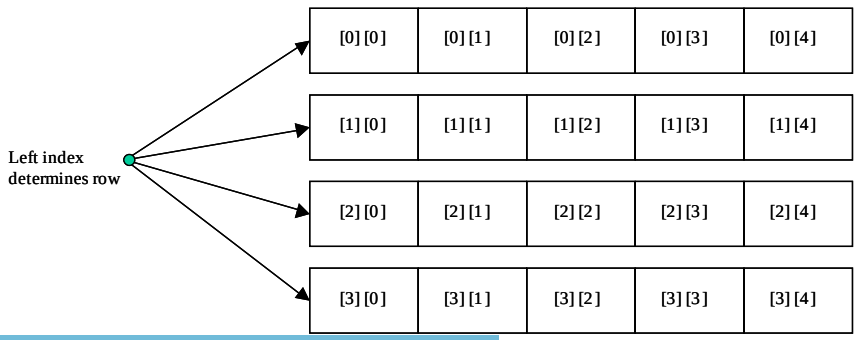
```
// Demonstrate a one dimensional array class array {
public static void main(String s[ ])
{
        int month-days[ ];
        month-days[0] = 31;
        month-days[1] = 28;
        month-days[2] = 31;
System.out.println ("march has" + month-days[3] + "days"];
}
}
class promote {
public static void main(String args[ ])
{
        byte b = 42;
        Char c = 'a';
        Short s = 1024;
        int i = 50000;
        float f = 5.67f
        double d = .1234;
        double result = (f*b) + (i/c) - (d*s);
        System.out.println (f*b) + "+" + (i/c) + " - " (d*x));
        System.out.println ("result = " + result);
}
}
```

Multidimensional Arrays

In Java, multimensional arrays are actually arrays of arrays. As you might expect, they look and act like regular multidimensional arrays. However as you will see there are a couple of subtle differences. To declare a multidimensional array variable, specify each additional index using another set of square brackets.

inttwoD[ ] [ ] = new int [4] [5];

This allocates a 4 by 5 array and assigns it to twoD.




```
int twoD[ ] [ ] = new int [4] [5];
```

5.9 VECTORS

Vector implements a dynamic array. It is similar to ArrayList but with two differences; vector is synchronized and it contains many legacy methods. The methods are as follows:

- Vector ( )
- Vector (int Size)
- Vector (int size, int incr)

The form creates a default vector which has an initial size of 10. The second form creates a vector whose initial capacity is specified by size. The third form creates a vector whose initial capacity is specified by size and whose increment is specified by incr.

Vector defines these protected data members

int capacityIncrement;

int elementCount;

object elementData[ ]k;

Some important methods are

- public void addElement (object Element)
- final int capacity( )
- final Object elementAt(int index)
- final int indexOf(object element)
- final object firstElement( )
- final void InsertElementAt(object element, int index)
- final void removeElementAt(int index)
- final void setSiz(int Size)
- String toString( )

5.10 ARRAY DECLARATION SYNTAX

There is a second form that may be used to declare an array

```
type [ ] varname;

int a1[ ] = new int [3];

int [ ] a2 = new int [3]
```

Student Activity 3

1. What is an array?
2. Describe various types of arrays.
3. Give the syntax to declare an array in Java.
4. What are vectors?

5.11 SUMMARY

- Datatypes in Java are strictly defined with aspect to their sizes, structures, usage and casting.

- These datatypes are light primitive datatypes like byte, int, char etc. and literals which indicate explicit values.
- Identifiers are names given to variables. Separators like period, parenthesis and semicolons are used to define the structure of the program.
- Casting between mixed datatypes of similar nature is performed implicitly and can also be stated explicitly.
- Array is a way of storing more than one same datatype values in a single variable. Arrays can be of two types – single and multidimensional.
- Vector is another way of storing similar datatypes. It has different methods to manipulate it.

5.12 KEYWORDS

Data: Unprocessed facts and figures.

Syntax: The standard format used to declare any statement, method etc. in a programming language.

Constants: Data items that never change their value during a program run.

Variables: Identifiers used to store a data value.

Integer: Whole number without any fractional part.

Real Numbers: Numbers having fractional part.

String Literal: A sequence of characters surrounded by double quotes.

Data types: Means to identify the type of data and associated operations for handling it.

Initialization: The process of giving initial values to variables.

Default Value: The value assigned to the Java variables by compiler if they are not initialized in the program.

5.13 REVIEW QUESTIONS

1. Fill in the blanks
   a. _____ are reserved identifiers that cannot be used to name user defined variables or methods.
   b. _____ method is used to force a conversion from one datatype to another.
   c. To convert a string into an int type, the function required is _____ .
   d. When data types are converted explicitly, the method is called _____ .
   e. Two main points for Java's automatic conversion are _____ and _____.
2. State whether the following are true or false:
   a. There is a conversion type defined between an int value to a long.
   b. A variable is a basic unit of storage in a Java program.
   c. Multidimensional array is not possible in Java.
   d. Strings in Java work as an array of characters.
   e. True and false are not 0 and 1.

Answers to Review Questions

1. (a) keywords (b) type casting (c) intValue()
   (d) automatic conversion (e) compatibility, size
2. (a) true (b) true (c) false (d) true (e) true

5.14 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *Java: The Complete Reference, J2SE TM,* 2005, Mc Graw-Hill Professional.

UNIT

6

OPERATORS IN JAVA

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe various types of operators available in Java
- Describe operator precedence in Java.
- Understand how to override operator precedence.

U N I T S T R U C T U R E

6.1 INTRODUCTION

Java provides a rich operator environment. Most of its operators can be divided into the following four groups: arithmetic, bit wise, relational and logical. Java also defines some additional operators that handle certain special situations. Each operator performs a specific task it is designed for. This unit describes some basic and important operators in Java. At last we will also talk about expressions in Java and their type conversion.

6.2 ARITHMETIC OPERATORS

Arithmetic operators are used in mathematical expressions in the same way as they are used in Algebra. The following table lists the Arithmetic operators.

Operator	Function
+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Modulus
+=	Addition Assignment

+ =	Subtraction Assignment
– =	Multiplication Assignment
* =	Division Assignment
% =	Modulus Assignment
- -	Decrement

6.3 BASIC ASSIGNMENT OPERATORS

=	**Assignment**
^ =	bitwise XOR and assign
& =	bitwise AND and assign
% =	take remainder and assign
- =	Subtract and assign
* =	Multiply and assign
/ =	bitwise OR and assign
>> =	Shift bits right with sign extension and assign
<< =	Shift bits left and assign
>>> =	unsigned bit shift right and assign

6.4 RELATIONAL OPERATORS

Relational operators are binary operators that require two operands to work with. The operands can be either constants or variables or expressions. The operators are as follows

>	**greater than**
> =	greater than or equal to
<	less than
< =	less than or equal to
!	boolean not
! =	not equal to

6.5 BOOLEAN LOGICAL OPERATORS

The boolean logical operators shown here operate only on boolean operands. All of the binary logical operators combine two boolean values to form a resultant boolean value.

Operator	**Result**
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short Circuit OR

&&	Short Circuit AND
!	Logical Unary Not
& =	AND ASSIGNMENT
| =	OR ASSIGNMENT
^ =	XOR Assignment
= =	Equal To
! =	Not Equal To
? :	Ternary if then else

6.6 TERNARY OPERATOR

Java includes a special ternary (three-way) operator that can replace certain types of, if then else, statements. This operator works in Java much like it does in C and C++. It has this general form

```
expression1 ? expression2 ? : expression3
class Ternary {
public static void main (String args[ ])
{
        int i, k;
        i = 10;
        k = i < 0 ? -i : i; // get absolute value of i
        System.out.println ("Absolute value of");
        System.out.println (i + "is" + u);
        i = -10;
        k = i < 0 ? -i : i;
        System.out.print ("Absolute value of");
        System.out.println (i + "is" + u);
}
}
        class Boollogic {
public static void main (String s[ ])
{
boolean a = true;
boolean b = false;
boolean c = a | b;
boolean d = a & b;
boolean e = a ^ b;
boolean f = (!a & b) | (a & !b);
boolean g = !a;
System.out.println (" a = " + a);
System.out.println (" b = " + b);
System.out.println (" a/b = " + c);
```

```
System.out.println (" a&b = " + d);

System.out.println (" a^b = " + e);

System.out.println (" !a&b|a&b = " + f);

System.out.println (" !a = " + g);
```

6.7 OPERATOR PRECEDENCE

Parentheses raise the precedence of the operations that are inside them. This is often necessary to obtain the desired result. For example, consider the following expression

```
a >> b + 3

Highest

( )             [ ]

+ +             - -     ~       !

*               /       %

+               –

>>              >>>     <<

>               > =     <       < =

= =             ! =

&

^

|

&&

| |

?:

=               op =

Lowest
```

Overriding Precedence

You can override the precedence by imposing brackets on the part which needs to be executed first.

```
a >> (b + 3)

(a >> b) + 3
```

Student Activity 1

1. What are arithmetic operators?
2. What is integer arithmetic?
3. What are relational operators?
4. List the relational operators alongwith their meanings.
5. What are logical operators? Make the truth table for each logical operator.

6.8 SUMMARY

- Java being the most powerful of all the available languages so far, comprises of the following operators for usage within the program:
  - Arithmetic Operators

- Assignment Operators
- Relational Operators
- Increment and Decrement Operators
- Bitwise Operators
- Logical Operators
- Ternary Operators

6.9 KEYWORDS

Argument: A value that is sent to a method when the method is called.

Assignment expression: Assigns a value to a variable.

Character: A value used in text. For example, the Petters A-the digits 0-9 (when not used as mathematical values), spaces, and even tabs and carriage returns are all characters.

Data type: The type of value represented by a constant, variable, or some other program object.

Expression: A line of program code that can be reduced to a value or that assigns a value.

Logical expression: An expression that results in a value of true or false.

Logical Operators: Operators like && (AND) and ||(OR) that enables you to create logical expressions that yield true or false results.

Operator precedence: Determines the order in which mathematical operations are performed.

6.10 REVIEW QUESTIONS

1. What are assignment operators?
2. What is operator precedence ? How can you override it?
3. What is the difference between ternary and boolean logical operators?
4. Write a program to calculate the average of three marks.
5. Write a program to check whether two input values are similar or not and then print the message accordingly.

6.11 FURTHER READINGS

E. Balaguruswamy, *Programming with Java*, Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naught on, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased.

Herbert Schildt, *Java: The Complete Reference, J2SE TM,* 2005, Mc Graw-Hill Professional.

UNIT 7

CONTROL STATEMENTS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe various types of selection statements available in Java
- Describe various iteration constructs available in Java

U N I T S T R U C T U R E

7.1 INTRODUCTION

A programming language uses control statements to cause the flow of execution to advance and branch based on changes in the state of a program. Java's program control statements can be put into the following categories : selection, iteration and jump. These have been discussed in this unit.

7.2 JAVA'S SELECTION STATEMENTS

Java supports two selection statements – if and switch. These statements allow you to control the flow of your program's execution based upon conditions known only during run time.

If

The if statement is Java's conditional branch statement. It can be used to route program execution through two different paths. Here is the general form

```
if (condition) statement1;
else statement2;
```

Here each statement may be a single statement or a compound statement enclosed in curly braces (that is, a block). The condition is any expression that returns a boolean value, the else clause is optional.

The if works like this : if the condition is true, then statement1 is executed, otherwise statement 2 is executed.

```
int a, b;
if (a<b) a = 0;
else b = 0;
```

Another Example for if-else if-else ladder is

```
int score = 65;
char grade;
if (score > = 90)
        grade = 'A';
else if (score > = 70)
        grade = 'C';
else if (score > = 60)
        grade = 'D';
else
        grade = 'F';
```

Student Activity 1

1. What is a selection statement? Which selection statements does Java provide?
2. What is the significance of a test-condition in an if statement?
3. Correct the following code fragment:

```
If (x = 1)
                k=100;
else
                k=10;
```

4. Write a program in Java to accept three integers and print the largest of the three. Make use of only if statement.
5. Write a program in Java to create the equivalent of a four-function calculator. The program requires the user to enter two numbers and an operator. It then carries out the specified arithmetical operation: addition, subtraction, multiplication or division of the two numbers. Finally, it displays the result.

7.3 SWITCH

The general form of a switch is

```
switch (expression)
{
case Value1 :
        Codesegment1
case Value2 :
        Codesegment2
. . .
case Value :
        Codesegment N
default :
        defaultCodeSegment
}
```

A switch statement is used for multiple way selection that will branch to different code segments based on the value of a variable or an expression. The optional default label is used to specify the code segment to be executed when the value of the variable or expression does not match with any of the case values. If there is no break statement as the last statement in the code segment for a certain case, the execution will continue on into the code segment for the next case clause without checking the case value.

An example of a switch statement

```
class switchTest
{
        public static void main (String args[ ])
        {
                int month = 4;
                String season;
                switch (month)
                {
                        case 12 :
                        case 1 :
                        case 2:
                        season = "Writer";
                        break;
                        case 3 :
                        case 4 :
                        case 5 :
                        season = "Spring";
                        break;
                        case 6 :
                        case 7 :
                        case 8 :
                        season = "Summer";
                        break;
                        case 9 :
                        case 10 :
                        case 11 :
                        season = "Autumn
                        break;
                        default :
                        season = "Bogus Month";
                }
        System.out.println ("April is in the" + season + ".");
                }
}
```

7.4 NESTED SWITCH

You can use a switch as part of the statement sequence of an outer switch. This is called a nested switch. Since a switch statement defines it own block, no conflicts arise between the case constants in the inner switch and those in the outer switch.

```
switch (count)
{
        case 1 :
        switch (target)
        {
                case 0 :
                System.out.println ("target is zero");
                break;
                case 1 :
                System.out.println ("target is one");
                break;
        }
        break;
        case 2 : // . . .
```

Here, the case 1 : statement in the inner switch does not conflict with the case 1 : statement in the outer switch. The count variable is only compared with the list of cases at the outer level. If count is 1, then target is compared with the inner list cases.

To summarize, there are three important features of the switch statement to note : the switch differs from the if, in that switch can only test for equality, whereas if can evaluate any type of boolean expression. That is, the switch looks only for a match between the value of the expression and one of its case constants.

No two case constants in the same *switch* can have identical values. A switch statement enclosed by an outer *switch* can have case constants in common.

A switch statement is usually more efficient than a set of nested ifs.

The last point is particularly interesting because it gives insight into how the Java compiler works. When it compiles a *switch* statement, the Java compiler will inspect case of the case constants and create a "jump table" that it will use for selecting the parts of execution depending on the value of the expression. Therefore, if you need to select among a large group of values, a switch statement will run much faster than the equivalent logic coded using a sequence of if-elses. The compiler can do this because it knows that the case constants are all of the same type and simply must be compared for equality with the switch expression.

Student Activity 2

1. One out of several different alternatives can be selected with the help of which statement?
2. Write one limitation and one advantage of a switch statement.
3. What is the significance of a break statement in a switch statement?
4. Write a program in java to input number of week's day (1-7) and translate it to its equivalent name of the week.

7.5 ITERATION CONSTRUCTS

Java's iteration statements are – for, while and do-while. These statements create what we commonly called Loops. As you probably know, a loop repeatedly executes the same set of instructions until a termination condition is met.

While loop

The while loop is Java's most fundamental looping statement. It repeats a statement or block while its controlling expression is true.

```
while (condition)
{
        // body of Loop
}
class whileTest
{
        public static void main (String s[ ])
{
        int n = 10;
        while (n > 0)
{
        System.out.println ("tick" + n)
n - -;
        }
}
}
```

do-while

As you just saw, if the conditional expression controlling a while loop is initially false, then the body of the loop will not be executed at all. However, sometimes it is desirable to execute the body of a while loop at least once, even if the conditional expression is false to begin with. Its general form is

```
do
{
        // body of Loop
}
while (condition);
```

Each iteration of the do-while loop first executes the body of the loop and then evaluates the conditional expression.

If this expression is true, the loop will replay. Otherwise the loop terminates,

```
class dowhileTest
{
        public static void main (String args[ ])
        {
                int n = 10;
                do
                {
                        System.out.println ("tick" + n);
                        n - -;
```

```
                }
                while (n > 0);
                        }
        }
```

Break

A Break statement exits a loop before an entry condition fails. In the example shown below an error message is printed and you break out of the for loop if j becomes negative.

```
Class uptec
{
        public static void main (String args[ ])
        {
                int i, j, k;
                j = i;
                k = 0;
                for (i = 1; i < = 64; i ++)
                {
                        j * 2 ;
                        if (j < = 0)
                        {
                                System.out.println    ("Error    :
                                overflow");
                                break;
                        }
                        k + = j;
                        System.out.print (k + "It");
                        if (i = = 0)
                        System.out.println ( );
                }
                System.out.println ("All done!");
        }
}
```

The most common use of break is in switch statement. Another general form of break statement is break label; where the label is optional. Without a label, the break statement will transfer the program control to the statement just after the innermost enclosing loop or switch statement. With a label, it will transfer the program control to the statement just after the enclosing statement or block of statements carrying the same label.

Example:

```
PrintWriter out = new PrintWriter (System.out);
Outerloop :
for (int i = 0, count = 0; i < dailyhigh.length; i++)
for (int j = 0; j < dailyhigh [i].length; j++)
        if ((dailyhigh [i] [j] > 70) & (++ count = = 3))
```

```
        {
out.println ("The data is : month : " + (i + 1) + , day =" + (j + 1));
            break  Outerloop;
        }
```

7.6 CONTINUE

The general form of a continue statement is :

```
continue  Label;
```

where the label is optional. Without a label, it behaves exactly the same as in C and C++. The program control is transferred to the point right after the last statement in the enclosing loop body. In while and do statement, the condition at expression1 will not be retested.

In for loop, the increment statements will be executed next. With a label the program, control will be transferred to the end of the enclosing loop body with the same label, instead of the innermost one.

For example, the following code segment defines a method to return the offset position of the first occurrence of one string str2 in the other string str1.

```
int  indexof  (String  Str1,  String  str2)
{
        int  len1  =  str1.length  (  );
        int  len2  =  str2.length(  );
        char  str2.firstChar  =  str2.  CharAt(0);
        advanceOneCharAtStr1:
        for  (int  i  =  0;  i  +  len2  <  =  len1;  i++)
        if  (str1.  CharAt  (i)  =  =  str2  FirstChar)
        {
                for  (int  j  =  1;  j  <  len2;  j++)
                if  (Str1.CharAt  (i+j)  !  =  str2.  CharAt  (j))
                continue  advanceOneCharAtStr1;
                return  i;
        }
        return  -  1;
        }
```

It is sometimes necessary to exit from the middle of a loop. Sometimes you will want to start over at the top of the loop. Sometimes you will want to leave the loop completely. For these purposes Java provides the break and continue statements. A continue statement returns to the beginning of the innermost enclosing loop without completing the rest of the statements in the body of the loop. If you are in a for loop, the counter is incremented.

```
for  (int  i  =  0;  i  <  m.length;  i++)
        {
                if  (m  (i)%2  =  =  0)  continue;
                //  process
        }
```

The continue statement is rarely used in practice because most of the instances where it is useful have simpler implementations. For instance, the above fragment could equally well have been written as :

```
for (int i = 0; i < m.length : i++)
{
        if (m[i] % 2 ! = 0)
        {
        }
}
```

7.7 RETURN

The general form of return statement is

```
return Expression;
```

A return statement is used to return control to the caller from within a method or constructor. If the method is defined to return a value, the expression must be evaluated to the return type of that method. Otherwise only an unlabeled return statement can be used.

Student Activity 3

1. What are iteration statements? Name the iteration statements provided in Java.
2. What is the difference between a while and do-while loop?
3. How many times is the loop body executed in a do loop?
4. How many times is the following loop executed?

```
int s =0, i= 0;
while (i++<5)
s+=I;
```

5. How many times is the following loop executed?

```
int s=0, i =0;
do
s+=i
while (i<5);
```

6. Write a program in Java to print largest even and largest odd number from a list of numbers entered through keyboard. The list terminates as soon as one enters 0(zero).

7.8 THE FOR STATEMENT

The for loop is the easiest to understand of the Java loops. All its loop-control elements are gathered in one place (on the top of the loop), while in the other loop construction of C++, they (top-control elements) are scattered about the program. The syntax of the for loop statement is for (initialization expression(s); test condition; update expression)

```
{
        loop-body
}
```

The following example program illustrates the use of for statement.

```
//program to print numbers from 1 to 10
class ForTest
{
        public static void main(String args[ ])
{
```

```
        int i;
        for (i=1, i<=10; i++)
System.out.prinln (i);
}
}
```

The following lines explain the working of the above given for loop.

1. Firstly, initialization expression is executed i.e. i=1 which gives the first value 1 to variable i.
2. Then, the test condition is evaluated i.e. i<=10 which results into true i.e. 1.
3. Since, the test-condition is true, the body-of-the loop i.e. System.out.prinln (i); is executed which prints the current value of i on the next line.
4. After executing the loop-body, the update expression i.e.++i is executed which increments the value of i.
5. After the update expression is executed, the test-condition is again evaluated. If it is true, the sequence is repeated from step number 3, otherwise the loop terminates.

Additional Features of For Loop

Java offers several features of for loop that increases the flexibility and applicability of for loop.

1. Multiple initialization and update expressions: A for loop may contain multiple initialization and /or multiple/update expressions. These multiple expressions must be separated by commas. For example ,

```
–
––
for (i=1, sum=0; i <=n, sum +=i, ++i)
system.println (i);
–
––
```

   The above code fragment contains two initialization expressions i=1 and sum =0 and two update expressions sum+ =i & ++i. These multiple expressions area executed in sequence i.e. during initialization firstly i=1 takes place followed by sum=0. Similarly, during updation, firstly sum +=i takes place followed by ++i

2. Optional expressions: In a for loop, initialization expressions, test expression and update expression are optional i.e. you can skip any or all of these expressions. Say, for example, you already have initialized the loop variable and you want to scrap off the initialization expression then you can write for loop as follows:

```
for (;test-condition; update expressions (s))
loop body
```

   See, even if you skip initialization expression of a for loop, but the ';' following it must be present.

3. Infinite loop: Although any loop statement can be used to create an infinite (endless loop) yet for is traditionally used for this purpose. An infinite for loop can be created by omitting the test-expression as shown below:

```
–
––
for (i=25; ;–i)
System.out.println ("An Infinite for loop")
```

4. Empty loop: If a loop does not contain any statement in its loop-body, it is said to be an empty loop. In such cases, a C++ loop contains an empty statement i.e., a null statement. For example,

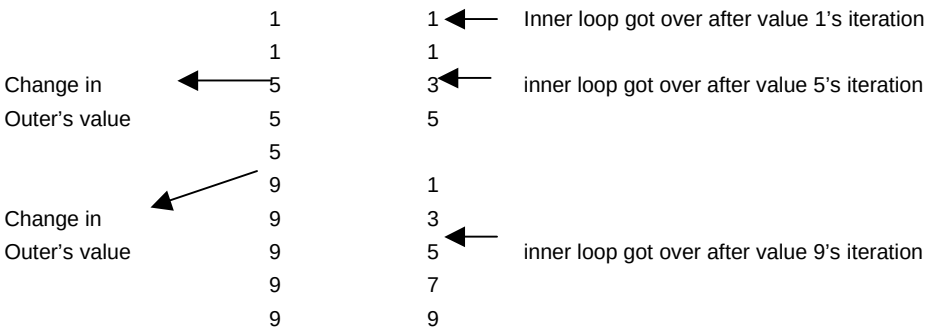
```
for (j=20; (j); –j);
is an empty loop.
```

Nesting of For Loops

A for loop may contain another loop in its body. This form of a loop is called nested loop. In a nested loop, the inner loop must terminate before the outer loop. The following is an example of a nested loop:

```
for (outer=1; outer<10; outer+=4)
{
        for (inner=1; inner<=outer; inner +=2)
        {
                System.out.print(outer);
                System.out.print(" ");
                System.out.println(inner);
        }
}
:
```

While working with nested loops, you need to understand one thing and that is, the value of outer loop variable will change only after the inner loop is completely finished (or interrupted). The above code will produce the output as:0



7.9 SUMMARY

- Java's selection statement, is if-else, is used to decide an action on various conditions.
- Java also has a switch statement to decide on various conditions but it is used mostly when there are more conditions.
- Iteration statements are statements that repeat the action until the specified condition is met, which are while and do-while.
- while checks the condition first and then the action is taken and do while checks the condition when the action at least once gets performed.
- Jump statements are break, continue and return.

7.10 KEYWORDS

Infinite loop: A loop that never ends.

Iteration statement: Statement that allows a set of instructions to be performed repeatedly.

Jump statement: Statement that unconditionally transfers program control within a function.

Nested loop: A loop that contains another loop inside its body.

7.11 REVIEW QUESTIONS

1. Fill in the blanks

   a. To exit a loop before it is completed, the _____ statement can be used.

   b. To check a condition which in turn checks a condition again is done through _____.

   c. To check the condition first and then execute the action can be done through _____ .

   d. To start with a certain point during the loop can be done through _____.

2. State whether the following are true or false

   a. The while loop is a post tested loop.

   b. Continue statement is used to skip the rest of the loop and start again from the beginning of the loop.

3. Write a Java program to convert metres to kilometres and display the output.

4. Write a Java program that will reverse a given number (i) passed to it as a command line argument.

5. Declare a constant in the program and display the reversed number.

6. Write a Java program that will display the Fibonacci Series 112358 . . .

7. Write a program that will display the command line arguments passed to it with the provisions that the (i) loop will break if one of the command line arguments is "Goodbye" (ii) the loop will continue from the beginning, skipping the rest of the loop if one of the arguments is "Welcome".

Answers to review questions

1. (a) break (b) nested loops (c) while (d)continue

2. (a) false (b) true

7.12 FURTHER READINGS

E. Balaguruswamy, *Programming with Java*, Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now*, Microsoft Press.

Naughton, Patrick, *The Java Hand Book*, OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2*, Tata McGraw-Hill.

UNIT

8

CLASSES : AN INTRODUCTION

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Define a class
- Give the general form of a class
- Declare a simple class
- Define methods
- Add methods to a class
- Define constructors
- Describe the usage of 'this' keyword
- Describe garbage collection
- Describe Finalize() method

U N I T S T R U C T U R E

8.1 INTRODUCTION

At the end of this lesson you should be able to define a class and create its objects using new operator. Also you should be able to write instance methods with / without parameter and with / without return value. You will also learn the special method called constructor. You should understand the meaning of 'this' keyword.

8.2 WHAT IS A CLASS

A class is a user defined data type with a template that serves to define its properties and operations. Once the class has been defined, we can create 'Variables' of that type. These variables are termed as instances of classes, which are the actual objects.

A General form of Class

The general form of a class definition is

```
class classname
{
```

```
                [variable declarations;]

                [methods declarations;]

        }
```

for e.g. ;

- The data, or variables, defined within a class are called instance variables. The code is contained within methods. These are also called members of the class.
- In order to describe the characteristics of all pens, you might define a pen class and specify operations such as write and refill and attributes such as ink colour and ink remaining. When you create individual pen objects to represent my blue pen, teacher's red pen, you may initialize ink colour and ink amount.

Class as Datatype

Java has four kinds of named data types and two categories of variables. The data types are primitives, classes, interfaces and arrays. Variables can either contain primitive values or refer to objects.

A class creates a new data type that can be used to create objects. A class creates a logical framework that defines the relationship between its members.

When you declare an object of a class, you are creating an instance of that class. Thus, a class is a logical construct. An object has physical reality.

Simple Class

```
class Movie
{
        String title;
        int length;
        int cost;
        char type;
void getdata (int x, int y, int z)
{
        length = x;
        cost = y;
        type = z;
}
Movie (String s)
{
        title = s;
}
}
```

Class declaration only creates a template. It does not create an actual object, and hence does not occupy memory. To actually create movie object you will use a statement:

```
Movie mov1 = new Movie ("Mr. India");
Movie mov2 = new Movie( );
```

Declaring Objects

Class is a static definition that enables us to understand all the objects of that class. Objects exist

at run time. They hold attributes and they send messages to each other Classes, however, do not exist at run time.

Each instance of the class (that is, each object of the class) contains its own copy of the variables.

In case of the movie class, each individual movie is an instance of movie. "Gone with the wind" is one distinct instance of movie while "Last action her" is another. Each has its own set of variables. For e.g. mov1 and mov2 objects have four instance variables each – mov1.title, mov1.length, mov2.title, mov2.length.

New Operator

Each new object is identified within Java by a unique object reference.

objectref = new Classname( );

For e.g. to create two movie objects

- Movie Mov1 = new Movie ("Gone with the wind");
- Movie mov2 = new Movie ("Last action Hero");

The above statements create new instance variables of Movie type named mov1 and mov2. New operator assigns object's reference to mov1 and mov2 which points to the location of that object in memory. New operator allocates memory for the new objects and also calls a special initialization method in the class called constructor. It returns a reference to the new object. New operator knows how much memory is required by looking at the class definition. It calls the constructor to initialize the instance variables in the new object. However, declaring a new object variable receives a chunk of memory only large enough to hold a reference to an object.

Class definition typically includes:

- Access Modifier
- Class Keyword
- Instance Fields
- Constructors
- Instance Methods
- Class Fields
- Class Methods

A class is an encapsulated collection of data and methods to operate on data.

Access Modifier	:	Specifies the availability of the class from other classes. The access modifiers are private, public and protected.
Class Keyword	:	Tells Java that the following code block defines a class.
Instance Fields	:	Variable and constants that are used by objects of the class.
Constructors	:	Methods having the same name as class, used to control the initial state of any class object.
Instance Methods	:	Define the functions that can act upon objects in this class.
Class Fields	:	Variables and constants that belong to the class and are shared by all objects of that class.
Class Methods	:	Methods used to control the values of class fields.

```
public class employee
{
```

```
        // Instance Variables
        String custName;
        String Address;
        float Amt;
        // Class Variable
        static int ctr = 0;
        // Class Methods
        public static void counter ( )
        {
ctr = ctr + 1;
}
// constructor
employee( )
{
        custname = " ";
        Address = " ";
        Amt = 0.0;
employee (String a, String b, float f)
{
        custname = a;
        Address = b;
        Amt = f;
}
// Instance Method
void getdata (String a, String b, float f)
{
        custname = a;
        Address = b;
        Amt = f;
}
}
```

Object Reference Variable

Object reference variables are declared in Java with a statement that gives the type of object to which the variable is expected to refer.

e.g.:

```
        Object anyRef;
        String myString;
```

The type of references these variables can hold depend on the object hierarchy. Because every class in Java descends from Object, anyRef could refer to any object. However, because String cannot be subclassed, myString can refer to a string object only. The content of newly created reference variable depends on where it is declared. For instance, in variables and class variables, the content is null. Variables that are declared inside the blocks are uninitialized.

The instanceof operator returns true for any type in the operand's parentage. For instance consider the following hierarchy

```
java.lang.object
|-java.awt.component
        |-java.awt.container
                |-java.awt.Panel
                |-java.awt.ScrollPane
                |-java.awt.Window
                        |-java.awt.Dialog
                                |-java.awt.FileDialog
Panel testObj = new Panel( );
```

The instanceof operator returns true for any type in the operand's parentage.

testObj instanceof java.awt.Panel

testObj instanceof java.aw.component

This feature of Java can be used for manipulating collection of objects. For e.g. Vector and Stack class can accept and return any references of an Object reference.

Students Activity 1

1. What is a class?
2. Give the general form of a class?
3. Why and how are objects of a class declared?
4. What are objects?
5. What is the role of access modifier?
6. How are object reference variables declared in Java?

8.3 WHAT ARE METHODS

A Java method is equivalent to a function, procedure, or subroutine in other languages except that it must be defined inside a class definition. Instance methods form the foundation of encapsulation and are a key to providing a consistent interface to the class.

Adding Methods to Classes

Methods are declared inside the body of the class but immediately after the declaration of instance variables. The general form of a method declaration is

```
Returntype methodName (parameter list)
        {
        Method-body;
        }
```

A Returntype can be a primitive type such as int, a class type such as string or void.

A MethodName begins with a lowercase letter and compound words in the method name should begin with uppercase letter according to Java convention.

An optional parameter list/argument list must be inside parentheses, separated by commas.

The method-body must be enclosed in braces.

Return Value

The return type specifies the type of value the method would return. This could be a simple data type such as int as well as any class type or it could even be void type if the method does not return any value.

```
e.g.: class Box
        {
                double width;
                double height;
                double depth;
                double volume( )
                {
                        return width * height * depth;
                }
        class ABC
        {
                public static void main (String args[ ])
                {
                        Box mybox1 = new Box( );
                        double vol;
                        mybox1.width = 10;
                        mybox1.height = 20;
                        mybox1.depth = 15;
                        vol = mybox1.volume( );
                        System.out.println ("Volume is" + vol);
                }
        }
```

The type of the data returned by a method must be compatible with the return type specified by the method.

The variable receiving the value returned by a method must also be compatible with the return type specified for the method.

Method Accepting Parameters

The parameter list is always enclosed in parentheses. This list contains variable names and types of all the values we want to give to the method as input. If the method does not take any argument, simply leave the parentheses empty.

```
class Rectangle
        {
                int length; int width;
                void getdata (int x, int y)
                        {
                                length = x;
                                width = y;
```

```
                }
        int rectArea( )
                {
                        int area = length * width;
                        return (area);
                }
}
```

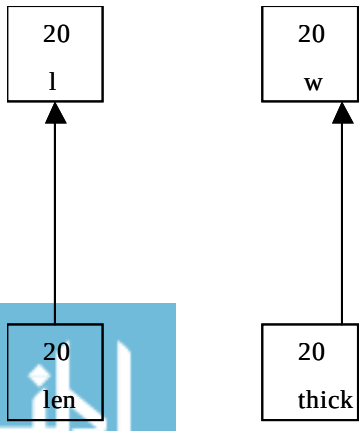
In the above example, length and width are instance variables.

x and y are parameters.

void indicates method, does not return a value.

When a primitive is passed to a method, a copy of the value is generated. If the method changes the value of the argument in any way, only local argument will be affected. When the method terminates, this local argument is discarded and the original variable in the calling method is unchanged.

```
class Rectangle
{
        int length;
        int width;
        void change (int l, int w)
        {
                l = l + 10;
                w = w + 10;
                System.out.println ("l =" + l + "w = " +w);
        }
        public static void main (String args[ ])
        {
                int len = 20;
                int thick = 20;
                Rectangle rect = new Rectangle( );
                rect.change (len, thick);
                System.out.println ("len" + len + "thickness" + thick);
        }
}
```



Accessing Class members

```
class Rectangle
{
        int length, width;
        void getData (int x, int y)
        {
                length = x;
                width = y;
        }
        int rectarea ( )
        {
                int area = length * width;
                return (area);
                }
        }
class RectArea
{
        public static void main (String args[ ])
        {
                int area1, area2;
                Rectangle rect1 = new Rectangle ( );
                Rectangle rect2 = new Rectangle( );
                rect1.length = 15 // Accessing variables
                rect1. width = 10
                area1 = rect1.length * rect1.width
                rect2.getData (20, 12);
                area2 = rect2.rectArea( );
                System.out.println ("Area1 = "+ area1);
                System.out.println ("Area2 = "+ area2);
        }
}
```

Use the dot operator with instance variable and to call instance methods. The general syntax is

```
Objref.instance variable;
Objref.MethodName (arguments);
```

Student Activity 2

1. What are methods?
2. How will you add methods to a class?
3. What is the function of return value?
4. What happens when a primitive is passed to a method?

Constructor

When an object is created, Java performs default initialization, char variables are set to '\u0000', byte, short, int, long to zero, boolean to false, float to 0.0

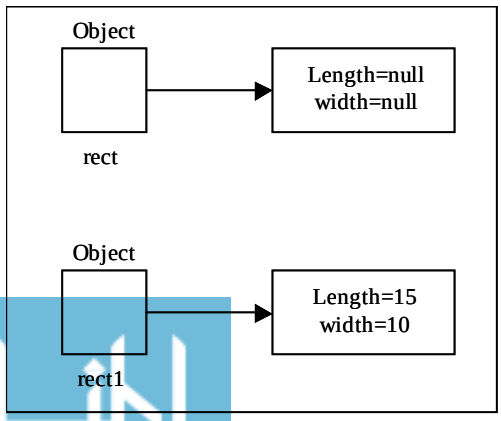
All objects that are created must be given initial values. We can do it,

1. by assigning the value to instance variable using dot operator.
2. through instance method for e.g. getdata( )
3. by using constructor method to initialize an object when it is first created.
   - Constructors have the same name as class itself.
   - Constructors do not specify a return type.
   - Constructor is called automatically by the run time system.

For e.g.

```
class Rectangle
        {
                int length, width;
                Rectangle (int x, int y) // constructor
                {
                        length = x;
                        width = y;
                }
                int rectArea( )

                {
                        return (length * width);
                }
        }
class ABC
{
        public static void main (String args[ ])
        {       Rectangle rect = new Rectangle( )
        Rectangle rect1 = new Rectangle (15, 10) // calling constructor
                int area1 = rect1. rectArea ( );
                System.out.println ("Area1 = "+ area1);
        }
}
```



If you do not provide any constructors, a default no-arg constructor is provided for you. This constructor takes no arguments and does nothing. If you want a specific no argument constructor as well as constructors that take arguments, you must explicitly provide your own no argument constructor.

"this" Keyword

All instance methods receive an implicit argument called "this", which may be used inside any method to refer to the current object i.e., the object on which method was called. Inside an instance method, any unqualified reference is implicitly associated with the this reference.

```
public void setRating (String st)
        {
                rating = st;
                this.rating = st;
        }
```

Usage of this keyword:

- When you need to pass a reference to the current object as an argument to another method.
- When the name of the instance variable and parameter is the same, parameter hides the instance variable.

```
class Rectangle
{
        int length, width;
        Rectangle (int length, int w);
        {
                length = length;            // incorrect
                width = w;           // correct
                this.length = length;       // incorrect
        }
}
```

Garbage Collection

Garbage collection is the process that handles memory allocation. It is incharge of cleaning the memory space allocated to the objects that are not in use.

```
Movie mov1 = null;
Rectangle Rect1 = null;
```

This declares mov1, rect1 with a special reference called null. This indicates that the reference does not refer to a real object yet.

Discarding an Object :

- When you have finished using an object, you can set its reference to null. This indicates that the variable no longer refers to the object and object will be marked for garbage collection. The Java Virtual Machine automatically decrements the number of active references to an object whenever an object is de-referenced, goes out of scope.
- All Java objects automatically grab the memory that they need when they are created, and when the object is no longer needed, the Java garbage collection process reclaims the memory.

finalize ( ) Method

JVM is supposed to run an object's finalize method after it has decided that an object has no references and thus, should be recovered, but before actually reclaiming the memory.

Java manages memory automatically– so an object does not need to explicitly free up any secondary memory it might have allocated. To allow an object to clean up resources other than memory, such as open files, Java allows a class to provide a finalize( ) method. Unfortunately, there is no guarantee as to when this will happen. It is implementation specific.

If an object holds another resource such as a file, the object should reclaim it. Finalize method is called automatically before garbage collection.

Student Activity 3

1. What are constructors?
2. How can you give an initial value to an object?
3. Describe the usage of “this” keyword.
4. What do you mean by garbage collection?
5. How can you discard an object?
6. Describe the usage of finalize() method.

8.4 SUMMARY

- An object is an abstraction of a real world object and is a run time entity.
- A class is a template for objects.
- An object is an instance of a particular class.
- Object can be created by using new operator.
- Manipulate an object by using its instance methods.
- Instance method receives a "this" reference to the current object.
- Most classes provide one or more constructors to initialize instance variables.
- When all references to an object are lost it is marked for garbage collection. Garbage collection reclaims memory used by the object. Garbage collection is automatic.
- The finalize method is called just before garbage collection.

8.5 KEYWORDS

Classes: A collection of variables and methods that an object can have, or a template for building objects.

Objects: An instantiation of a class.

Methods: A routine that belongs to a class.

Fields: A data object encapsulated in a class.

Instance: A concrete representation of a class or object. A class can have many instances.

Inheritance: A property of object-oriented languages where a class inherits the methods and variables of more general classes.

8.6 REVIEW QUESTIONS

1. Define each of these terms with an example:

   a. Class b. Instance Variable c. Instance Method

2. Identify instance variables, class variables, method variables, classname in the following example

```
class ABC
{
        static int z;
        int x;
        int y;
        ABC( )
        {
                x = 0;
                y = 0;
        }
        void initz ( )
        {
                z = 10;
        }
        int calc (int a, int b)
        {
                int sum;

                sum = a + b;
                return sum;
        }
}
```

3. Find out errors.

```
        Class house
        {
                int width;
                int length;
                int height;
        }
```

4. What is a constructor ?

5. What is the use of "this" keyword ?

6. Create a class called student with the following characteristics and behaviour. The student is given a unique serial number starting with '1'. The student has name, address. He appears in the exams for three different subjects. The class should be able to provide the grade of the student on the basis of the following criterion.

Total Marks	Grade
80-100	"Excellent"
65-80	"Good"
50-65	"Pass"
< 50	"Fail"

Use constructor to input Roll No and name, and one instance method to store marks.

8.7 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

UNIT

9

METHOD AND CLASSES IN DETAIL

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Define method overloading
- Describe constructor overloading
- Describe a default constructor
- Define objects as parameters
- Describe recursion
- Describe access control
- Describe static and final variables and methods
- Describe nesting of classes
- Describe string class
- Describe command line arguments.

U N I T S T R U C T U R E

9.1 INTRODUCTION

In the previous unit we talked about classes, objects, instance methods and constructors. This unit focuses primarily on method overloading and constructors. After going through this unit you will be able to write methods with primitive parameters in addition to passing objects as

parameters and returning primitive objects to the calling method. You will also be able to explore the different types of access modifiers and usage of command line arguments.

9.2 METHOD OVERLOADING

In Java, it is possible to create methods that have the same name, provided they have different signatures. A signature is formed from its name, together with the number and types of its arguments. The method in return type is not considered a part of the method signature. Method overloading is used when objects are required to perform a similar task but by using different input parameters. When we call a method in an object, Java matches up the method name first, and then the number and type of parameters, to decide which method to execute. This process is known as polymorphism.

```
public class Movie
{       float price;
        public void setprice( )
        {
                price = 3.50;
        }
        public void setprice (float newprice)
        {
                price = newprice;
        }
}
```

```
Movie mov1 = new Movie( );
    mov1.setprice( );
          mov1.setprice (3.25);
```

As you can see setprice( ) is overloaded two times. The first version takes no argument and sets the price to 3.50 and second constructor takes single argument which sets the price to 3.25.

When an overloaded method is called, Java looks for a match between the argument used to call the method and method's parameters. However, this match need not be exact. In some cases Java's automatic type conversion takes the important role in overloaded resolution.

Example :

```
public class Movie
{
        float price;
        public void disp( )
        {
                System.out.println ("No parameter");
        }
        public void disp (int x, int y)
        {
                System.out.println ("x =" + x + "y =" + y);
        }
        public void disp (double x)
        {
```

```
                System.out.println ("x =" + x);
        }
}
class overload
{
        public static void main (String args[ ])
        {
                Overload ob = new overload( );
                int i = 100;
                ob.disp( ) // invokes no argument constructor
                ob.disp (10, 20) // invokes disp (int, int)
                ob.disp (i) // invokes disp (double)
                ob.disp (100.50) // invokes disp (double)
        }
}
```

Class Movie does not define disp (int). When disp (int) is called from main( ), when no matching method is found, Java automatically converts an integer into a double. Java uses its automatic type conversion only if no exact match is found.

Java's standard class library has abs( ) method. This method returns absolute value of all type of numeric types e.g. int, double, float. This method is overloaded in Math class. Whereas in 'C' language there are three methods – abs( ), fabs( ), labs( ) – to return absolute of int, float and long values respectively, the java.awt.Graphics class has six different drawImage methods.

As far as Java is concerned, these are entirely different methods. The duplicate names are really just a convenience for the programmer.

9.3 CONSTRUCTOR OVERLOADING

Constructors are chunks of code used to initialize a new object. As with overloaded methods, if you want to provide more than one constructor in a class, each one must have a different signature. Constructors always have the name of the class and no return type.

Example :

```
public class Movie
{
        private String title;
        private String rating = "PG";
        public Movie( )
        {
                title = "Last Action--";
}
public Movie (String Ntitle)
{
        title = Ntitle;
}
}
```

```
Calls no argument Constructor

Calls Single argument Constructor
```

```
Movie mov1 = new Movie( );

Movie mov2 = new Movie("Gone with the Wind");
```

Class can have any number of constructors as long as they have different signatures.

Default Constructor

If no constructors are declared in a class, the compiler will create a default constructor that takes no parameter.

When you create new Movie object, the compiler decides which constructor to call, based on the argument specified in parentheses in the new statement. Constructor can call another constructor of the same class by using the this( ) syntax. By using this( ), you avoid duplicate code in multiple constructors. This technique is used when Initialization routine is complex.

```
public class Movie
        {
                private String title;
                private String rating;
                public Movie( )
                {
                        this ("G");
                }
                public Movie (String newRating)
                {
        rating = newRating;
                }
        }
```

Rules : The call to this( ) must be the first statement in the constructor. The argument to this( ) must match those of the target constructor.

Student Activity 1

1. What do you mean by method overloading?
2. Give an example to describe polymorphism.
3. What do you mean by constructor overloading?
4. What is a default constructor?

9.4 OBJECTS AS PARAMETERS

When a primitive value is passed into a method, a copy of its value is passed into the method argument. If the method changes the value of the argument in any way, only the local argument is affected. When the method terminates, this local argument is discarded.

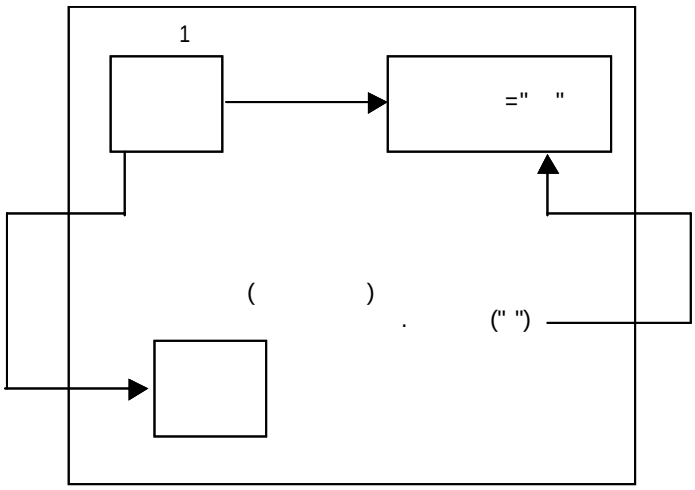
```
int num=150;                                    num
obj.nmethod (num);                              150
System.out.printIn ("num"+num);

public void amethod (int a)                     A=150
{
If (a<0 || a>100)                               Output
a=0;                                            arg: 0
System.out.printIn ("arg:"+a)                   num: 150
}
```

This way of method calling is called call-by-value.

When an object reference is passed to a method the argument refers to the original object :

```
Movie mov1 = new movie ("Gone with the Wind");
        mov1.setRating ("PG");
        anobj.amethod (mov1);
                public void aMethod (Movie ref)
                {
                        ref.setRating ("R");
                }
```



When mov1 is passed into aMethod( ), the method receives a reference to the original Movie object. When aMethod( ) terminates, the original Movie object referred to by mov1 has a rating of 'R' not 'PG'. Inside the subroutine, 'ref' is used to access actual argument specified in call. This means that the changes made to the parameter will affect the argument used to call the subroutine. This way of method calling is known as call-by-reference.

9.5 RETURNING OBJECTS

A method can return primitive datatypes or an object.

Example :

```
class test
{
        int a;
        test (int i)
        {
                a = i;
        }
        test incrbyten( )
        {
                test temp = new test (a+10);
                return temp;
        }
```

```
}
class Retob
{
        public static void main (String args[ ])
        {
                test ob1 = new test (2);
                test ob2;
                ob2 = ob1.incrbyten( );
                System.out.println ("ob1.a:" + ob1.a);
                System.out.prinln ("ob2.a:" + ob2.a);
        }
}
Output :            ob1.a:2
                    ob2.a:12
```

Each time when incrbyten( ) is invoked, a new object is created, and a reference to it is returned to the calling routine.

One of the most important uses of object parameters involves the constructor. You may want to construct a new object so that it is initially the same as the existing object.

```
class rectangle
{
        int width;
        int height;
        rectangle (int w, int h)
        {
                width = w;
                height = h;
        }
        rectangle (rectangle r)
        {
                width = r.width;
                height = r.height;
        }
}
class ABC
{
        public static void main (String args[ ])
{
                Rectangle rect1 = new Rectangle (10, 15);
                Rectangle rect2 = new Rectangle (rect1);
        }
}
```

9.6 RECURSION

Recursion is a process by which a function calls itself repeatedly, until some specific condition has been satisfied. The process is used for repetitive computations in which each action is stated in terms of the previous result.

For e.g. to calculate the factorial of a positive integer quantity, the following expression would be given:

n! = 1 × 2 × 3 - - - - xn where n is the specified integer. It can also be written as

n! = n × (n-1) !

This is a recursive statement of the problem in which the desired action is written in terms of the previous result.

Now 1! = 1 by definition. This provides a stopping condition for recursion.

When a recursive program is executed, the recursive function calls are not executed immediately. Rather, they are placed on a stack until the condition that terminates the recursion is encountered.

The function calls are then executed in reverse order as they are "popped" off the stack.

```
class f
{
        int num;
        int fact (int n)
        {
                if (n = = 1)
                return (1);
                num = n * fact (n-1);
                return num;
        }
}
class ABC
{
        public static void main ( )
        {
        int Result;
        f f1 = new f( );
        Result = f1.fact (5);
        System.out.println ("Factorial" + Result);
        }
}
```

When fact is called with an argument of 1, the function returns 1; otherwise it returns product of fact (n-1) ´ n. To evaluate this expression, fact( ) is called with n-1. This process repeats until n equals 1 and the calls to the method begin returning. When a method calls itself, new local variables and parameters are allocated storage on the stack and the method code is executed with these new variables from the start. A recursive call does not create a new copy of the method, only the arguments are new. As each recursive call returns, the local variables and parameters are removed from the stack and the execution resumes at the point of the call inside the method.

The main advantage of recursive methods is that they can be used to create clearer and simpler versions of algorithms than can their iterative algorithms. Recursive programs may execute little slowly because of added overhead of the additional function calls. Recursive programs may cause Stack overrun. You must have an if statement to force the method to return to calling starting function.

9.7 ACCESS CONTROL / VISIBILITY

How a member can be accessed is determined by access specifier that modifies its declaration. We have seen that the method and variables of a class are visible everywhere in the program. However, it may be necessary in some situations to restrict the access to certain variables and methods from outside the class, e.g. we may not like the objects of a class directly as they alter the value of a variable or access a method. We can achieve this in Java by applying visibility / access modifiers. Java provides three types of visibility modifiers:

- Public
- Private
- Protected (only when inheritance is involved)

Public Access

- Variable or method is visible to the entire class in which it is defined. If we want to make it visible to all the classes outside this class then the variable or method has to be declared as public.

```
public int number;

public void calc( ) { - - - - - }
```

The main( ) is always preceded by public specifier, because it is called from outside i.e. by the JVM.

- By default access specifier is 'friendly'. The difference between the 'public' access and the 'friendly' access is that the public modifier makes fields visible in all classes, regardless of their packages while the friendly access makes fields accessible only to classes in the same package, but not in other packages (a package is a group of related classes stored separately).

Protected Access

- Protected modifier makes the field available to all the classes and subclasses in all the packages. Note that nonsubclasses in other packages cannot access the protected member.

Private Access

- Private access provides the highest degree of protection that is accessible only within its own class, but not in the subclasses.

Private Protected Access

- A field can be made visible in all subclasses regardless of what package they are in. These fields are not accessible by other classes in the same package.

  Example :

```
class access
{
        int a; // default access
        public int b; // public access
        private int c; // private access
```

```
        void setc (int i)
        {
                c = i;
        }
        int getc ( )
        {
                return (c);
        }
}
class test
{
        public static void main (String args[ ])
        {
                test ob = new test( );
                ob.a = 10 // OK
                ob.b = 20 // OK
                ob.c = 30 // Error
                ob.setc (30); // OK
        }
}
```

Student Activity 2

1. Describe the method of call-by-value.
2. Describe the method of call-by-reference.
3. What is recursion?
4. What is the advantage of recursive methods?
5. Describe various types of visibility modifiers available in Java.

9.8 UNDERSTANDING STATIC, FINAL

Static Variables and Methods

Instance variables are created every time a new copy of the class is created. They are accessed using the objects (with dot operator). If we want to define a member that is common to all the objects and accessed without using a particular object, we use static keyword. Member declared as static belongs to the class as a whole rather than objects created from the class. Instance variables declared as static are, essentially, global variables. When objects of its class are declared, no copy of a static variable is made. Instead, all instances of the class share the same static variable.

Static variables are used when we want to have a variable, common to all instances of a class. For e.g. to have a variable that could keep a count of how many objects of a class have been created, Java creates only one copy of a static variable. Like static variables, static methods can be called by using

```
Classname.method name( );
```

e.g. :

```
class Rectangle
```

```
        {
                static Area (int x, int y)
                {
                        return x * y;
                }
        }
        class ABC
        {
        public static void main (String args[ ]) )
        {
                int area = Rectangle. Area (10, 20);
                System.out.println ("Area" + area);
        }
}
```

Static methods can only call other static methods. They can only access static data. They cannot refer to "this" pointer and super keyword because method does not operate on object.

Class variables belong to a class and are declared as static in class definition.

Example :

```
public class Movie
{
        static double minprice;
        static int count = 0;
        private string title, rating;
        Movie ( )
        {
                minprice = 100.00;
                count = count + 1;
        }
        public void getdata (String t, String r)
        {
                title = t;
                rating = r;
        }
public static void main (String args [ ])
{
        Movie mov1 = new Movie( );
        Movie mov2 = new Movie( );
        Movie mov3 = new Movie( );
        mov1.getdata ("Gone With the Wind", "PG");
}
}
```

Class variables are initialized when the class is loaded. The default values for class variables are; numbers are set to 0, boolean variables to false, characters are set to '\u40000', and references set to null class variables can be initialized with non default values at the time of Initialization. Complex initialization of class variables is performed in a static initialization block. For e.g., if you need to do computation in order to initialize your static variables, you can declare a static block which gets executed only once, when the class is first loaded.

Example :

```
class Staticvar
{
        static int a = 3;
        static int b;
        static void calc (int x)
        {
                System.out.println ("x =" + x); // x = 40
                System.out.println ("a =" + a); // a = 3
                System.out.println ("b =" + b); // b = 12
        }
        static
        {
                System.out.println ("Static Block");
                b = a * 4;
        }
        public static void main (String args[ ])
        {
                calc (40);
        }
}
```

Final Variables

A final variable is a named constant which cannot be modified. This means that the final variable must be initialized. When it is declared, variables declared as final do not occupy memory on a per instance basis.

```
final float PIE = 2.73;
```

Final Classes

You may also declare a class to be final. A final class is one that cannot be inherited from. In fact by declaring a class as final you are making a strong design decision that the class is complete enough to meet all its requirement, it means you are preventing the class from being inherited. Declaring a class as final implicitly declares all its methods as final too.

```
final class A
{
        void show( )
        {
                System.out.println ("Class A");
        }
```

```
}
class B extends A // ERROR ! cannot subclass A
{
        void disp( )
        {
                System.out.println ("Class B");
        }
}
```

Final Method

A final method is one that cannot be overridden in a subclass. If a programmer inherits from the class, he or she is not allowed to provide an alternative version of this method. This is a useful technique to prevent programmers from maliciously redefining core methods. To disallow a method from being overridden, specify final as a modifier at the start of its declaration.

```
class A
{
        final void show( )
        {
                System.out.println ("This is a final Method");
        }
}
class B extends A
{
        void show( )
{
                // cannot override, compiler gives error
                System.out.println ("ERROR");
        }
}
```

9.9 NESTED AND INNER CLASSES

Inner classes are classes which are defined within other classes. These inner classes have the same scope and access as other variables and methods defined within the same class i.e., the scope of a nested class is bound by the scope of its enclosing class.

```
public class PrivateAirport extends Airport
{
        String owner;
        private Airport (String str){owner = str;}
        class PrivateFlight extends Flight
        {
                string flightowner;
                private Flight( ) {flightowner = owner;}
        }
```

```
}
Flight getFlight( )
{
        return new PrivateFlight( );
}
        }
class Airport {                          Class Flight {
        String Airportname;              int altitude;
               :                         int speed;
               :                                :
               }                                 :
}
```

There are four types of inner classes:

- Static Inner Class
- Member Inner Class
- Local Inner Class
- Anonymous Inner Class

Static inner classes only have access to the static component of the outer class with no access to instance components directly. The static inner class can access instance components through its object.

Member inner classes are non static classes. These can access all the components of the outer class. An instance of the outer class needs to exist to be able to create a new instance of the member inner class.

```
public class outer
{
              public class inner
              {
                     public void innerMethod( )
                     {
                     // 'this' refers to an instance of
                     // Outer.inner
                     // 'Outer.this' refer to an instance of outer
                     // 'super' super class of inner (object)
                     // 'outer.super' superclass of outer
                     }
              }
}
```

Member inner classes are compiled to separate class files. In this case, the outer/inner.class is created.

Local inner classes are declared within the code block of a method. They have access to final variables only. Local inner classes cannot be declared public, protected, private or static and cannot have static members.

Anonymous inner classes are local inner classes with no class name. They are commonly used to implement user interface adapter to perform event handling. These classes can access local variables in the method and parameters passed to the method only if they are defined final.

Student Activity 3

1. What are static variables? Describe their usage.
2. What is a static method?
3. What is a final variable?
4. What is a final class?
5. What is a final method?
6. What are inner classes? Name its various types.

9.10 THE STRING CLASS

A string is a sequence of characters. Java provides String class to manipulate strings, passed as arguments to methods.

```
System.out.println ("Hello world");

String str = "Action";

String str = new String ("Anil");
```

Java strings are more reliable than 'C' strings because there is lack of bound-checking in C. A Java string is not a character array and is not Null terminated.

To get the length of string, use length method of the string class:

```
int m = str.length( );
```

Strings can be concatenated with '+' operator:

```
String FullName = name1 + name2;

System.out.println (Firstname + "Kumar")
```

String Buffer Class

String creates strings of fixed length while StringBuffer creates strings of flexible length that can be modified in terms of both length and content. We can insert characters and substrings in the middle of a string or append another string to the end.

String and StringBuffer classes are discussed in a later chapter in detail.

9.11 COMMAND LINE ARGUMENTS

Command line arguments are parameters that are supplied to the application program at the time of invoking it for execution. We can write Java programs that can receive and use the arguments provided in the command line :

public void static main (String args[ ]) where args is a string object which contains array of strings. Any argument passed through command line are passed to array args [ ].

Example:

```
Java Test One Two Three

        One          ®      args [0]

        Two          ®      args [1]

        Three        ®      args [2]

Individual elements can be accessed using an index.
```

```
class Comparetext
{
        public static void main (String args[ ])
        {
                int count, i = 0;
                String string;
                count = args.length;
        System.out.println ("No of arguments = "+ count);
                while (i < count)
                {
                        string = args [i];
                        i = i+1;
        System.out.println (i + "th argument is "+string);
                }
        }
}
```

The above program will display all the parameters given from the command line.

Student Activity 4

1. What is a string?
2. Describe the String class available in Java.
3. What are command line arguments?

9.12 SUMMARY

- Several methods in class can have the same name with different signatures.
- Most classes provide overloaded constructors to Initialize new objects in different ways.
- When a primitive is passed to a method, a copy of the value is generated.
- When an object reference is passed to a method, the argument refers to the original object.
- A method can return any type of data including class types.
- A method which calls itself is said to be recursive.
- A variable / method declared as public has the widest possible visibility and is accessible everywhere.
- Private variables are accessible only within their own class.
- Every string you create is actually a string object. Object of type string is immutable.
- StringBuffer class allows strings to be altered.
- Command line arguments are received by Java program in object of String class, which contains an array of Strings.

9.13 KEYWORDS

Default constructor : Constructor created by the compiler with no parameters, created if no constructor is declared in the class.

Recursion : A process by which a function calls itself repeatedly, until some specific condition has been satisfied.

Static variables : Used when when we want to have a variable common to all instances of a class.

Final variable : A named constant which cannot be modified.

Final method : A method that cannot be overridden in a subclass.

Inner Class : A class defined within other class.

Command line Arguments : Parameters that are supplied to the application program at the time of invoking it for execution.

9.14 REVIEW QUESTIONS

1. Answer the following questions.

1.1 Overloaded methods can have any return type.

a. True

b. False

1.2 The signature of a method consists of

a. type of parameters

b. number of parameters

c. name of the method

d. All of the above

1.3 If a class Aquarium has following two methods

a. public int addfish (Fish f)

b. protected boolean addfish (Fish f[ ])

Which of the following statements is correct ?

a. This technique is called overloading.

b. Compiler will reject the method because return type is different.

c. You cannot have two methods with the same name.

1.4 We can overload methods with differences only in their return type.

a. True

b. False

1.5 State whether the following are true or false:

a. Class must have a constructor written into it.

b. A class can have any number of constructors.

c. The constructor should have a void return type.

d. You can call a constructor from other constructors in the same class.

e. Java always provides a default constructor to a class.

1.6 Which keyword can protect a class in a package from accessibility by the classes outside the package ?

a. Private

b. Protected

c. Final

d. Do not use any keyword at all.

1.7 To make a member of a class visible in all subclasses regardless of what package they are in, which one of the following keywords would achieve this?

a. Private

b. Protected

c. Public

d. Private protected

1.8 The use of protected keyword to a member in a class will restrict its visibility as follows

a. Visible only in the class and its subclass in the same package.

b. Visible only inside the same package.

c. Visible in all classes in the same package and subclasses in other packages.

d. Visible only in the class where it is declared.

1.9 Class rectangle

```
{
        static final int width = 40;
        int height;
        public rectangle (int w, int h)
        {
                width = w;
        height = h;
        }
}
```

When we create new object using

Rectangle rect = new Rectangle (50, 50);

What will happen when you compile the program

a. The program compiles and runs fine.

b. Gets Runtime error in line 5.

c. The compiler objects to line 5.

d. The compiler objects to line 6.

1.10 class Rectangle

```
{
        Static private int rc = 0;
        public int width;
        int height;
```

```
        static int addrc( )
        {
                rc ++;
                width = 10;
                return rc;
        }
        public Rectangle( )
        {
                height = addrc( );
        }
}
```

What happens when you compile the program?

a. A class compiles and each rectangle will get a unique number.

b. The compiler objects to line 9.

c. You cannot store value of rc into height.

1.11 Members of a class specified as private are accessible only to the methods of the class.

a. True

b. False

1.12 A method declared as static cannot access non-static class members.

a. True

b. False

1.13 A static class method can be invoked by simply using the name of the method alone.

a. True

b. False

1.14 Every method of a final class is implicitly final.

a. True

b. False

1.15 Why would a responsible programmer want to use a nested class ?

a. To impress the boss with his/her knowledge of Java by using nested classes.

b. To keep the code for a very specialized class in close association with the class it works with.

c. To support a new user interface that generates custom events.

1.16 When programming a local inner class inside a method code block, which of the following statements is true?

a. The inner class will have only access to static variables in the enclosing class.

b. The inner class can use any variables declared in the method.

c. The inner class can use only local variables that are declared final.

d. The inner class can use only local variables that are declared static.

2. Design a class to represent an employee. Include the following members

   – Employee Number

   – Name

   – Address

   – Ph No.

   – Department

   – JDT

   Use overloaded constructors to

   a. provide initial values to Name, employee Number, Address and Ph No.

   b. to give initial values to all fields except Department

   Write method to assign department to the employee. Find out total number of employee objects created in the program. Display the data of all the employees.

3. Design a class to represent a bank account. It includes the following members

   Data members

   – Account Number

   – Name

   – Type of Account

   – Balance amount

   Methods

   – To give initial values for type of account and balance

   – To deposit an amount

   – To withdraw an amount

   – To display name and balance

   Initial value for Account Number and Name has to be taken from command line.

9.15 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book*, OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2*, Tata McGraw-Hill.0

UNIT

10

INHERITANCE

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Define inheritance
- Define subclass
- Describe member access and inheritance
- Describe super class variable and sub class object
- Describe the usage of super to call superclass constructors
- Describe multilevel hierarchy
- Describe how to call constructor
- Describe method overriding
- Describe abstract classes methods

U N I T S T R U C T U R E

10.1 INTRODUCTION

In the previous unit you learnt about adding methods to the class and writing methods with/ without parameters. You also studied the concept of recursion, meaning and uses of access control, static and final keywords. The concept of overloading was also discussed in the previous

unit. In this unit you will be able to extend the concept of overloading in inheritance. In the previous unit you have studied the definition of public and private in detail. In this unit you will be able to appreciate the meaning and usage of protected modifier in detail. You will also be able to use super keyword to refer to the constructor, method or variable of the superclass. Finally, you will learn about the Object class.

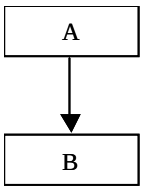
10.2 INHERITANCE BASICS

Reusability is one of the important aspect of the OOP paradigm. Java supports the concept of reusing the code that already exists. Inheritance allows the creation of hierarchical classifications. Using inheritance, you can create a general class that defines traits common to a set of related items. This class can then be inherited by other, more specific classes each adding those things that are unique to it. A class that is inherited is called superclass. The class that does the inheriting is called a subclass.

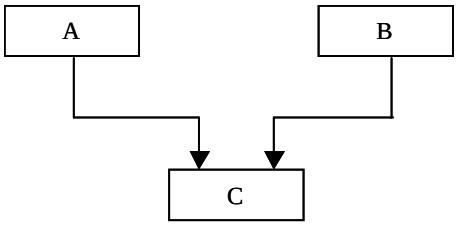
What is Inheritance

Inheritance defines a relationship between classes where one class shares the data structure and behaviours of another class. It enables software reuse by allowing you to create a new class based on the properties of an existing class. As a result, the developer is able to achieve greater productivity in a short time. The mechanism of deriving a new class from an old one is called inheritance. Inheritance may take different forms:

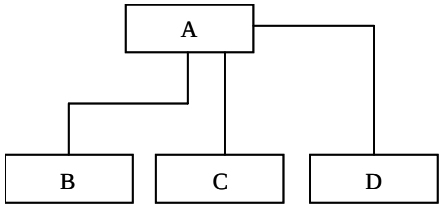
1. Single Inheritance



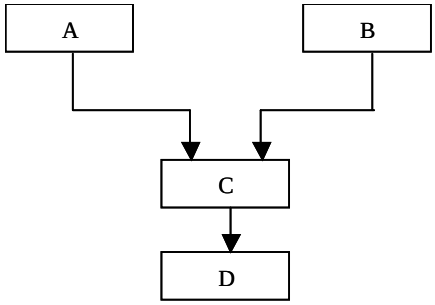
2. Multiple Inheritance (interfaces)



3. Hierarchical Inheritance



4. Multilevel Inheritance



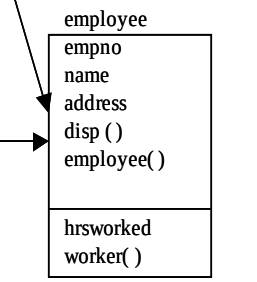
Defining Subclass

```
class subclassname extends superclassname
{
        variable declarations;
        method declarations;
}
```

The subclass now contains its own variables and methods as well as those of the superclass.

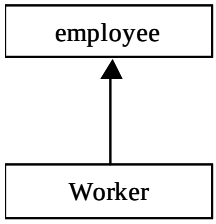
```
class employee
{
        int empno;
        String name;
        String address;
        employee (int eno, String ename, String addr)
        {
                empno = eno;
                name = ename;
                address = addr;
        }
        disp( )
        {
                System.out.println ("eno" + empno + "name" + name);
        }
}
```

```
class worker extends employee
{
        int hrsworked;
        worker (int eno, String ename,
        String addr, int hr)
        {
                super (eno, ename, addr);
                hrsworked = hr;
        }
}
```



Object of Worker class will have all the above variables and methods

```
class Test
{
        public static void main(String args[ ])
        {
                worker W1 = new worker (1, "ABC", "XYZ", 5);
                W1.disp( );
        }
}
```



The employee class defines the attributes and methods that are relevant for all kinds of worker's details. Attributes such as empno, name and address. Methods such as disp( ) to display

0employee's empno and name. You might need to represent a specific type of employee in a particular way. You can use inheritance to define a separate subclass of employee for each different type of employee. For example you might define manager, engineer, scientist etc.

Each subclass automatically inherits the attributes and methods of employee, but can provide additional attributes and methods. Attributes such as number of projects, type of project, duration of project, status can be stored in scientist class. Methods such as reviewing the project progress, setting the current status can be added in the scientist subclass. Subclass can also override a method from the superclass if they want to provide more specialized behaviour for the method. If the subclass method has the same name, parameter list and return type as a superclass method, then you can say that the subclass method overrides the superclass method.

10.3 MEMBER ACCESS AND INHERITANCE

A subclass includes all of the members of its superclass; it cannot access those members of the superclass that have been declared as private. When you create an object, you can call any of its public methods plus any public methods declared in its super class.

```
class A
{
        int i;
        private int j ;
        void getdata (int x, iny y)
        {
                i = x;
                j = y;
        }
}
class B extends A
{
        int total;
        void sum( )
        {
                total = i + j;
        }
}
class AB
{
        public static void main (String args[ ])
        {
                B Sub0b = new B( );
                Sub0b.getdata (10, 12);
                Sub0b.sum( );
                System.out.println ("Total" + Sub0b.total);
        }
}
```

The program will not compile because the reference to j inside the sum( ) method of B causes an access violation. Subclass has no access to j. A class member that has been declared as private will remain private to its class; it is not accessible by any code outside its class, including subclasses.

10.4 SUPER CLASS VARIABLE AND SUB CLASS OBJECT

A reference variable of a superclass can be assigned a reference to any subclass derived from that superclass.

```
class Box
{
        double w;
        double h;
        double d;
        Box (Box 0b)
        {
                w = 0b.w;
                h = 0b h;
                d = 0b.d;
        }
        Box (double x, double y, double z)
        {
                w = x;
                h = y;
                d = z;
        }
        double volume( )
        {
                return w * h * d;
        }
}
class BoxWeight extends Box
{
        double weight;
        BoxWeight (double x, double y, double z, double r)
        {
                w = x;
                h = y;
                d = z;
                weight = r;
        }
}
class RefDemo
```

```
{
        public static void main (String args[ ])
        {
                BoxWeight weightbox = new BoxWeight (3,5,7,8.37);
                Box plainbox = new Box( );
                double vol;
                vol = weightbox.volume( );
                System.out.println ("Volume =" + vol);
                plainbox = weightbox;
                vol = plainbox. volume( );
                System.out.println ("Volume of plainBox is" + vol);
                System.out.println ("Weight of plainBox is" +
                                    plainbox.weight);
        }
}
```

Since BoxWeight is a subclass of Box, it is permissible to assign plainbox a reference to the weight box object. When a reference to a subclass object is assigned to a superclass reference variable, you will have access only to those parts of the object defined by superclass. Plainbox can't access weight, even when it refers to a BoxWeight object, because superclass has no knowledge of what a subclass adds to.

10.5 USING SUPER TO CALL SUPERCLASS CONSTRUCTORS

Super has two general forms. The first calls the superclass constructor. The second is used to access a member of the superclass that has been hidden by a member of a subclass

- Super (parameter-list);
- Super membername;

A subclass constructor is used to construct the instance variables of both the subclass and the super class. The super keyword invokes the constructor method of the super class. Super may only be used within a subclass constructor method as the first statement. The parameter in the super call must match the order/type of the instance variable declared in the superclass.

```
class Box
{
        double w;
        double h;
                double d;
        Box (double x, double y, double z)
        {
                w = x;
                h = y;
                d = z;
        }
}
```

```
class BoxWeight extends Box
{
        double weight;
        BoxWeight (double x, double y, double z, double m)
        {
                super (x, y, z);
                weight = m;
        }
}
class demoSuper
{
        public static void main (String args[ ])
        {
                BoxWeight mybox1 = new BoxWeight(10, 20, 15, 34.3);
                System.out.println ("Weight of mybox1 is =
                                    "+mybox1.weight);
        }
}
```

When a subclass calls super( ), it is calling the constructor of its immediate superclass. Call to super( ) avoids duplicating the code in subclass but it implies that subclass must be granted access to these members. However, in some cases we would like to keep the variable private to itself. Access to these private variables of the superclass can be achieved by the keyword super.

10.6 ANOTHER USE OF SUPER

The second form of super always refers to the superclass of the subclass in which it is used. It is most applicable to the situation in which member names of a subclass hide members by the same name in the superclass.

Example:

```
class A
{
        int i;
}
class B extends A
{
        int i;
        B (int a, int b)
        {
                super.i = a;          // i in A
                i = b;                        // i in B
        }
}
class demosuper
{
```

```
        public static void main (String args[ ])
        {
                B b1 = new B(1, 2);
        }
}
```

In the above example, class A and class B both have instance variable called i. Class B will also inherit superclass version of variable i. To refer to superclass version of variable i, use super.i.

10.7 MULTILEVEL HIERARCHY

In multilevel hierarchy, each subclass inherits all of the traits found in all of its superclasses. It is perfectly acceptable to use a subclass as superclass of another.

```
class Box
{
        private double width;
        private double height;
        private double depth;
        Box (double w, double h, double d)
        {
                width = w;
                height = h;
                depth = d;
        }
double volume( )
{
        return (width * height * depth);
}
}
class BoxWeight extends Box
{
        double weight;
        BoxWeight (double w, double h, double d, double m)
        {
                super (w, h, d);
                weight = m;
        }
}
class shipment extends BoxWeight
{
        double cost;
        Shipment(double w, double h, double d, double m, double c)
        {
```

```
        super (w, h, d, m);
        cost = c;
    }
    class DemoShipment
    {
        public static void main (String args[ ])
        {
            Shipment Shipment1 =
            new Shipment (10, 20, 15, 10, 3.41);
            double vol;
            vol = shipment1.volume( );
            System.out.prinln ("Volume =" + vol);
            System.out.println ("Weight" + Shipment1.weight);
            System.out.println("ShippingCost"+,Shipment1.cost);
        }
    }
```

Because of inheritance, shipment can make use of previously defined classes of Box, BoxWeight, adding only the extra information it needs for its own specific application. Hence, inheritance allows reuse of code.

In the above example, subclass BoxWeight is used as a superclass to create the subclass called Shipment. Shipment inherits all of the traits of BoxWeight and Box and adds a variable called cost.

10.8 CALLING CONSTRUCTOR

Java compiler does a lot of extra work in compiling constructor code. If the first line of the code body doesn't have super( ) constructor, the compiler inserts "super( )", an invocation of the default constructor that takes no parameters belonging to parent class.

In a class hierarchy, constructors are called in order of derivation, from superclass to subclass.

For example:

```
    class A
    {
        A( )
        {
            System.out.println ("Inside A's constructor");
        }
    }
    class B extends A
    {
        B( )
        {
            System.out.println ("Inside B's constructor");
        }
```

```
}
class C extends B
{
        C( )
        {
                System.out.println ("Inside C's constructor");
        }
}
class democonstructor
{
        public static void main (String args[ ])
        {
                C c1 = new C( );
        }
}
Output :        Inside A's constructor
                Inside B's constructor
                Inside C's constructor
```

A class can have any number of constructors as long as they have different signatures. One constructor can invoke another but the statement has to be the first one in code body. Constructors are not inherited, so each derived class must specially implement any constructors it needs. However, derived class can make use of its parent class constructor using the super keyword.

For example

```
public class employee
{
        int empno;
        String name;
        String address;
        employee (int e, String n, String a)
        {
                empno = e;
                name = n;
                address = a;
        }
}
class worker extends employee
{
        int noofhrs
        worker (int c, String n, String a, int h)
        {
                super (e, n, a); // calls superclass constructor
```

```
                noofhrs = h;
        }
}
```

Points to remember

- Super may only be used in subclass.
- The call to superclass constructor must appear as the first statement in the subclass.
- The parameter in the super( ) must match the order and type of superclass constructor.
- If you don't define any constructor in the class code, the compiler will provide a default constructor that takes no parameter.

Student Activity 1

1. What is Inheritance?
2. How will you define a sub class?
3. What is the use of super?
4. Which members are accessible by a subclass?
5. Write the precautions to be taken while calling a constructor.

10.9 OVERRIDING METHODS

I0nheritance enables us to define and use methods repeatedly in subclasses without having to define the methods again in subclass because subclass inherits all of the methods of its superclass. However, a subclass can modify the behaviour of a method in a superclass by overriding it. To override a superclass method, the subclass defines a method with exactly the same signature and return type as a method somewhere above it in the hierarchy. Then, when the method is called, the method defined in the subclass is invoked and executed instead of the one in super class. This is known as overriding. In the previous example we change worker class as follows:

```
class worker extends employee
{
        int hrsworked;
        worker (int eno, String ename, String addr, int hr)
        {
                super (eno, ename, addr);
                hrsworked = hr;
        }
disp( )
{
        System.out.println ("eno" + eno + "name" + name);
        System.out.println ("ename"+ename+"hrsWorked"+hrsworked);
}
```

The method in the subclass effectively hides the superclass method. Now, when disp( )method is called for, the worker object method defined in the worker object will be invoked rather than the one defined in the employee class. Further, in the above example we do not need to repeat the code which already exists in the parent class. We can modify disp( ) method of worker class as follows:

```
disp( )
{
super.disp( );
system.out.println ("HrsWorked" + hrsworked);
}
```

Note that the super notation can be used to execute a method in the immediate superclass.

There are certain restrictions on overriding methods:

- **Access Modifier** – An overriding method cannot be made more private than the method in the parent class.
- **Return Type** – The return type must be the same as in the parent class method.
- **Exception Thrown** – Any exception declared must be of the same class as that thrown by the parent class or of a subclass of that exception (will be discussed in later chapter).
- Methods that are declared final cannot be overriden. Methods that are declared private cannot be seen outside the class and therefore, cannot be overriden. However, because the private method name cannot be seen, you can reuse the method name in the subclass, but then this is not the same as overriding.

Difference Between Overloading and Overriding

The terms overloading and overriding are applied to situations in which you have multiple Java methods with the same name. Within a particular class, you can have more than one method with the same name as long as they differ in number, type and order of the input parameters. This is described as overloading. For Java, these are two different methods. The duplicate names are just for the convenience of the programmer. Compiler will not allow methods with same signature but with different return type, even if one method is declared in parent class and the other in the subclass.

If a subclass method has the same name, parameter list and return type as superclass method, you say that the subclass method overrides the superclass method.

Overloaded methods are resolved at compile time, based on the arguments you supply. Overriden methods are resolved at run time.

Now consider the following code:

```
worker w1 = new worker ( )
employee e = w1;
w1.disp( ) // which method is called
```

Casting w1 to employee reference does not change the object type. Because the JVM resolved method calls at runtime using the actual object, the worker version disp( ) method is executed.

10.10 ABSTRACT CLASSES METHOD

An abstract class is simply a class which cannot be instantiated. For example , employeeClass does not provide sufficient details to provide anything meaningful about the employee. It must either be a worker, manager, staff or employee on deputation. Only its nonabstract superclasses may be instantiated. If you try to instantiate abstract class, compiler flags an error.

An abstract method defined within an abstract class must be implemented by all of its subclasses. This technique allows the class designer to decide exactly what behaviours a subclass must be able to perform. Java provides abstract keyword, which indicates that a class or a method is abstract.

```
abstract class employee
{
```

```
        int empno;
        String name;
        abstract void getdata( );
        abstract void displaydata( );
}
```

When you declare abstract method you just provide the signature for the method which comprises of its name, its argument list and its return type. You do not provide body for it.

NOTE We cannot declare abstract constructor or abstract static method.

Any class with abstract method must also be declared as abstract. This is done so that all classes extending the abstract class and overriding the abstract method are compatible.

Abstract classes can contain methods that are not declared as abstract. Those methods can be overriden by the subclasses, but it is not mandatory.

In the above class employee, we create a subclass called worker and we do not override the getdata( ) method which is declared abstract in the super class. If user calls workerobj.getdata( ), it calls the method available in the super class and that is not the desired method. Hence, the solution could be to declare the method as abstract in the superclass and override it in the subclass.

10.11 FINAL AND INHERITANCE

All methods and variables can be overridden by default in the subclass. If we wish to prevent the subclass from overriding the members of the superclass, we can declare them as final using the keyword final. Many of the methods in java.net classes are final.

```
final int basic = 7000;
final void disp( ) { - - - - - - - - - - }
```

If a method is performing some vital operation, such as authorization checking, it should be declared final to prevent anyone from overriding the method and changing your security checks. Sometimes, we may like to prevent a class being further inherited for security reasons. A class that cannot be subclassed is called a final class.

```
final class worker {              }
```

Any attempt to inherit these classes will cause an error and compiler will not allow it. If you declare a class as final, it can never be extended by any other class. The Color class in java.awt is declared final. Final Classes and final Methods produce more efficient compiled code. If the compiler encounters an object reference to final class and you call a method using that object reference, the compiler does not perform run time method binding. Instead, the compiler can perform static binding and avoid the overhead of run-time polymorphism. Similarly, if you call a final method anywhere in your program, the compiler can call that method statically.

10.12 OBJECT CLASS

Object class is the base class for all Java objects. All classes in Java are automatically inherited from the root class called Object, which sits at the top of the inheritance tree. If a class does not specify an explicit superclass, as in the case with employee class (previous example), then the class is deemed to extend directly from 'Object', as if it were defined as follows :

```
public class employee extends Object
{
}
```

Here is a summary of the Object class methods.

- *clone* : Creates a new object by copying the current one and returns the reference which has to be typecasted to the particular type of object being cloned.
- *equals* : Compares an object to this one. It returns true if the two references point to the same object.
- *finalize* : For garbage collection related methods.
- *toString* : Returns a string representing the object. For classes that don't override the toString method, the String produced by toString method will be name of the Class@object's HashCodeValue.

Other methods in Object class are:

- getclass
- notify
- notifyall
- wait

getclass( ) method returns the class object used to create the object. Other methods will be discussed later.

Student Activity 2

1. What do you mean by overriding?
2. How will you override methods?
3. Differentiate between overloading and overriding.
4. What is an abstract method?
5. What is the use of final keyword?
6. Describe the methods available in an object class.

10.13 SUMMARY

- A subclass inherits all the variables and its methods in its superclass.
- You can specify additional variables, methods and override methods.
- Java supports single inheritance, hierarchical inheritance, multilevel inheritance.
- It supports multiple inheritance using interfaces.
- Member access can be controlled by public, private and protected keywords in subclass.
- Super is used to call superclass constructor, superclass method and to use superclass variables if the variable names are common in the child class and its superclass.
- Two methods with the same signature in a class are known as overloaded methods.
- Two methods with the same signature and also the same return type written one in child class and other in parent class is known as overriden methods.
- An abstract class cannot be instantiated.
- A class that cannot be subclassed is called a final class.
- Object class is the base class of all Java objects.

10.14 KEYWORDS

Inheritance : A relationship between classes where one class shares the data structure and behaviors of another class.

Abstract Class : A class that cannot be instantiated.

Object Class : The base class of all Java objects.

Final Class : A class that cannot be subclassed.

10.15 REVIEW QUESTIONS

1. Find out the errors in the following code, if any.

```
Class A
{
        int type;
        A (int t)
        {
                type = t;
        }
        Class B extends A
        {
                String x = " ";
        }
```

2. What is a constructor ? What are its special properties and how do we invoke it?
3. It is an error to have a method with the same signature in both superclass and its subclass. True/False.
4. Constructor must always invoke its superclass constructor in its first statement. True/ False.
5. It is perfectly legal to assign a subclass object to a superclass reference. True/False.
6. Every method of a final class is implicitly final. True/False.
7. Which of the following are legal statements to construct A type object ?
   a. A a = new A( );
   b. A a = new A (4, 2, 7)
   c. A a = new A (5, 6)
   d. A a = new A (10);
   e. A a = new A (Base (4, 5), 6)
8. Which of the following are overloading the method int sum (int x, int y) { }
   a. int sum (int x, int y, int z) { }
   b. float sum (int x, int y) { }
   c. int sum (float x, float y) { }
   d. int sum (int a, int b) { }
   e. float sum (int x, int y, float z) { }

9. Consider the following code

```
Class A
{
        public static static void main (String args
        [ ] )
        {
                class B b = new classB( );
        }
        A (int x) { }
}
class classB extends A
{
}
```

What will happen when we compile and run this code?

a. Compile and run successfully.

b. error : class A does not define no argument constructor.

c. error : Class B does not define no argument constructor.

d. error : There is no code in class B.

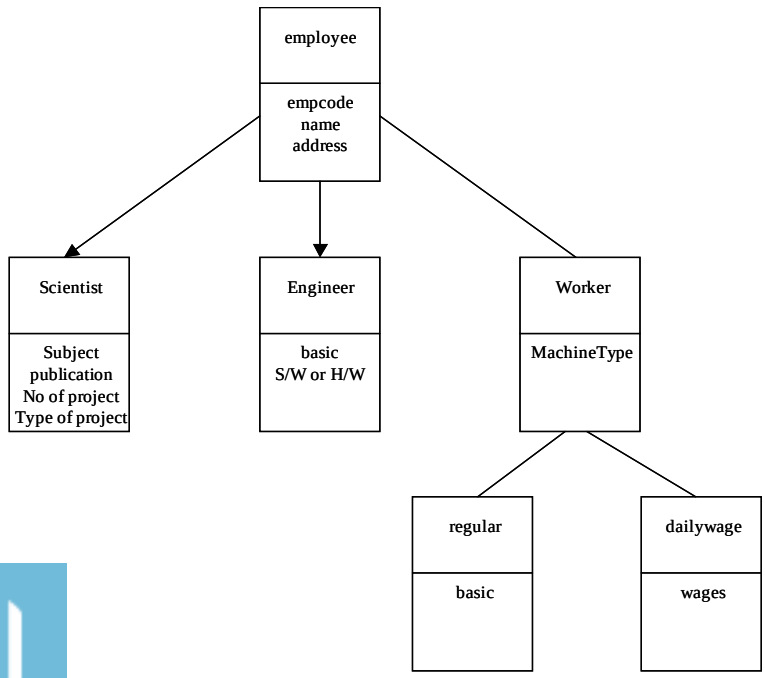
e. error : There is no code in constructor of class A(int x).

10. When do we declare a method or class final ?

11. When do we declare a method or class abstract ?

12. An organization wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationship is shown in the figure along with the minimum required information for each class.

Write a program to store data and retrieve individual information as and when required.



13. In Java, System.out.println is used to display data on the print stream.

In System.out.println

"System" is a predefined class. "Out" is object of printStream Class, and "println" is the method of printStream class.

Simulate System.out.println in your program such that if you invoke S.O.disp(int) it should be able to print the data on the screen.

10.16 FURTHER READINGS

Vangalur S. Alagar, R. Missaoui ; *Object Oriented Technology for Database and Software System* ; 1995, World Scientific.

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

UNIT

11

INTERFACES AND PACKAGES

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe interface
- Understand the implementation and application of interface
- Understand how to extend interfaces
- Describe a package
- Describe the benefits of a package
- Understand how to create packages
- Describe class path variables
- Describe some of the important packages.

U N I T S T R U C T U R E

11.1 INTRODUCTION

In the previous unit, we studied the concept of inheritance and also explored various forms of inheritance. We cannot do multiple inheritance in Java because we cannot create a subclass by extending more than one superclass. This way of multiple inheritance adds to the complexity in the program. Hence, it is not permitted in Java.

To achieve multiple inheritance in Java, we can implement multiple interfaces to create subclasses. At the end of this unit you should be able to use interfaces. One of the main features of OOP is its ability to reuse the code by extending the classes and implementing interfaces. If we need to reuse the classes from other programs without copying them, we use packages. We will study user defined packages and Java API packages. We will also study visibility and access control of different class members in relation to the packages.

11.2 DEFINING INTERFACE

An interface is like a fully abstract class, except that it cannot have any concrete method or instance variables. It is a collection of abstract method declarations and constants, that is, static final variables. This means that interfaces do not specify any code to implement these methods. Any class that implements an interface must implement all of the methods specified in that interface. A class can implement many interfaces but can extend only one class. The general syntax for defining interface is:

```
interface  InterfaceName
{
        variable  declaration;
        method  declaration;
}
```

Here interface is a keyword and interfaceName is any valid Java variable.

Example: interface Item

```
{
        static  final  int  code  =  1001;
        static  final  String  name  =  "fan";
        void  display  (  );
}
```

Variables are declared as constants using static final keywords. Note that the code for display ( ) is not included in the interface. The class that implements this interface must define the code for the method.

Implementation and Application

Interfaces are used as "superclasses" whose properties are inherited by classes. It is, therefore, necessary to create a class that inherits the given interface.

```
class  classname  extends  superclass
        implements  interface1,  interface2,  -  -  -  -
        {
        }
```

Class can extend another class while implementing interfaces.

Example:

```
interface  Area
{
        final  static  float  pi  =  3.14F
        float  compute  (float  x,  float  y);
}
class  Rectangle  implements  Area
{
        public  float  compute  (float  x,  float  y)
        {
                return  (x  *  y);
```

```
        }
}
class Circle implements Area
{
        public float compute (float x, float y)
        {
                return (pi * x * x);
        }
}
class InterfaceTest
{
        public static void main (String args[ ])
        {
                Rectangle rect = new Rectangle ( );
                Circle cir = new Circle ( );
                Area area;          // interface object
                area = rect;
                System.out.println (area.compute(10, 20));
                area = cir;
                System.out.println ("Area of Circle" +
                            area.compute(10,0));
        }
}
```

As you can see, the version of compute( ) that is called is determined by the type of object that area refers to at runtime. Note that if a class that implements an interface does not implement all the methods of the interface, then the class becomes an abstract class and cannot be instantiated. When you implement an interface method, it must be declared as public.

Variables in Interfaces

Interface can be used to declare a set of constants that can be used in different classes. The constant values will be available to any class that implements the interface.

```
class Students
{
        int rollNumber;
        void getNumber (int n)
        {
                rollNumber = n;
        }
        void putNumber( )
        {
                System.out.println ("RollNo." + rollNumber);
        }
```

```
}
class Test extends Students
{
        float part1, part2;
        void getMarks (float m1, float m2)
        {
                part1 = m1;
                part2 = m2;
        }
        void putMarks ( )
        {
                System.out.println ("Marks obtained");
                System.out.println (part1);
                System.out.println (part2);
        }
}
interface sports
{
        float sport wt = 6.0
        void putwt ( );
}
class Results extends Test implements sports
{
        float total;
        public void putwt ( )
        {
                System.out.println ("sports wt = "+ sportwt);
        }
        void display ( )
        {
                total = part1 + part + sportwt;
                putNumber( );
                putMarks( );
                putwt( );
                System.out.println ("Total score"+ total);
        }
}
class ABC
{
        public static void main (String args[ ]))
```

```
{
        Results stud1 = new Results ( );
        stud1.getNumber (1, 2, 3, 4);
        stud1.getMarks (27.5, 33.0);
        stud1.display ( );
        }
}
```

In the above example, the class "Results" implements "Sports" interface. The calculation of total marks for object of "Results" class includes sports weightage also.

NOTE All methods specified in an interface are implicitly public and abstract. Any variable specified in an interface is implicitly public, static and final.

Extending Interfaces

Like classes, interfaces can also be extended. The new subinterface will inherit all the members of the super interface in the manner similar to subclasses.

```
interface ItemConstants
{
        int code = 1000;
        String name = "Fan";
}
interface Item extends ItemConstants
{
        void display ( );
}
```

While interfaces are allowed to extend to other interfaces, subinterfaces cannot define the method declared in the super interfaces. Instead, it is the responsibility of any class that implements the derived interface to define all the methods.

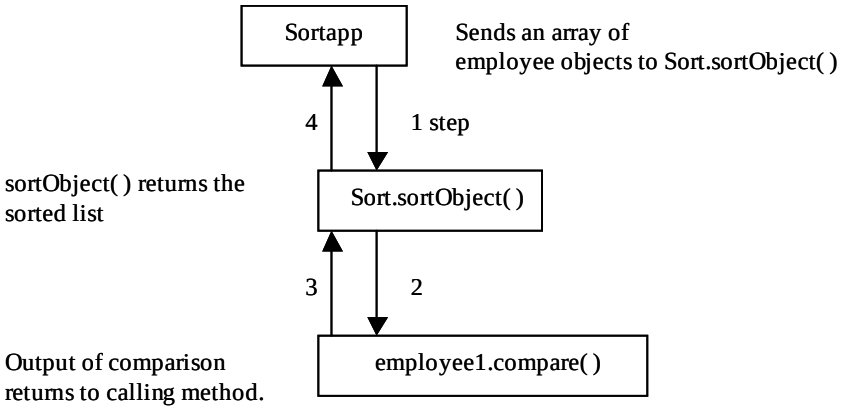
Interfaces can have access modifiers of public or blank, just like classes. An interface can be defined as extending another interface, similar to class hierarchy, but there is no base interface analogous to the Object class.

Another Example:

Sort is a classic example of the use of an interface. Many completely unrelated classes may use a sort. A sort is a well defined process that does not need to be written repeatedly.

The "Sortable" interface in the example specifies the methods required to make the sort work on each type of object that needs to be sorted. Only the class needs to know its object comparison or sorting rules.

Suppose there is a need to sort employee class database on the basis of empno. Employee class will be created by the organization which knows everything about the employee. Classes required to do sorting can be created by a separate developer who knows about sorting algorithm and nothing about employee class. Thus, example may use a sortable interface which declares one method i.e., compare( ). This method must be implemented by any class that wants to use the sort class methods. Sort class is an abstract class that contains sortObject methods to sort an array of objects. sortObject( ) calls compare( ) method on the objects in the array. Sortapp represents any application containing main that needs to sort list of employees.



The same method can be used to sort the department objects. (Implement this in Lab)

Student Activity 1

1. What is interface?
2. Give an example showing the implementation and application of interface.
3. How will you extend an interface?
4. Give any one classic example of interface.

11.3 WHAT IS A PACKAGE

One way of reusing the code is by extending classes and implementing the interfaces. But this is limited to reusing the classes within a program. If we need to use classes from some other programs without physically copying them into the program under consideration, we use packages. Packages act as containers for grouping logically related classes and interfaces.

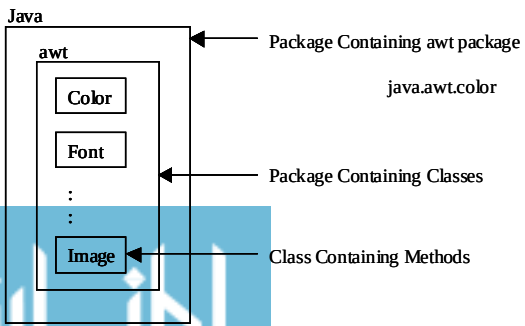
Benefits of Packages

1. Classes contained in packages of other programs can be easily reused.
2. Two classes in two different packages can have the same name. They can be uniquely identified by packagename.classname.
3. Packages also provide a way for separating "design" from "coding".
4. You can define classes inside a package that are not accessible by code outside that package. You can also define class members that are only exposed to other members of the same package.

In practical applications, we may have to build our own classes and use existing class libraries for designing user interfaces. Java packages are classified into two types:

- Java API package
- User defined packages

Using Java API packages



If we want to refer to color class in awt package once in our program, we can do so by

```
java.awt.color
```

But when we want to use the same class at many places or we want to use many classes from the same package we can use

```
import  packagename.classname;
        or
import  packagename.*;
```

The second statement imports every class contained in specified package. Packages can be named using the standard Java naming rules.

```
java.awt.Point  pts  [ ];
```

This statement declares an array of Point type objects using the fully qualified class name.

Creating Packages

To create a package, simply include a package command as the first statement in a Java source file. Any class declared within that file will belong to the specified package. If you omit the package statement, the class names are put into a default package which has no name.

```
package  first_package;
public  class  firstclass
{
}
```

The file should be stored as firstclass.java and should be located in a directory named first_package. The compiled class file should also be stored in the directory that has the same name as the package. Java also supports the concept of package hierarchy. This is done by specifying multiple names in a package statement, separated by dots – package firstpackage.secondPackage;

Store this package in a subdirectory named

```
first_package\secondPackage.
```

There is no practical limit on the depth of a package hierarchy except that which is imposed by file system. A Java package file can have more than one class definition. In such cases, only one of the classes may be declared as public and that class name with .Java extension is the source file name. More than one file can include the same package statement. The package statement simply specifies to which package the classes defined in a file belong. Most real world packages are spread across many files.

A simple example :

```
Package  firstpack;
class  Acct
{
        String  Name;
        int  AcctNo;
        float  Balance;
        Acct  (String,  int  A,  Float  Bal)
        {
                name  =  n;
                AcctNo  =  A;
```

```
                Balance = Bal;
        }
    void disp( )
    {
        if (Balance < 0 )
        System.out.println ("Warning");
        System.out.println ("name" + "Rs" + Balance);
    }
    }
    class AccBalance
    {
    public static void main (String args[ ] )
    {
        Acct currentAcct [ ] = new Acct [3];
        currentAcct [0] = new Act ("Anil", 1, 10000);
        current Acct [1] = new Acct ("Sunil", 2, 5000);
        currentAcct [2] = new Acct ("Abhinav", 3, 11000);
        for (int i = 0; i < 3; i++)
        currentAcct [i].disp( );
    }
    }
```

Store this file with the name AccBalance.java in a directory called firstpack. AccBalance.class should also be stored in the same directory. You can execute this program by giving following at the command line.

```
java firstpack.AccBalance
```

You need to set the appropriate classpath using set classpath command on the command line.

Importing Classes From Other Packages

Packages are a good mechanism for compartmentalizing diverse classes from each other. All of the standard classes are stored in some named packages. Classes within packages must be fully qualified with their package name or names with the dot operator. It would be tedious to type in the class name with long dot separated package. By importing a package, a class can be referred to directly, using only its name. In a Java source file, import statements occur immediately following the package statement and before any class definition.

Example:

```
        import java.util.*;
        class Mydate extends Date { }
                    OR
        class Mydate extends java.util.Date.
```

The general syntax would be

```
        import pkg1[.pkg2]. (classname/*);
```

All of the standard Java classes included with Java are stored in a package called java. The basic language functions are stored in package inside of the java package called java.lang.

Given below are two packages.

```
package pack1;
public class Teacher
{ - - - - }
public class student
{- - - - }
```

```
package pack2;
public class Courses
{- - - - - }
public class student
{- - - - - -}
```

```
import pack1.*;
import pack2.*;
pack1.student stud1;        // OK
pack2.student stud2;        // OK
Teacher teacher1;                   // OK
Courses course1;                    // OK
```

Subclassing an Imported Class

```
import package2.classB; // Only ClassB from package2 will be imported
class classC extends classB
{
        int n = 20,
        void displayC( )
        {
                System.out.println ("class C");
                System.out.println ("m = "+m);
                System.out.println ("n = "+ n);
        }
}
class ABC
{
        public static void main (String args[ ])
        {
                classC objc = new classC( );
                objc.displayC( );
        }
}
```

Note that the variables m, n have been declared as protected in classB. It would not have been possible if it had been declared as either private or default.

11.4 CLASSPATH VARIABLE

Every package must be mapped to a subdirectory of the same name in the file system. Nested packages will be reflected as a hierarchy of directories in the file system. For e.g. classfiles of java.awt.image must be stored under the directory java\awt\image.

You can put package directories anywhere in the file system. In addition, you can place whole directory structure of class files within .zip or .jar files. In order to tell the Java compiler where to locate the packages, you must set the CLASSPATH environment variable because the specific location that the Java compiler will consider as the root of any package hierarchy, is controlled by CLASSPATH.

```
SET  CLASSPATH  =  .;\ABC\myclass.jar
```

When any class reference is encountered, the directory list is searched from start to end. The JVM also checks for .jar or .zip entities in the CLASSPATH entries.

11.5 ACCESS PROTECTION

While using packages and inheritance in a program, we should take care of the visibility restrictions imposed by access protection modifiers. The access modifiers are private, public, protected. The least restrictive access modifier is public. Variables and methods declared as public may be seen and used by any other class.

If an access modifier is not specified (default), the variables and methods may be used by any other class within the same package. The next access modifier is protected. These variables can be seen from any subclass of that class. It may also be seen from any class within the package in which it exists.

The most restrictive access modifier is private. A private method or variable may not be accessed by any other class.

Access Modifier / Access Location	Public	Protected	Friendly	Private Protected	Private
Same Class	Y	Y	Y	Y	Y
Sub Class in Same Package	Y	Y	Y	Y	N
Other Classes in Same Package	Y	Y	Y	N	N
Sub Class in Other Package	Y	Y	N	Y	N
NonSub Class in other package	Y	N	N	N	N

11.6 IMPORTANT PACKAGES

a. java.lang : They include classes for primitive types, strings, math functions, threads and exceptions.

b. java.util : Language utility classes such as vectors, hash tables, random numbers, date etc.

c. java.io : They provide facility for the input and output of data.

d. java.awt : Include classes for windows, buttons, lists, menus and so on.

e. java.net : Include classes for communicating with local computers as well as with Internet server.

f. java.applet : Classes for creating applets.

g. java.sql : Classes for sending SQL statements to relational databases.

g. java.math : Classes for extended precision arithmetic.

Student Activity 2

1. What is a package?
2. What are the bene3fits of packages?
3. Name various types of Java packages.
4. How can a package be created?
5. Can a class be imported from other package? If Yes, How?
6. What are classpath variables?
7. List any five important packages.

11.7 SUMMARY

- An interface is a collection of abstract methods and final variables to be implemented in subclasses.
- A class can implement many interfaces.
- Implementing more that one interface is equivalent to multiple inheritance.
- Java has several levels of hierarchy for code organization, the highest of which is the package.
- Java packages can be accessed either using a fully qualified class name or using import statement. General syntax for import is

```
import pack1[.pack2] [.pack3]. class/*;
```

- A class stored in the package is executed using

```
java packname.classname.
```

- The access specifiers – private, public and protected – provide different categories of visibility for class members.

11.8 KEYWORDS

Interface : A collection of abstract methods and final variables to be implemented in subclasses.

Packages : A container for grouping logically related classes and interfaces.

11.9 REVIEW QUESTIONS

1. How do we design a package? How do we add classes or interfaces to a package? How do we execute a class stored in a package ?
2. Any class may be inherited by another class in the same package. True / False.

3. Members of a class specified as private are accessible only to the methods of the class. True / False.

4. Which keyword can protect a class in a package from accessibility by the classes outside the package ?

   a. Private

   b. Protected

   c. Final

   d. Default (no modifier)

5. We would like to make a member of a class visible in all subclasses regardless of what package they are in. Which keyword would achieve this ?

   a. Private

   b. Protected

   c. Public

   d. Private protected

6. Which of the following keywords are used to control access to a class member?

   a. Default

   b. Abstract

   c. Protected

   d. Interface public

   e. Public

7. A package is a collection of

   a. Classes

   b. Interfaces

   c. Editing tools

   d. Classes and interfaces

8. Study the following code.

```
package  p1;
public  class  student
{              }
class  test
{              }
```

```
import  p1.*
class  Result
{
        student  s1;
        Text  t1;
}
```

Code will not compile because

a. Class Result should be declared public.

b. Student class is not available.

c. Test class is not available.

d. package p1 is not complete.

9. Consider the following code.

```
interface Area
{
        float compute (float x, float y);
}
class Room implements Area
{
        float compute (float x, float y)
        {
return (x * y);
}
}
```

What is wrong in the code?

a. Interface definition is incomplete.

b. Method compute( ) in interface Area should be declared public.

c. Method compute( ) in class Room should be declared public.

d. All of the above.

10. Concept of multiple inheritance is implemented in Java by

a. Extending two or more classes.

b. Extending one class and implementing one or more interfaces.

c. Implementing two or more interfaces.

d. All the above.

11. Given below are two files.

Employee.java

```
package purchase;
public class Employee
{
        protected int age = 10;
}
```

org.java

```
import purchase.*;
public class org
{
        public static void main (String args[ ])
        {
        Employee e = new Employee( );
        System.out.println ("Age =" + e.age);
        }
}
```

Will the file org.java compile ? Yes or No? If No, why ?

12. Consider the example program written in section 10.1.2. Design a package to contain the class student and another package to contain the interface sports. Run the complete program.

13. Write a program to implement sort class explained in section 10.1.2.

14. Declare an interface called function that has a method named evaluate( ) that takes an arbitrary integer value as a parameter and returns an integer value. Create a class half that implements function. Make the implementation of method evaluate. Return the value obtained by dividing the integer value by 2. Create a method that takes an arbitrary array of integer values as parameters and return an array that has the same length but the value of an element in the new array is half that of the value in the corresponding element in the array passed as the parameter.

11.10 FURTHER READINGS

E.Balaguruswami, *Programming with Java*, Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now*, Microsoft Press.

Naughton, Patrick, *The Java Hand Book*, OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2*, Tata McGraw-Hill.

UNIT

12

EXCELPTION HANDLING

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe the fundamentals of exception handling
- Describe various types of exceptions
- Understand the use of Try and Catch keywords
- Understand the use of Throw, Throws and Finally keywords
- Understand some Java built-in exception
- Understand user defined exceptions

U N I T S T R U C T U R E

12.1 INTRODUCTION

As programs become more complicated, making them robust is a much more difficult task. Traditional programming languages like C rely on the heavy use of handlers and cryptic return codes for propagating the abnormal conditions back to the calling methods. Using an exception-handling mechanism, Java provides an elegant way to build programs that are both robust and clear. After completing this chapter you should be able to learn the concept of exception handling. You should be able to write code to catch, handle and throw exceptions and to create your own exceptions. This unit introduces you to exception handling capabilities built into Java and teaches you to write code to handle exceptions.

12.2 FUNDAMENTALS OF EXCEPTION HANDLING

An exception is a condition that is caused by a runtime error in the program. In reality, your program may have to deal with many unexpected problems such as missing files, bad user input, dropped network connections etc. If these abnormal conditions are not prevented or at least handled properly, either the program will be aborted abruptly or the incorrect results or status will be carried on, causing more and more abnormal conditions. Java provides an elegant approach to handling these problems with the exception mechanism.

A Java exception is an object that describes an exceptional condition that has occurred in a piece of code. When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the error. At some point, the exception is caught and processed. Exceptions can be generated by the Java run-time system or they can be manually generated by your code.

Java exception handling is managed by 5 keywords:

try, catch, throw, throws, finally.

When an error occurs, its mechanism performs the following tasks:

- Find the problem (hit the exception).
- Inform that an error has occurred (throw the exception).
- Receive the error information (catch the exception).
- Take corrective actions (handle the exception).

When an exception occurs within a Java method, the method creates an exception object and hands it over to the run-time system. This process is called throwing an exception. The exception object contains information about the exception including its type and the state of the program when the error occurred.

When a Java method throws an exception, the Java runtime system searches all the methods in the call stack to find one that can handle this type of exception. This is known as catching the exception. If the runtime system does not find an appropriate exception handler, the whole program terminates.

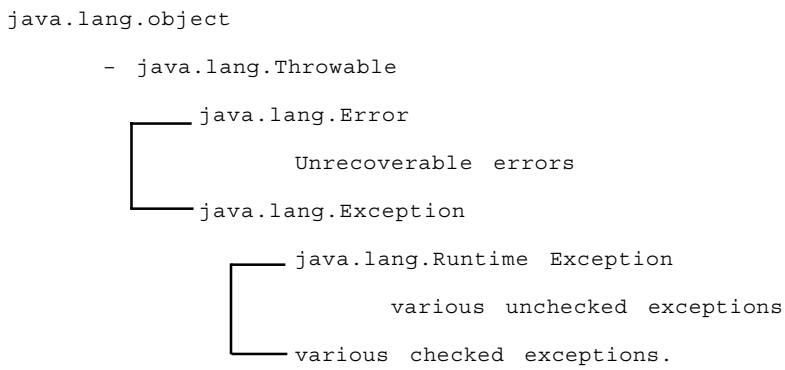
In traditional programming, error handling often makes the code more confusing to read. Java separates the details of handling errors from the main work of the program and they cannot be ignored.

The general form of exception handling block-

```
try
{
        // block of code to monitor for errors
}
catch (ExceptionType exob)
{
        // code to handle exception occurred
}
catch (ExceptionType exob2)
{
        // code to handle exception
}
```

12.3 TYPES OF EXCEPTIONS

All exception types are subclasses of the built in class Throwable. The hierarchy of Throwable class is shown in the diagram.



Exception class is used for exceptional conditions that user programs should catch. This is also the class to create your own custom exception. The java.lang.Error defines the exception, which are not expected to be caught. e.g. stack overflow. If an error is generated, it normally indicates a problem that will be fatal to the program.

Unchecked (run-time) exceptions are extensions of Runtime Exception class. Exceptions of this type are automatically defined for the program and include things like, divide by zero and invalid array indexing. If a runtime exception occurs and your program does not handle it, the JVM will terminate your program. Checked exceptions are extensions of the exception class – the exceptions must be caught and handled somewhere in your application. Checked exceptions must be handled either by try and catch block or by using throws clause in method declarations.

12.4 UNCAUGHT EXCEPTIONS

```
class Error2
{
        public static void main (String args[ ])
        {
                int a = 10;
                int b = 5;
                int c = 5;
                int x = a/b-c;
                System.out.println ("x = "+ x);
                int y = a / (b+c);
                System.out.println ("y =" + y);
        }
}
```

When the Java run-time system detects attempt to divide by zero, it constructs a new exception object and then throws this exception. This causes the execution of Error2 class to stops because once an exception has been thrown, it must be caught and dealt with. In this example, exception is caught by the default handler.

Any exception that is not caught by your program, will be processed by default handler. Java Runtime System generates an error condition which causes the default handler to print the error and to terminate the program.

The errors are printed by Stack Trace. The stack trace will always show the sequence of method invocations that led up to the error. The call stack is useful for debugging because it pinpoints the precise sequence of steps that led to the error.

12.5 TRY AND CATCH KEYWORDS

The programmer can provide for handling of exceptions using the Java keywords try and catch, with or without finally. Try and catch allow us to fix the error. It prevents the program from automatically terminating. A try statement is used to enclose a block of code in which an exception may occur.

The try block can have one or more statements that could generate an exception. If any one statement generates an exception, the remaining statements in the block are skipped and execution jumps to the catch block that is placed next to the try block. The catch block can have one or more statements that are necessary to process the exception. Every try statement should be followed by at least one catch statement. Catch statement is passed a single parameter which is reference to the exception object thrown. If the catch parameter matches with the type of exception object, then the exception is caught and statements in the catch block will be executed. Otherwise, the exception is not caught and the default exception handler will cause the execution to terminate. The scope of the variable named in the catch clause is restricted to the catch code block.

A succession of catch clause can be attached to a try statement. If the thrown exception cannot be assigned to the variable in the first catch clause, the JVM looks at the second and so on. If the code block in the try statement executes without throwing an exception, all of the catch clauses are ignored and execution resumes after the last catch.

The order of the catch clause must reflect the exception hierarchy, with the most specific exception first. If the first catch was the most general, none of the others could be reached.

```
class Error3
{
        public static void main (String args[ ])
        {
                int a = 10, b = 5, c = 5, x, y;
                try
                {
                        x = a/(b-c);
                }
                catch (Arithmetic Exception e)
                {
                System.out.println ("Division by zero");
                }
                y = a/(b+c);
                System.out.println ("y =" + y);
        }
}
Output :        Division by zero
                y = 1
```

Note that the program did not stop at the point of exceptional condition. It catches the error, prints appropriate error message and then continues the program execution after catch block.

12.6 THROW, THROWS AND FINALLY

It is possible for your program to throw an exception explicitly rather than let it be thrown by the Java run-time system. The exceptions that you throw can be standard system exceptions or you can create your own. The general syntex is

```
throw expression;
```

Here, expression is an object of type Throwable or subclass of Throwable. There are two ways of creating Throwable object:

- Using a parameter into catch clause.
- Creating one with the new operator.

```
throw new NullPointerException ("demo");

throw new ArithmeticException( );
```

The flow of execution stops immediately after the throw statement.

The nearest enclosing try block is inspected to see if it has a catch statement that matches the type of the execution. If not, then the next enclosing try statement is inspected and so on. If no matching catch is found, then default exception handler halts the program and prints stack trace.

```
class ThrowDemo
{
        static void demoproc( )
        {
                try
                {
                        throw new NullPointException ("Demo");
                }
                catch (NullPointerException e)
                {
        System.out.println ("Caught inside demoproc");
        throw e;
                }
        }
        public static void main (String args[ ])
        {
                try
                {
                        demoproc( );
                }
                catch (NullPointerException e)
                {
                        System.out.println ("Recaught" + e);
                }
        }
}
Output :        Caught inside demoproc
                Recaught : java.lang.NullPointerException : demo
```

main( ) method sets up an exception context and then calls demoproc( ). demoproc( ) method sets up another exception context and throws NULLPointerException. The exception is then rethrown.

Throws

The alternative to using try and catch wherever a checked exception occurs, is simply to declare that the method throws the exception. If a method is capable of causing an exception that it does not handle, it must specify this behaviour so that the callers of the method can guard themselves against that exception. In other words, a method that throws an exception within it must catch that exception or have that exception declared in its "throws" clause, unless the exception is a subclass of either the Error clause or the RuntimeException class.

A throws clause lists the type of exception that a method might throw. General form of throws clause:

```
type method name (parameter List) throws exceptionlist
{
        // body of method
}
```

Example: int calc (String filename) throws IOException, InterruptedException

When you have a method in a subclass that has the same name, parameter list and return type as a method in the parent class, the subclass method is said to override the parent class. The Java compiler will allow the overriding subclass method to be defined as, throwing the same exception as its superclass method, throwing one or more subclasses of superclass exception or not throwing any exceptions. It will not allow the overriding subclass to be declared as throwing a more general exception or exception from other hierarchy.

Example:

```
class Throwdemo
{
static void throwOne( ) throws IllegalAccessException
        {
                System.out.println ("Inside throwOne");
                throw new IllegalAccessException ("demo");
        }
        public static void main (String arge [ ])
        {
                try
                {
                        throwOne( );
                }
                catch (lllegalAccessException e)
                {
                        System.out.println ("Caught" + e);
                }
        }
}
Output :        inside throwOne
                Caught java.lang.IllegalAccessException : demo
```

The compiler relies on the declaration of throws clauses to determine if an exception may occur in an expression, a statement, or a method. The compiler issues an error message for any method that does not declare all exceptions in its throws clause.

Finally

The idea behind finally is that the programmer needs to have a way to correctly dispose of system resources, such as open files, no matter which exceptions are thrown by the code in a try block. If try statement has a finally clause attached, the code block associated with finally is always executed regardless of how the try block exits.

- Normal termination, by falling through the end brace.
- Because of return or break statement.
- Because an exception was thrown.

The finally clause is optional. However, each try statement requires at least one catch or a finally clause.

Example:

```
public int testx (String x)
{
        try
        {
                return someMethod(x);
        }
        catch (NullPointerException nex)
        {
                System.out.print ("Null Pointer,");
                return -1;
        }
        catch (Run timeException rex)
        {
                System.our.print ("Runtime");
                return -2;
        }
        finally
        {
                System.out.println ("Finally");
        }
}
        Output :      No exception thrown in SomeMethod( )
                      "Finally" is printed
```

If NullPointerException is thrown

"NullPointer, Finally" is printed

If ArithmeticException is thrown in someMethod( )

"Runtime, Finally" is printed.

An uncaught exception is thrown in someMethod "Finally" is printed.

Student Activity 1

1. What is an exception?
2. List the exception handling keywords available in Java.
3. Describe various types of exceptions.
4. What are uncaught exceptions?
5. What is the function of try and catch statements? Give an example to explain it.
6. How will you create throwable objects?
7. Describe the usage of throws and finally keywords.

12.7 NESTED TRY STATEMENTS

Try statement can be nested inside the block of another try. Each time a try statement is entered, the context of that exception is pushed on the stack. If inner try does not have a catch handler for particular exception, the stack is unwound and the next try statement's catch handlers are inspected for a match. This continues until one of the catch statements succeeds or until all of the nested try statements are exhausted.

```
class NestTry
{
        public static void main (String args [ ])
        {
                try
                {
                        int a = args.length;
                        int b = 42/a;
                        System.out.println ("a =" + a);
                        try
                        {
                                // nested try block
                                if (a = = 1)
                                a = 9/(a-a); (// zero divide)
                                if (a = = 2)
                                {
                                        int c[ ] = {1};
                        c [42] = 99; // out of bound array
                                }
                        }
                        catch (ArrayIndexoutofBoundExceptione)
                        {
System.out.prinln (Array index out-of-bound" + e);
                        }
                }
                catch (ArithmeticException e)
```

```
                    {
                    System.out.Println ("Divide by 0:" + e);
                    }
                }
            }
```

When no command line argument is passed, divide by-zero exception is generated by outer try block. With one command line argument, divide by zero is generated by nested try block. Since the inner block does not catch this exception, it is passed on to the outer try block. With two command line arguments, an array boundary exception is generated and handled by inner try block.

12.8 JAVA BUILT IN EXCEPTIONS

Inside the standard package java.lang, Java defines several exception classes. The most general exceptions are subclasses of type Runtime Exception. Since java.lang is implicitly added, most exceptions derived from Runtime Exception are automatically available.

EXCEPTION	MEANING
Arithmetic Exception	Arithmetic error, such as divide by zero.
ArrayIndexOutOfBoundException	Array index is out of bound.
ArrayStoreException	Assignment to an array element of an incompatible type
ClassCast Exception	Invalid cast.
IllegalArgumentException	Illegal argument used to invoke a method.
Illegal MonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
NegativeArraySizeException	Array created with a negative size.
Class NotFoundException	Class not found.
No SuchMethodException	A requested method does not exist.
InterruptedException	One thread has been interrupted by another thread.
StackOverflowException	Caused when the system runs out of stack space.

12.9 USER DEFINED EXCEPTIONS

You can define your own exception types to handle situations specific to your applications. It is done just by defining a subclass of Exception. All exceptions which you create have the methods defined by Throwable available to them. You can override one or more methods in exception class that you create. For e.g. String toString( ), void print StackTrace( ), String getMessage( ).

Subclassing

Exception is a subclass of Throwable.

Example:

```
import java.lang.exception;
class MyException extends Exception
```

```
{
        MyException (String message)
        {
                Super (message);
        }
}
class TestMyException
{
        public static void main (String args[ ])
        {
                int x = 5, y = 1000;
                try
                {
                        float z = (float)x / (float) y;
                        if (z < 0.01)
                        {
                throw new MyException ("Number is too small);
                        }
                }
                catch (MyException e)
                {
                System.out.println ("Caught my exception");
                        System.out.println (e.getMessage( ));
                }
                finally
                {
                System.out.println ("I am always here");
                }
        }
}
Output :        Caught my exception
                Number is too small
                I am always here
```

The object e which contains the error message "Number is too small" is caught by the catch block which then displays the message using the getMessage( ) method.

The common practice in creating a customized exception class is to subclass the "Exception" class. This ensures that compiler checks if it is dealt with properly. However, if you are writing system or hardware related utilities, you may be justified in creating subclasses from either Error or RuntimeException classes. You may choose to create exception classes in a hierarchy.

Student Activity 2

1. What is the advantage of nesting try statements?
2. Give any five Java built-in exceptions.
3. What are user defined exceptions?
4. Give an example to show that “exception is a subclass of Throwable.”

12.10 SUMMARY

- Java provides a clean and robust mechanism for handling abnormal conditions.
- First you try to execute block of statements. If an abnormal condition occurs, the system throws an exception and you catch the exception and finally there can be a code block you always want to execute.
- Exceptions are objects of "throwable" class.
- You can create your own exception classes as subclasses of the exception class. You can create a hierarchy of exception classes so that the handler has more flexibility in handling the exceptions.
- A method that may cause an exception to be thrown, must catch that exception or have that exception defined in its "throws" clause.

12.11 KEYWORDS

A Java exception is an object that describes an exceptional condition that has occurred in a piece of code.

Try and catch allow us to fix the error. It prevents the program from automatically terminating. A try statement is used to enclose a block of code in which an exception may occur.

12.12 REVIEW QUESTIONS

1. What is an exception ? How do we define try/catch block ?
2. What is a finally block ? When and how is it used ?
3. Is it essential to catch all types of exceptions ?
4. How many catch blocks can we use with one try block ?
5. A catch can have comma separated multiple arguments. True / False.
6. Throwing an exception always causes program termination. True / False.
7. What will be the output of

```
class test
{
        public static void main (String args[ ])
        {
                try
                {
                        double x = 0.0;
                        throw (new Exception ("Thrown"));
                        return;
                }
                catch (Exception e)
                {
                System.out.println ("Exception caught");
                        return;
                }
```

```
                finally
                {
                        System.out.println ("finally");
                }
        }
}
```

8. Complete the following code with given choices such that the method would calculate the sum of all the elements in an array of int.

```
public int total (int [ ] x)
{
        int i, t = 0;
        {
                t = t + x {i++];
        }
        return t;
                }
```

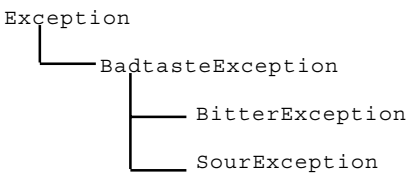
a. `for (int i = 0; i <x.length;)`

b. `for (i = 0; i <x.length;)`

c. `for (i = 0; i <x.length; i++)`

d. `for (i = 1; i <=x.length;; i++)`

9. You create custom class hierarchy for handling exception as follows :



Your application class Basecook has a method declared as follows:

int flavor (Ingredient list[ ]) throws BadtasteException

A class derived from Basecook has a method that overrides flavor. Which of the following are legal declarations of the overriding methods?

a. int flavor (Ingredient list [ ]) throws BadtexteException

b. int flavor (Ingredient list [ ]) throws BitterException

c. int flavor (Ingredient list [ ]) throws Exception

d. int flavor (Ingredient list[ ])

10. Assume that the above custom Exceptions have constructors taking a String parameter. Which of the following shows a correct complete statement.

a. new SourException ("hello);

b. throws new SourException ("hello");

c. throw new SourException ("hello");

d. throw SourException ("hello");

11. Define an exception "NoMatch found" that is thrown when a string is not equal to "UPTEC". Write a program that uses this exception.

12. Write a program that throws an exception whenever an attempt is made to divide a given number by an even number in a loop which runs from 1 to 10.

12.13 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book*, OSborne McGraw- Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

UNIT

13

MULTITHREADED PROGRAMMING

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe the Java thread model
- Describe a thread and its life cycle
- Describe priorities
- Describe synchronization
- Describe messaging
- Describe the thread class and runnable interface
- Describe how to create multiple threads
- Describe synchronization and deadlock
- Describe how to suspend, resume and stop threads.

U N I T S T R U C T U R E

13.1 INTRODUCTION

Java programs contain only single sequential flow of execution. A typical program is executed sequentially and is single threaded. If a single threaded program's execution is blocked while waiting for some I/O operation, no other portion of the program can proceed. All modern operating systems may execute multiple programs simultaneously, even if there is only a single CPU available

to run all the applications. Multithreading allows multiple tasks to execute concurrently within a single program. The advantage of multiple threads in a program is that it utilizes system resources (like CPU) better because other threads can grab CPU time when one line of execution is blocked. By using multithreading, you can create programs showing animation, play music, display document and download files from the network simultaneously. One unique feature of Java is the ease with which a programmer can write multithreaded programs. In C or C++, implementation of multithreading involves platform specific toolkit. Java was designed from the start to accomodate multithreading. In this unit you will learn how to create threads using thread class and runnable interface and how to allocate priority of execution to different threads in a program. You will be able to create a synchronized method to avoid collision in which more than one thread attempts to modify the same object at the same time.

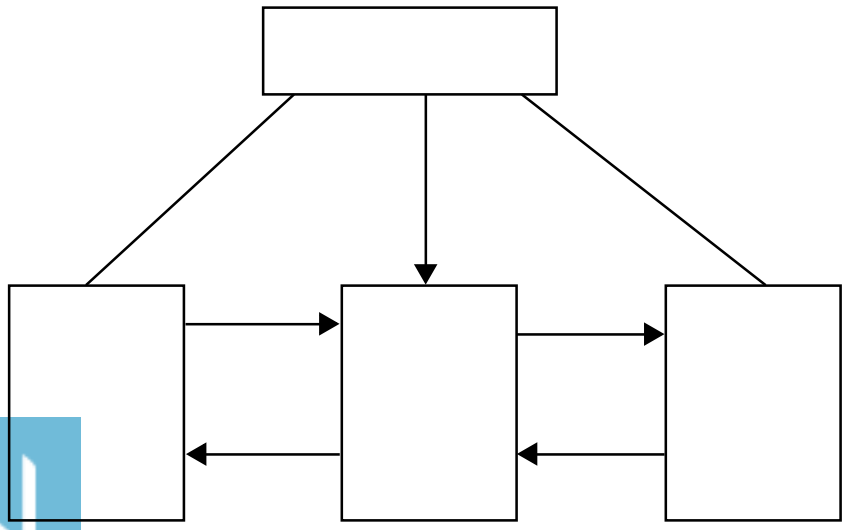
13.2 THE JAVA THREAD MODEL

Multithreading allows a running program to perform several tasks, apparently, at the same time. Java uses threads to enable the entire environment to be asynchronous. It makes maximum use of CPU because idle time can be kept to minimum. This helps reduce inefficiency by preventing the waste of CPU cycles. For example, it is useful for networked environment as data transfer over network is much slower than the rate at which the computer works. Multithreading lets you gain access to this idle time and put it to good use. Single-threaded system works on an event loop with polling approach.

In this model, a single thread of control runs in an infinite loop, polling a single event queue to decide what to do next. Once this polling mechanism returns with a signal (e.g. that a network file is ready to be read), then the event loop dispatches control to the appropriate event handler. Until this event handler returns, nothing can happen in the system. This wastes CPU time. The benefit of Java multithreading is that the main loop / polling mechanism is eliminated.

The Java runtime system depends on threads for many things and all class libraries are designed with multithreading in mind. Threads are used extensively in Java enabled browsers such as Hot Java. A thread exists in several states.

- Running thread
- Ready to run as soon as it gets CPU time
- Suspended thread
- Suspended thread can be resumed
- Thread can be blocked
- Thread can be terminated.



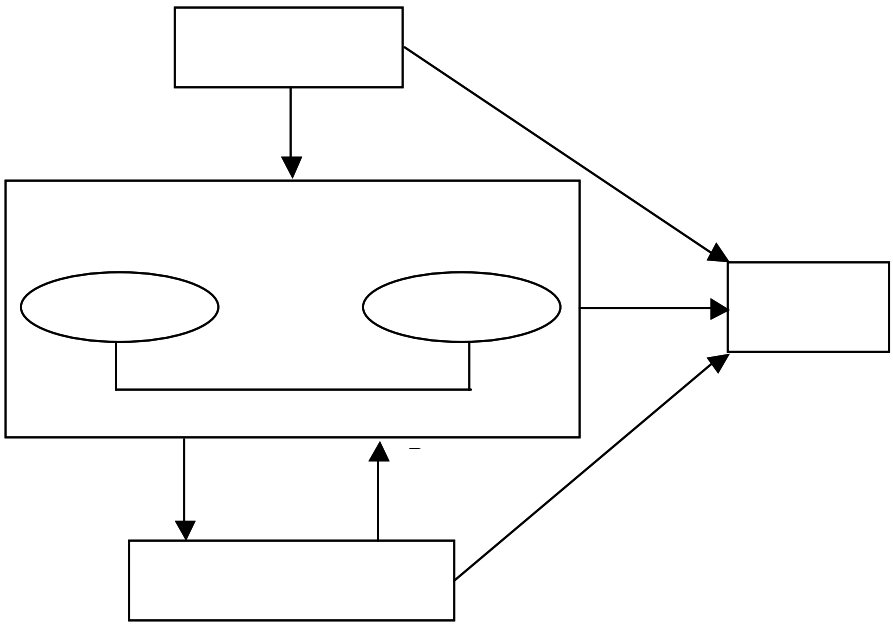
What is a Thread

Any single path of execution is called thread. The path is defined by elements such as stack (for handling calls to other methods with the code) and set of registers to store temporary memory. These elements are collectively known as context. Every program has at least one thread. Each thread has its own stack, priority and virtual set of registers. If there are two pieces of code that do not have the same context, then they are executed in two different threads.

Threads subdivide the runtime behaviour of a program into separate, independently running subtasks. Switching between threads is carried out automatically by JVM. Even in a single processor system, thread provides an illusion of carrying out several tasks simultaneously. This is possible because switching between threads happens very fast.

Thread's Life Cycle

A thread is always in one of five states.



When we create a thread object, it is said to be in **newborn state**. At this stage we can start the thread using start( ) method or kill it using stop( ) method. The thread in **runnable state** means that the thread is ready for execution and is waiting for CPU time. **Running state** means CPU has given its time to the thread for execution.

A thread is said to be in **blocked state** when it is suspended, sleeping or waiting in order to satisfy some condition. The suspended thread can be revived by using the resume( ) method. The thread which is sleeping can enter into runnable state as soon as the specified time period is elapsed. The thread in the wait( ) state can be scheduled to run again using the notify( ) method. The thread dies when its run( ) method completes its execution. A thread can be killed using the stop( ) method.

Where are Threads Used ?

Word processing program handles a print request in a separate thread. Many users find it convenient to continue working in wordprocessing application even after submitting print request. Downloading a file from Internet and browsing another site are implemented in different threads in the browser program. The Oracle database is another example of a program that uses threads. Database "connect" request to a common resource is handled by different threads. Operating Systems (specially networked OS) are written using threads. Operating systems usually run many applications such as a web browser and a word processing program at the same time. A multithreaded program can indirectly benefit the end user because it can efficiently use the CPU.

If a program is waiting for user response over the network, during this waiting time other useful tasks can be performed simultaneously. A thread program can directly benefit the user in applications that employ GUI components. When a button is pressed to execute one task, the program can make other buttons available to perform other independent programs.

Student Activity 1

1. What is the advantage of multithreading?
2. What is a thread?
3. Name various states in which a thread exist.
4. Describe various states of a thread's life cycle.
5. Where are threads used?

13.3 PRIORITIES

In Java, each thread is assigned a priority which affects the order in which the thread is scheduled. A newly created thread inherits the priority of the thread that created it. Thread priorities are integers that specify the relative priority of one thread over another. Thread's priority is used to decide when to switch from one running thread to the next. The switching can be implemented by yielding, sleeping or blocking a thread and CPU time is given to the highest priority thread. The high priority threads can dominate the processor.

A lower priority thread can be preempted by a higher-priority thread. In cases where two threads have the same priority, for e.g., operating system such as windows 98, are time sliced automatically in a round-robin fashion. For some other OS, threads of equal priority must voluntarily give control to others. The implementation of Java threads depends on the scheduling policy of the Operating System (OS).

Priority of a thread is set using setPriority( ) method. The Priority values are MIN_PRIORITY = 1, NORM_PRIORITY = 5 MAX_PRIORITY = 10. If the value is larger than the maximum priority value of its parent thread group, the max priority (of the group) value will be used.

In general, you should set process-intensive threads to a lower priority than threads that must respond quickly to user-interface events such as button click. Most of the time, the thread waiting for user interaction event is blocked which allows the lower-priority thread to run. However, as soon as the user interface event occcurs, the higher priority thread will immediately preempt a low priority thread and pushes the low priority thread into the queue for its next turn, and keeps the user happy. This is called preemptive scheduling. In some cases, thread execution is time-sliced. Even threads with lower priorities will get a small portion of execution time, roughly proportional to their priority values.

13.4 SYNCHRONIZATION

Synchronization is the way to avoid data corruption caused by simultaneous access to the same data. Multithreading introduces an asynchronous behaviour in your program. This makes it possible for two threads to run synchronously so that they can share the data between them e.g. one thread is writing data and other is in the middle of reading data. Suppose that you have a program to handle a user's bank account. To deposit money in the account, program gets the current balance from the remote server which may take as long as five seconds. The program adds newly deposited amount to current balance and sends the updated amount to remote server which may again take five seconds to complete. If two deposit threads, each making a Rs 5000 deposit are started roughly at the same time, database may reflect the result of just one deposit. Such a conflict should not occur. Java maintains interprocess synchronization through object's synchronized method. Once a thread is inside a synchronized method, no other thread can call any other synchronized method on the same object. Each object has an associated lock variable that can be manipulated by the JVM. This lock provides a mechanism that can be used to allow only one thread at a time to have access to an object.

Since it would take additional time for the JVM to check the lock condition of an object every time it is accessed, the lock is ignored by default. The keyword synchronized indicates a method or block to be guarded by the lock mechanism.

Once obtained, a thread's lock on an object is not released until the thread exits the synchronized code block or user uses wait( ) method in the object class. The synchronized keyword affects only the code block in the original class.

A synchronized method consumes extra CPU time on entry and exit, so you should use synchronized method for a good cause.

Any number of threads can be ready to execute a synchronized code block. A single thread can obtain locks on many objects and / or multiple locks on the same object. The JVM ensures that the lock is removed when a thread exits the code block.

13.5 MESSAGING

In a multithreaded program, you need to define how the threads will communicate with each other. Java implements messaging through calls to predefined methods that all object's have e.g. notify( ), notifyall( ). In most of the other languages, you must depend on the operating system to establish communication between threads.

13.6 THE THREAD CLASS AND RUNNABLE INTERFACE

Thread programming has two parts. In the first part we create a thread class. In the second part we actually run the thread. A new thread can be created in two different ways.

- Define a class that extends thread class and override its run( ) method.

  The thread class defines several methods.

  getName( ), getPriority( ), isAlive( ), join( ) run( ), sleep( ), start( )

- By converting a class to a thread by implementing runnable interface. The runnable interface has only one method run( ).

Every program runs in a thread. When a Java program starts up, one thread begins running immediately and is called the main thread of the program. The main thread is the thread from which other child threads will be spawned. Putting a thread to sleep is a technique to allow other threads to run. When the main thread stops, the program terminates. Main thread cannot be controlled by methods of the thread object. To do so, you must obtain a reference to the main thread using currentThread( ) method and then use the other methods on main method.

Example :

```
public static void main (String args[ ])
                {
                        Thread t = Thread.currentThread( );
                        t.setName ("new main"),
                }
```

To get more information about the thread following methods can be used:

getName( )	:	Returns the current name of the thread.
getPriority ( )	:	Returns the current priority of the thread.
isAlive( )	:	Returns true if the thread is started but is not yet dead.
join( )	:	Waits for the thread to terminate.
start( )	:	Puts the new thread into a runnable state.
run( )	:	When a thread is started, the scheduler calls its run( ) method.
sleep( )	:	Suspends a thread for a period of time.

13.7 CREATION OF THREADS

When you create a new thread object, it is created in blocked state. Calling a thread's start( ) method removes the initial block and calls the scheduler to call the run( ) method. This places the thread in a queue.

Creating Thread Using the Thread Class

Creation of thread is a two step process-

1. Create the new class:
   a. Define a subclass of thread.
   b. Override its run( ) method.
2. Instantiate and run the thread:
   a. Create an instance of class.
   b. Call its start( ) method.

This causes the scheduler to launch the new thread by calling its run( ) method and putting the thread in the queue for execution. When the other running threads release CPU, it will run.

```
class A extends Thread
{
        public void run( )
        {
                for (int i = 1; i < = 5; i++)
                {
        System.out.println ("\t From Thread A : i = "+ i);
                }
        System.out.println ("Exit from A");
        }
}
class B extends Thread
{
        public void run( )
        {
                for (int j = 1, j < = 5; j++)
                {
        System.out.println ("\t From Thread B : j = "+j);
                }
        System.out.println ("Exit from B");
        }
}
class C extends Thread
{
        public void run( )
```

```
        {
                for (int k = 1; k < = 5; k++)
        {
        System.out.println ("\t From Threadc : k = " +k);
        }
        System.out.println ("Exit from (");
        }
}
class ThreadTest
{
        public static void main( )
        {
                new A( ).start( );
                new B( ).start( );
                new C( ).start( );
        }
}
```

In the above program, main thread initiated three new threads and started them. The output of the program will not be sequential. They do not follow any specific order. They are running independently and each executes whenever it has a chance. Note that every time we run this program we get different output sequence.

Creating Threads Using Runnable Interface

To create a thread, declare a class that implements the runnable interface. To implement runnable, a class needs to only implement a single method called run( ). After you create a class that implements runnable, you will instantiate an object of type thread using the constructor of thread class.

Thread (Object of Runnable Interface)

The new thread will not start running until you call its start( ) method. Start( ) executes a call to the run( ) method.

class X implements Runnable

```
        {
                public void run( )
                for (int i = 1; i < = 10; i++)
                {
                        System.out.println ("\t Thread x: " +i);
                }
                System.out.println ("End of ThreadX");
        }
        class RunnableTest
        {
                public static void main( )
```

```
        {
                X runnable = new X( );
                Thread threadX = new Thread (runnable);
                threadX.start( );
        System.out.println ("End of main Thread");
        }
}
```

In main method, we first create an instance of X and then pass this instance as the initial value of the object thread X. Whenever the new thread threadX starts up, its run( ) method calls the run() method of the target object supplied to it.

In multithreaded program, the main thread must be the last thread to finish running. If the main thread finishes before a child thread has completed, then the Java run-time system may "hang". One simple method to finish child thread before the main thread is to use sleep method in the main thread.

Stopping and Blocking Thread

Sometimes, suspending execution of thread is useful. For example, a separated thread can be used to display the time of day. If the user doesn't want a clock, then its thread can be suspended. Whenever we want to stop a thread from running further, we may do so by calling its stop method.

```
aThread.stop( );
```

Thread automatically stops when it reaches the end of method. The stop( ) method may be used when the premature death of a thread is desired. The thread can also be temporarily suspended or blocked from entering into the runnable and subsequently running state by using –

```
sleep( )
suspend( )
wait( )
```

The thread will return to the runnable state when the specified time is elapsed in case of sleep( ). The resume( ) method is invoked in the case of suspend( ), and the notify( ) method is called in the case of wait( ).

Student Activity 2

1. Write a short note on the following :
   (a) Priorities
   (b) Synchronization
   (c) Messaging
2. How can you create a new thread?
3. How can you create a thread using runnable interface?
4. How can you suspend execution of a thread?

13.8 CREATING MULTIPLE THREADS

Using IsAlive( ) and Join( )

You can have threads communicate with each other through shared data and by using thread control methods e.g. suspend( ), resume( ) etc. All threads in the same program share the same memory space. If the reference to an object is visible to different threads, or explicitly passed to different threads, these threads share access to the data member of that object.

In multithreaded application, the main thread must be the last thread to finish. This can be accomplished by calling sleep( ) within main( ) with long enough delay to ensure that all child threads terminate prior to the main thread.

The threads communicate by waiting for each other. For example, the join( ) method can be used for the caller thread to wait for the completion of the called thread. Also, a thread can suspend itself and wait at some point using suspend( ) method; another thread can wake it up through the waiting thread's resume( ) method, and both threads can run concurrently.

One thread can know when another thread has ended using isAlive( ) method which returns true if the thread upon which it is called is still running. If the thread is alive, it does not mean it is running – it is in runnable state. Another method used for this purpose is join( ). This method waits until the thread on which it is called terminates. join( ) also allows you to specify a maximum amount of time that you want to wait for the specified thread to terminate.

```
class Nthread implements runnable
{
        string name;
        Thread t;
        NThread (String threadname)
        {
                name = threadname;
                t = new Thread (this, name);
                System.out.println ("New Thread" + t);
                t.start( );
        }
public void run( )
        {
                try
                {
                        for (int i = 5; i > 0; i - - )
                        {
                        System.out.println (name +" : " + i);
                        Thread.sleep (1000);
                        }
                }
                catch (InterruptedException e)
                {
                System.out.println (name + "interrupted");
                }
                System.out.println (name + "exiting");
        }
}

class DemoJoin
{
```

```
        public static void main( )
        {
                Nthread ob1 = new Nthread ("One");
                Nthread ob2 = new Nthread ("Two");
                Nthread ob3 = new Nthread ("Three");
                System.out.println ("Thread one is alive" +
                                    ob1.t.isAlive( );
                System.out.println ("Thread Two is alive" +
                                    ob2.t.isAlive( );
                System.out.println ("Thread Three is alive"
                                    + ob3.t.isAlive( );
                try
                {
        System.out.println ("waiting for thread to finish");
                        ob1.t.join( );
                        ob2.t.join( );
                        ob3.t.join( );
}
catch (InterruptedException e)
{
        System.out.println ("Main thread Interrupted");
}
        System.out.println ("Thread one is alive" + ob1.t.
                                        isAlive( );
        System.out.println ("Thread Two is alive" + ob2.t.
                                        isAlive( );
        System.out.println ("Thread Three is alive" +
                            ob3.t. isAlive( );
        System.out.println ("Main thread exiting");
}
}
Sample Output :     New thread : Thread [One, 5, main]
                    New thread : Thread [Two, 5, main]
                    New thread : Thread [Three, 5, main]
                    Thread Three is alive : true
                    waiting for thread to finish
                    One : 5
                    Two : 5
                    Three : 5
                    One : 4
                    Two : 4
                    Three : 4
                    One : 3
```

```
Two : 3
Three : 3
One : 2
Two : 2
Three : 2
One : 1
Two : 1
Three : 1
Two exiting
Three exiting
Thread One is alive : False
Thread Two is alive : False
Thread Three is alive : False
Main Thread exiting
```

NOTE After the calls to join( ), the threads have stopped.

Thread Priorities

Priorities are the way to make sure that important or time-critical threads are executed frequently or immediately. Thread priorities are used by the thread scheduler to decide when each thread should be allowed to run. In theory, higher priority threads get more CPU time than lower-priority threads. In practice, the amount of CPU time that a thread gets often depends on several factors besides its priority. (For e.g., how an OS implements multitasking). A higher priority thread can also preempt a lower priority one when higher priority thread resumes from sleep. Use setPriority( ) method to set a thread's priority.

```
final void setPriority (int level)
```

Here, level specifies the new priority setting for calling thread. The value of level must be within the range MIN_PRIORITY and MAX_PRIORITY. If the value to be set is outside the legal range an IllegalArgumentException will be thrown. To return a thread to default priority, specify NORM_PRIORITY. Using getPriority( ) method you can obtain the current priority settings.

```
class A extends Thread
{
        public void run( )
        {
                System.out.println ("threadA started");
                for (int i = 1; i < = 4; i ++)
                {
        System.out.println ("\t From Thread A : i = "+ i);
                }
                System.out.println ("Exit from A");
        }
}
        class B extends Thread
{
        public void run( )
```

```
        {
System.out.println ("thread B started");
for (int  = 1; j < = 4; j ++)
System.out.println ("\t From Thread B : j = " + j);
}
        System.out.println ("Exit from B");
}
class C extends Thread
{
        public void run( )
        {
        System.out.println ("thread c started");
        for (int k = 1; k < = 4; k++)
        {
        System.out.println ("\t from Thread C: k = "+k);
}
                System.out.println ("Exit form C");
        }
}
class ThreadPriority
{
        public static void main( )
        {
        A threadA = new A( );
        B threadB = new B( );
        C threadC = new C( );
        threadC. setPriority (Thread.MAX_PRIORITY);
        threadB. setPriority (Thread.get_Priority( )+1);
        threadA. setPriority (Thread.MIN_PRIORITY);
        System.out.println ("Start thread A");
        threadA. start( );
        System.out.println ("Start thread B");
        threadB. start( );
        System.out.println ("Start thread C");
        threadC. start( );
        System.out.println ("End of main thread");
        }
}
```

The above program illustrates the effect of assigning higher priority to a thread. Note that although the thread A started first, the higher priority thread B has preempted it and started printing the output first. Immediately, the thread C that has been assigned the highest priority takes control over the other two threads. The thread A is the last to complete.

13.9 SYNCHRONIZATION AND DEADLOCK

When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. For example, one thread may try to read a record from a file while another is still writing to the same file. This situation may give strange results. Java enables us to overcome this using synchronization. A key to synchronization is "monitor". When we declare a method synchronized, Java creates a "monitor" and hands it over to the thread that calls the method first time. A monitor is an object that is used as a mutually exclusive lock. Only one thread can own a monitor at a given time. While a thread is inside a synchronized method, all other threads that try to call it on the same instance have to wait. Whenever a thread has completed its work of using synchronized method, it will hand over the monitor to the next thread that is ready to use same resource.

The following program has three classes. The first one, Callme, has a single method call( ). This method prints the msgstring inside of square brackets. Notice that after call( ) prints the opening bracket and msg string it calls Thread.sleep(1000), which pauses the current thread for one second. The constructor of next class caller takes a reference to an instance of the Callme class and a String, which are stored in target and msg, respectively. The constructor also creates a new thread that will call this object's run( ) method. The run( ) method of caller calls the call( ) method on target instance of Callme, passing in msg string. Finally, Synch class starts by creating a single instance of Callme, and three instances of caller, each with a unique message string.

The result will be the mixed up output of the three message strings. In this program, nothing exists to stop all three threads from calling the same method, on the same object, at the same time.

```
class  Callme
{
        void  call  (String  msg)
        {
                System.out.println  ("["  +  msg);
                try
                {
                        Thread.sleep  (1000);
                }
                catch  (InterruptException  e)
                {
                        System.out.println  ("]");
                }
        }
class  Caller  implements  Runnable
String  msg;
Callme  target;
Thread  t;
public  Caller  (Callme  targ,  String  s)
{
        target  =  trag;
        msg  =  s;
        t  =  new  Thread  (this);
        t.start( );
```

```
}
        public void run ( )
{
        target.call (msg);
}
}
class Synch
{
        public static void main ( )
{
        Callme target = new Callme( );
        Caller ob1 = new Caller (target, "Hello");
        Caller ob2 = new Caller (target, "Synchronized");
        Caller ob3 = new Caller (garget, "World");
try
        {
                ob1.t.join( );
                ob2.t.join( );
                ob3.t.join( );
        }
        catch (interruptExectption e)
{
        System.out.println ("Interrupted");
}
}
}
Output :      [Hello [Synchronized [World]
              ]
              ]
Using Synchronized Statement
synchronized (object)
{
        // statement to be synchronized
}
```

In synchronized statement synchronized keyword has to be used with object reference. It does no good to synchronize local variables.

When we declare a method synchronized, Java creates a monitor and hands it over to the thread that calls the method first time. As long as the thread holds the monitor, no other thread can enter the synchronized section of code. Whenever a thread has completed its work of using synchronized method it will hand over the monitor to the next thread that is ready to use the same resource.

Deadlock

Situation may occur when two or more threads are waiting to gain control of a resource. Due to some reason, the condition on which the waiting threads rely on to gain control does not happen. This results in what is known as a deadlock. For example, assume that the thread 'A' must access 'MethodA' before it can release 'MethodB', but the thread 'B' cannot release 'MethodA' until it gets hold of MethodB. Because these create mutually exclusive conditions, a deadlock occurs. Suppose, one thread enters the monitor on object X and another thread enters the monitor on object Y. If the thread in X tries to call any synchronized method on Y, it will block as expected. However, if the thread in Y, in turn, tries to call any synchronized method on X, the thread waits forever, because to access X, it would have to release its own lock on Y so that the first thread could complete.

13.10 SUSPENDING, RESUMING AND STOPPING THREADS

The suspend( ) method of the thread class is deprecated in Java2 because suspend can sometimes cause serious system failure. Assume that a thread obtains locks on important resource and if that thread is suspended at that point, those locks are not relinquished. Other threads waiting for these resources can be deadlocked.

```
resume( ) method cannot be used without suspend( ).
```

The stop( ) method is also deprecated in Java2 because arbitrarily stopping a thread can leave an object in a damaged state that might then cause unpredictable results.

In Java2 we can suspend, resume or stop the thread by using wait( ), notify( ) methods.

```
class NThread implements Runnable
{
        String name;
        Thread t;
        boolean suspendFlag;

NThread (String threadname)
{
        name = threadname;
        t = new Thread (this, name);
        System.out.println ("New thread" + t);
        suspendFlag = false;
        t.start( );
}
public void run( ) {
try
{
        for (int i = 15; i > 0; i - -)
        {
                System.our.println (name + ": " +i);
                Thread.sleep (200);
                synchronized (this)
```

```
                        {
                        while (suspendFlag)
                        {
                                wait( );
                        }
                }
        }
}
catch (InterruptedException e)
        {
        System.out.println (name + "interrupted");
        }
        System.out.println (name + "exiting");
}
void mysuspend( )
{
        suspendFlag = true;
}
synchronized void myresume( )
{
        suspendFlag = false;
        notify( );
}
}
class SuspendResume
{
        public static void main( )
        {
                NThread ob1 = new NThread ("One");
                NThread ob2 = new NThread ("Two");
                try
                {
                Thread.sleep (1000);
                ob1.mysuspend( );
        System.out.println ("Suspending thread One");
        Thread.sleep (1000);
        ob1.myresume( );
        System.out.println ("Resuming thread One");
        ob2.mysuspend( );
        System.out.println ("Suspending thread Two");
```

```
                Thread.sleep (1000);
                ob2.myresume( );
                System.out.println ("Resuming thread Two");
                }
                catch (InterruptedException e)
                {
        System.our.println ("Main thread Interrupted");
                }
        try
{
        System.out.println ("Waiting for thread to finish");
        ob1.t.join( );
        ob2.t.join( );
}
catch (InterruptedException e)
{
        System.out.println ("Main thread Interrupted");
}
        System.out.println ("Main thread exiting");
}
}
New thread : Thread [One, 5, main]
One : 15
New thread : Thread [Two 5, main]
Two : 15
One : 14
Two : 14
One : 13
Two : 13
One : 12
Two : 12
One : 11
Two : 11
Suspending thread One
Two : 10
Two : 9
Two : 8
Two : 7
Two : 6
Resuming thread One
```

```
Suspending thread Two
One : 10
One : 9
One : 8
One : 7
One : 6
Resuming thread Two
Waiting for threads to finish
Two : 5
One : 5
Two : 4
One : 4
Two : 3
One : 3
Two : 2
One : 2
Two : 1
One : 1
Two exiting
One exiting
Main thread exiting
```

Student Activity 3

1. How will you create multiple threads?
2. Describe the function of Isalive( ) and Join( ) methods.
3. What are thread priorities? Where are they used?
4. What do you mean by synchronization?
5. What is a deadlock?
6. Name the methods that can suspend, resume or stop the execution of a thread.

13.11 SUMMARY

- Java programs create thread objects to perform concurrent tasks.
- A thread is a single line of execution within a program.
- You can extend thread class or implement runnable interface to define a class which runs in a separate thread.
- Thread is always in one of the five states : Newborn, Running, Runnable, Blocked, or Dead.
- Every thread has a priority value associated with it. Java's scheduling is preemptive; i.e., a higher priority thread will preempt the lower priority thread and can be executed immediately.
- You can use synchronized methods to serialize access to shared resources.
- Multithreading allows you for multiple paths of execution at the same time. Multiple threads running concurrently improve the utilization of resources. Using this technique you can create programs with multimedia and animation effects. Execution of each thread can be controlled by join( ), yield and sleep methods defined in thread class.

13.12 KEYWORDS

Thread : Any single path of execution.

Priority : The order in which the threads are scheduled.

Synchronization : The way to avoid data corruption caused by simultaneous access to the same data.

Deadlock : Situation when two or more threads are waiting to gain control of a resource.

13.13 REVIEW QUESTIONS

1. What is thread ? How do we start it? What are the two methods by which we may stop threads ?
2. Describe the complete life cycle of a thread.
3. What is synchronization ? When do we use it ?
4. How do we set priorities for threads ?
5. Every call to wait has a corresponding call to notify that will eventually end the waiting. True / False.
6. Declaring a method synchronized guarantees that the deadlock cannot occur. True / False.
7. A thread wants to make a second thread ineligible for execution. To achieve this, the first thread can call the yield( ) method on the second thread. True / False.
8. A thread can make a second thread ineligible for execution by calling the suspend( ) method on the second thread. True/False.
9. The sleep( ) method should be enclosed in try...catch block. True / False.
10. The yield( ) method must be enclosed in try...catch block. True / False.
11. The thread can be temporarily suspended from running by using the wait ( ) method. True / False.
12. A suspended thread using suspend( ) method can be revived using the resume( ) method. True / False.
13. When we implement the runnable interface, we must define which of the following methods?

    a. `start( )`

    b. `init( )`

    c. `run( )`

    f. `main( )`

    g. `runnable( )`

14. What will be the output of the following code ?

```
class t1 extends Thread
{
        public void run( )
        {
                System.out.prinln ("hello");
                suspend( );
                resume( );
                System.out.println ("Bye");
        }
}
```

```
class test
{
        public static void main (String args[ ])
        {
                t1 T = new t1( );
                T.start( );
        }
}
```

15. If you create a class that implements runnable, you have already written its run( ) method. Which of these start( ) methods should you use to call run method?

a.
```
public void start( )
{
        new Thread (this).start( );
}
```

b.
```
public void start( )
{
        Thread myT = new Thread( );
        myT.start( );
}
```

c.
```
public void start( )
{
        Thread myT = new Thread (this);
        myT.run( );
}
```

16. Which of the following statement is correct for the line given below?

```
Thread myT = new Thread( );
```

a. The myT is in runnable state.

b. The thread myT has the NORM_PRIORITY.

c. If myT.start( ) is called, the run method in the calling class will be executed.

d. If myT.start( ) is called, the run method in the thread class will be executed.

17. The object class has a wait method that is used to coordinate access to an object by multiple threads. Which of the following statements about the wait method are true ?

a. The wait method is an instance method of the object class.

b. The wait( ) method is a static method of the object class.

c. To call wait, a thread must have a lock on the object involved.

d. An object can have only one thread in waiting state at a time.

18. Create three classes – storage, printer and counter. The storage class should store an integer. The counter class should create a thread that starts counting from zero and stores

each value in the storage class. The printer class should create a thread that keeps reading the value in the storage class and printing it.

19. Create a program that creates an instance of the storage class and sets up a counter and a printer object to operate on it. The program is required to print 10 values only.

20. Fill in the blanks :

    a. ..... allows a running program to perform several tasks, apparently, at the same time.

    b. ..... between threads is carried out automatically by JVM.

    c. When we create a thread object, it is said to be in ..... state.

    d. A thread is said to be in ..... state when it is suspended, sleeping or waiting in order to satisfy some condition.

    e. ..... is the way to avoid data corruption caused by simultaneous access to the same data.

    f. The threads communicate by ..... for each other.

    g. ..... threads get more CPV time than ..... threads.

    h. The ..... method of the thread class is deprecated in Java 2.

Answers to Review Questions

20. (a) Multithreading (b) Switching (c) Newborn
    (d) Blocked (e) Synchronization (f) Waiting
    (g) Higher priority, lower priority (h) Suspend( )

13.14 FURTHER READINGS

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

UNIT

14

APPLETS AND INPUT OUTPUT

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe input / output basics of Java
- Describe Java Streams
- Understand how to read from and write to console.
- Describe the fundamentals of Applets
- Understand native methods and problems associated with them.

U N I T S T R U C T U R E

14.1 INTRODUCTION

This unit will deal with the objective of data handling, that is, input and output. Fundamentals of Applets and some important keywords related to it have also been discussed in the unit in addition to some important classes which help in reading and writing.

14.2 INPUT/OUTPUT BASICS

Java programs can be of both types – text based console programs and also applets that rely upon Java's Abstract Windows Toolkit (AWT) for interaction with the user. Input can be from the console or can also be from the AWT such as textfield, button etc. Output can be on the screen, file or printer.

14.3 STREAMS (BYTE AND CHARACTER)

Java program performs Input/Output through streams. A stream is an abstraction that either produces or consumes information. It is linked to a physical device by the Java Input/Output systems. Java implements streams within class hierarchies defined in java.io package.

Java defines two types of streams : byte and character. Byte streams provide a convenient means for handling input and output of bytes. They are used, for example, when reading or writing binary data. Character streams provide a convenient means for handling input and output of characters.

14.4 READING FROM AND WRITING TO CONSOLE

In Java, the only way to perform console input is to use a byte stream and an older code that uses this approach persists. The preferred method of reading console input for Java2 is to use a character oriented stream.

In Java, console input is accomplished by reading from system.in. To obtain a character based stream that is attached to the console, you wrap System.in in a Buffered Reader object to create a character stream.

Input Reader is the stream that is linked to the instance of BufferedReader that is being created. Reader is an abstract class, one of its concrete subclasses is InputStream-Reader, which converts bytes to characters. To obtain an InputStreamReader that is linked to System.in refers to an object of type InputStream it can be used for inputStream. The following line of code creates a BufferedReader that is connected to the keyboard:

```
BufferedReader br = new BufferedReader
(new InputStreamReader(System.in));
```

To read a character we use BufferedReader read( ) ® int read( ) throws IOException. Each time that read( ) is called, it reads a character from the input stream and returns it as an integer value. It returns -1 when the end of the stream is encountered.

The following program demonstrates read( ):

```
import java.io.*;
class BRead
{
        public static void main string args[ ])
        throws IOException
        {
                char C;

        BufferedReader br = new BufferedReader (new InputStreamReader
System.in));
                System.out.println ("Enter Characters, 'q' to quit");
                do
                {
                        c = (char) br read( );
                        System.out.println (c);
                }
                while (c ! = 'q')
```

```
        }
}
READING STRINGS
import java.io.*;
class BRreadlines
{
        public static void main (String args[ ])
        throws IOException
        {
                BufferedReader br = new BufferedReader (new
        InputStreamReader (System.in));
                String str;
                System.out.println ("Enter lines of text.");
                System.out.println ("Enter stop" to Quit.");
                do
                {
                        str = br.readLine( );
                        System.out.println (str)
                }
                while (! str.equals ("stop"));
        }
}
```

Writing Console Output

Console output is most easily accomplished using print( ) and println( ). These methods are defined by the class PrintStream. Even though System.out is a byte stream, using it for simple program output is still acceptable.

Because PrintStream is an output stream derived from OutputStream, it also implements the low level method write( ). Thus, write( ) can be used to write to the console. The simplest form of write( ) defined by PrintStream is:

```
void write (inbyteval) throws IOException.
// Demonstrate System.out.write( )
class WriteDemo
{
        public static void main (String args[ ])
        {
                int b;
                b = 'A';
                System.out.write (b);
                System.out.write ('\n');
        }
}
```

You will not often use write( ) to perform write because print( ) and println( ) are substantially easier to use.

14.5 READING AND WRITING FILES

Java provides a number of classes and methods that allow you to read and write files. In Java all files are byte oriented and Java provides methods to read and write bytes from and to a file. However, Java allows you to wrap a byte oriented file stream within a character based object.

Two of the most often used stream classes are FileInputStream and FileOutputStream, which create byte streams linked to files. To open a file, you simply create an object of one of these classes, specifying the name of the file as an argument to the constructor. While both classes support additional overridden constructors, the following are the forms:

```
FileInputStream (String filename) throws FileNotFoundException

FileOutputStream (String filename) throws FileNotFoundException
```

When you are done with a file you should close it by calling close( ). It is defined both in FileOutputStream and FileInputStream.

```
void cloase( ) throws IOException
```

To read from a file you can use a version of read( ) that is defined in FileInputStream.

```
int read( ) throws IOException.
```

Each time that it is called, it reads a single byte from the file and returns the byte as an integer value. read( ) returns -1 when the end of the file is encountered. It can throw an IOException.

```
/ * Display a text file
import java.io.*;
class showfile
{
        public static void main (String s[ ]) throws
                                                IOException
        {
                int i;
                FileInputStream fis;
                try
                {
                        fis = new FileInputStream (s[ ]);
                }
                catch (FileNotFoundException e)
                {
                System.out.println ("File not found" + e);
                        return;
                }
                catch (ArrayIndexOutOfBoundsException e)
                { System.out.println ("Usage : Showfile");
                return;
        }
```

```
        // read characters until EOF is encountered
        do
        {
                i = fis.read( );
                if (i ! = -1) System.out.println ((char););
        }
        while (i ! = -1);
        fis.close( );
        }
}
```

14.6 PRINTWRITER CLASS

Although using System.out to write to the console is still permissible under Java, its use is recommended mostly for debugging purposes or for simple programs. For real world programs the recommended method of writing to the console, when using Java, is through a PrintWriter Stream. PrintWriter is one of the character based classes which defines several constructors.

The one we will use is

```
PrintWriter (OutputStream os, boolean flushonnewline)
```

Here, os is an object of type OutputStream, and flushonnewline controls whether Java flushes the output Stream every time a newline ('\n') character is output. If flushonnewline is true, flushing automatically takes place. If false, flushing is not automatic.

PrintWriter supports the print( ) and println( ) methods for all types including Object. Thus, you can use these methods in the same way as you use with System.out. If an argument is not of simple type, the PrintWriter method calls the object's toString( ) method and then prints the result.

Example:

```
import java.io.*;
public class printwriterdeom
{
        public static void main (String s[ ])
        PrintWriter pw = new PrintWriter (System.out,true);
        pw.println ("This is a String");
        int i = -7;
        pw.println (i);
        double d = 4.5e-7;
        pw.println (d);
}
}
```

Student Activity 1

1. What is a stream in Java?
2. Describe various streams available in Java.
3. Describe print writer class of java.

14.7 FUNDAMENTALS OF APPLETS

Applets are small applications that are accessed from an Internet server, transported over the Internet, automatically installed and run as part of a web document. After an applet arrives on the client it has limited access to resources, so that it can produce an arbitrary multimedia user interface and run complex computations without introducing the role of viruses or breaching data integrity.

Example:

```
import   java.awt.*;
import  java.applet.  *;
public  class  simpleApplet  extends  Applet
{
        public  void  paint  (Graphics  g)
        {
                g.drawString  ("A  simple  Applet",  20,  20);
        }
}
```

The applet begins with two import statements, the first imports the Abstract Windows Toolkit (AWT) classes. Applets interact with the user through the AWT, not through the console based Input/ Output classes. The second import statement imports the applet package which contains the class Applet. Every applet you create must be a subclass of Applet. The next line in the program declares the class simpleApplet. This class must be declared as public because it will be acessed by code that is outside the program.

Inside simpleApplet, paint( ) is declared. This method is defined by the AWT and must be overriden by the applet. paint( ) is called each time the applet must redisplay its output. The situation can occur for several reasons. For example, the window in which the applet is running can be overwritten by another window and then uncovered. paint( ) method has one parameter of type Graphics. This parameter contains the graphics context which describes the graphics environment in which the applet is running.

Inside paint( ) is a call to drawString( ), which is a member of the Graphics class. This method outputs a string beginning at the specified X, Y location.

```
void  drawString  (String  msg,  intX,  intY)
```

X, and Y is the location on the screen where the drawString will display the message.

To view the output, a Java compatible web browser is needed. A small HTML file is needed where inside the applet tag the Java's class file is passed.

```
<applet code = "simpleApplet" width = 200 height = 60> </applet>
```

Then we can also use the command C:\> appletviewer app.html

14.8 TRANSIENT AND VOLATILE MODIFIER

Java defines two interesting type modifiers : transient and volatile. These modifiers are used to handle somewhat specialized situations. When an instance variable is declared as transient, then its value need, to persist when an object is stored.

For example

```
class  T
{
```

```
        transient int a; // will not persist

        int b; // will persist

}
```

Here if an object of type T is written to a persistent storage area, the contents of a would not be saved, but the contents of b would. The volatile modifier tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of your program. For efficiency considerations, the real or master copy of the variable is updated at various times, such as when a synchronized method is entered. In some cases, all that really matters is that the master copy of a variable always reflects its current state. To ensure this, simply specify copy of a volatile variable.

14.9 STRICTFP

Java2 adds a new keyword to the Java language, called strictfp. With the creation of Java2, the floating point computation model was relaxed slightly to make certain floating point computations faster for certain processors, such as Pentium.

For example, the following fragment tells Java to use the original floating model for calculations in all methods defined within Myclass.

```
strictfp class Myclass { //
```

14.10 NATIVE METHODS

Although it is rare, occasionally, you may want to call a subroutine that is written in a language other than Java. Typically, such a subroutine exists as an executable code for the CPU and environment in which you are working, that is, native code.

Java provides the native method– precede the method with the native modifier, but do not define any body for the method.

public native int meth( );

Most native methods are written in C. The mechanism used to integrate C code with a Java program is called JNI (Java Native Interface)

An example:

```
public class nativedemo
{
        int i;
        public static void main (String s[ ])
        {
                native demo nd = new nativedemo( );
                nd. i = 10;
                System.out.println ("This is ob.i before the native
                                        method" + ob.i);
                ob.test( ); // call a native method.
                System.out.println ("This is ob.i after the native
                                        method :" + ob.i);
        }
        public native void test( );
        static
        {
```

```
            System.loadlibrary  ("NativeDemo");
        }
    }
```

Notice that the test( ) method is declared as native and has no body . Also notice the static block. A static block is executed only once– when your program begins execution. The library is loaded by the loadLibrary( ) method which is a part of the System class.

Static void loadLibrary (String filename). Here filename is a string that specifies the name of the file that holds the Library. After you enter the program, compile it to produce nativedemo. Class. Next you must use javah.exe to produce one file.

```
javah  -  jni  NativeDemo
```

This command produces a header file called Nativedemo.h. This file must be included in the C file that implements test( ).

14.11 PROBLEMS WITH NATIVE METHODS

Native methods seem to offer great promises, because they enable you to gain access to your existing base of library routines. But they also introduce two significant problems:

- **Potential security risk** Because a native method executes actual machine code, it can gain access to any part of the host system. That is, native code is not confined to the Java execution environment.
- **Loss of Portability** Because the native code is contained in a DLL, it must be present on the machine that is executing the Java program. Further, each native method is CPU and operating system dependent, each DLL is inherently non portable. Thus, a Java application that uses native methods will be able to run only on a machine for which a compatible DLL has been installed.

The use of native methods should be restricted, because they render your Java programs non portable and pose significant security risks.

Student Activity 2

1. What is an applet?
2. Describe various streams available in Java.
3. Describe type modifiers available with Java.
4. What are the problems associated with native methods?

14.12 SUMMARY

- Streams, that is data passed, can be character or byte, as required.
- Data can be read from disk, file and also can be written to disk or file, shown on the screen.
- Some important classes are BufferedReader, InputStreamReader, OutputStreamWriter, FileInputStream, FileOutputStream.
- Important methods are read( ), close( ).
- Printwriter class is a class where the object of it is used as similar to system.out.
- Applets are small programs in Java which can be used to run in the browser, basically use GUI (Graphical User Interface) applications.
- Some important keywords related to Java are transient, volatile, strictfp, native methods.

14.13 KEYWORDS

Stream : An obstraction that either produces or consumes information.

Byte Stream : Provide a convenient means for handling input and output of bytes.

Character Stream : Provide a convenient means for handling input and output of characters.

Inputer Reader : The stream that is linked to the instance of Buffered Reader that is being created.

Applet : Small application that are accessed from an Internet server, transported over the Internet, automatically installed and run as part of a web document.

14.14 REVIEW QUESTIONS

1. State whether the following are true or false:
   a. There are two datatypes for handling input and output, that is, bytes and character.
   b. Two important classes used for reading and writing are FileInputStream and FileOutputStream.
   c. PrintWriter class is a class used to print the data on the printer.
   d. Applets are stand alone programs.
   e. When the value need not persist then we use volatile modifier.
2. Write something about strictfp, native and transient modifier.
3. What is an Applet? How do we run it?
4. Write an HTML code for running applet.

Answers to Review Questions

1. (a) True (b) True (c) False (d) True (e) True

14.15 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

UNIT

15

HANDLING STRINGS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Define string constructor
- Find string length
- Perform various operations on string
- Use extract character and string comparison methods
- Search / modify character / string
- Change case of characters
- Describe String Buffer

U N I T S T R U C T U R E

15.1 INTRODUCTION

String is probably the most commonly used class in Java's class library. The reason for this is that strings are a very important part of programming. Java provides two classes for handling strings. These classes exist in the java.lang package which is automatically imported in all Java classes. In this unit you will be able to create strings in the program and use different methods of String class, e.g. String length function, Substring extraction functions, String comparison functions, Uppercase and Lowercase conversion functions, conversions from primitive type to String type, Appending String etc. You will also study String Buffer class and will also be able to differentiate between String Class and String Buffer class.

15.2 STRING CONSTRUCTOR

A string is a sequence of characters. Java API provides a String class and StringBuffer class to help you manipulate strings. Literal string values in Java source code are turned into String objects by the compiler. Java implements strings as object of type String. The String class supports several constructors. To create an empty string you can call the default constructor:

```
String S = new String( );
```

To create a copy of the specified string use the following constructor:

```
String s = new String ("UPTEC");
```

To create a string initialized by an array of characters;

```
Char chars[ ] = {'a', 'b', 'c'};
String S = new String (chars)
```

This constructor initializes with the string "abc"

To create a String initialized by subrange of an array:

```
Char chars[ ] = {'a', 'b', 'c', 'd', 'e', 'f'};
String S = new String (chars, 2, 3);
```

This initializes S with the characters code.

You can create a String initialized by String object.

```
char c[ ] = {'J', 'a', 'v', 'a'};
String s1 = new String (c);
String s2 = new String (s1);
```

15.3 STRING LENGTH

To obtain number of characters that a string contains use the length( ) method.

```
char charsp[ ] = {'a', 'b', 'c'};
String s = new String (chars);
System.out.println (s.length( ));
```

15.4 OPERATIONS ON STRINGS

String literals

You can create a String instance using a String literal. For each string literal in your program, Java automatically constructs a string object.

For e.g. :

```
String s2 = "abc" / / use String Literal
```

You can use string literal at any place where you can use a String object. For e.g.

```
System.out.println ("abc".length( ));
```

String Concatenation

For convenience in working with strings, Java also uses the + and += operators to indicate concatenation. '+' operator concatenates two strings and produces a string object as the result.

```
String str1 = "UPTEC"
String str2 = "LTD"
```

```
String str3 = str1 + str2;
String S = "Four:" + 2 + 2 // display s Four:22
String S1 = "Four:" + (2+2) // display Four4
```

String Conversion and toString( )

The root of the Java object hierarchy, the Object class, has a toString( ) method that returns a descriptive string. Every class implements toString( ) because it is defined by Object. This default toString( ) method produces a rather cryptic result, so, many of the standard library classes implement toString( ) method which gives more appropriate result. For most classes that you create, you need to override the toString( ) method.

Example:

```
class Box
{
        double width;
        double height;
        double depth;
        Box (double w, double h, double d)
        {
                width = w;
                height = h;
                depth = d;
        }
public String toString( )
{
        return "Dimensions are" + width + "by" + depth + "by" +
height + ".";
}
        class toStringDemo
{
        public static void main (String args[ ])
{
        Box b = new Box (10, 12, 14);
        String S = "Box b:" + b;
        System.out.println (b);
        System.out.println (S);
        }
}
Output: Dimensions are 10 by 14 by 12
```

Box b : Dimensions are 10 by 14 by 12. toString( ) method is automatically invoked when a Box object is used in a concatenation or println.

Student Activity 1

1. What is a string?
2. Describe various constructors supported by class string.
3. How will you find the length of a string?

4. How will you concatenate two strings?

5. Describe to string( ) method of object class of Java.

15.5 EXTRACT CHARACTER METHODS

charAt( )

To extract a single character from a String.

char charAt(int where), here, where is the index of the character

```
char ch;

ch = "abc".charAt(1); // Assign "b" to ch.
```

getchars( )

To extract more than one character at a time,

```
void getchars(int sourceStart, int sourceEnd, char target[ ], int
targetStart)
```

where sourceStart specifies the index of the beginning of the subString, and sourceEnd specifies the index that is one past the end of the desired string.

```
String S = "This is a demo of the getChar method";

int start = 10;

int end = 14;

char buf [ ] = new char [end-start];

S.getchars(start, end, buf, 0);

System.out.println(buf);

Output :      demo
```

getBytes( )

Stores the characters in an array of bytes. The general form of getByte( ) is

```
byte[ ] getBytes( )
```

getBytes( ) is most useful when you are exporting a string value into an environment that does not support 16 bit Unicode.

toCharArray( )

Is used to convert all the characters in a String Object into a character array.

```
char[ ] tocharArray( )
```

15.6 STRING COMPARISON METHODS

The string class includes several methods that compare strings or substrings within strings.

equals( )

Compares two strings for equality.

```
boolean equals(Object str)
```

Here, str is the object compared with the invoking StringObject

```
String S1 = "Hello"
```

```
String S2 = "Hello"
System.out.println (S1 + "equals" + S2 + S1.equals (S2));
Output :      Hello equals Hello true
```

regionMatches( )

This method compares a specific region inside a string with another specific region in another string.

boolean regionMatches(int startindex, String str2, int str2Startindex, int numchars)

where, startindex specifies the index at which the region begins with the invoking String Object,

Str2 specifies the string being compared,

Str2Startindex specifies the index at which the comparison within Str2 starts,

numchars specifies the length of substring to be compared.

boolean regionMatches(boolean ignoreCase, int stindex, String str2, int str2stindex, int numchars)

If ignorecase is true, comparision will not be case sensitive.

equalsIgnoreCase( )

To compare two strings for equality, that ignore case difference, use the following :

boolean equalsIgnoreCase (String str) Here str is a string object compared with the invoking string

```
String s1 = "hello";
String S2 = "Hello";
System.out.println (s1 + "equals" + s2 + s1.equalsIgnoreCase);
Output :      hello equals Hello >> True.
```

equals( ) v/s = =

equals( ) method compares the characters inside a String object. The = = operator compares two object references to see whether they refer to the same instance.

Example :

```
String S1 = "Hello"
        String S2 = "Hello"
        System.out.println (S1 + "equals" + S2 + S1.equals(S2));
        System.out.prinln (S1 + "= =" +S2 + (S1 = = S2));
Output :      Hello equals Hello >> true
              Hello = = Hello >> false
```

The contents of the two string objects are identical but they have different object references.

startsWith( ) / endsWith( )

The startsWith( ) determines whether a given string begins with a specified string. The endsWith() determines whether the string in question ends with a specified string.

```
boolean startsWith (String str)
boolean endsWith (String str)
```

```
boolean startsWith (String str, int startIndex)

e.g. : "UPTEC LTD".endsWith("LTD"); // true

       "UPTEC LTD".startsWith("UPTEC"); // true
```

The second version of startsWith lets you specify a starting point.

compareTo( )

This method is used to find whether the string is less than, equal to, or greater than the next. A string is less than another if it comes before the other in dictionary order.

```
int compareTo(String str)
```

Here, str is string being compared with invoking string.

The result may be.

<0	The invoking string is less than str.
>0	The invoking string is greater than str.
=0	The two strings are equal.

compareTo( ) takes into account uppercase and lowercase letters.

int compareToIgnoreCase(String str) returns the same results as compareTo( ), except that the case differences are ignored.

Student Activity 2

1. Describe the function of the following methods :
   (a) charAt( )
   (b) getChars( )
   (c) getBytres( )
   (d) tcharArray( )
   (e) equals( )
   (f) region Matches( )
   (g) equals IgnoreCase( )
   (h) compareTo( )
2. Differentiate between the following methods :
   (a) equals( ) and ==
   (b) startWith( ) and endswith( )

15.7 SEARCHING AND MODIFYING

indexOf( )

Searches for the first occurrence of a character/string and returns the index at which the character or substring was found or -1 on failure.

```
int indexOf(int ch)

e.g. : Strings S = "UPTEC LTD";
       System.out.println("IndexOf(t)" + S.indexOf ('T'));
Output :      IndexOf (t) = 2
```

lastIndexOf( )

Searches for the last occurence of a character/string

```
int lastIndexOf(int ch)

System.out.println("lastIndexOf(t) = "+s.lastIndexOf ('T'));

Output :      lastIndexOf(t) = 7
```

substring( )

Returns substring that begins with startIndex and ends with endIndex.

```
String substring(int startIndex, int endIndex);
```

Here startIndex specifies the index at which the substring will begin and endIndex specifies the stopping point.

```
String S = "UPTEC LTD";

System.out.println(S.substring (6, 8))

Output :      LTD
```

concat( )

You can concatenate two strings.

```
String concat(String str)
```

This method creates a new object that contains the invoking string with the contents of str appended to the end.

```
e.g.:  String S1 = "One"

       String S2 = S1.concat("two");

Output :      S2 will contain "Onetwo"
```

replace( )

Replaces all occurrences of one character with other.

```
String replace(char original, char replacement)
```

Here, original specifies the character to be replaced by character specified by replacement.

```
String S = "Hello".replace ('l', 'w')

Output :      "Hewwo"
```

trim( )

Returns a copy of the invoking string from which any leading and trailing white space has been removed.

```
String trim( )

String S = "HelloWorld    ".trim( );

Output : "HelloWorld"
```

15.8 DATA CONVERSION AND VALUEOF( ) METHODS

valueOf( ) is a static method that is overloaded within String class for all built in types and also object.ValueOf converts data from other type to String.

```
static String valueOf(long num);

static String valueOf(Object ob);

static String valueOf(double num);
```

15.9 CHANGING CASE OF CHARACTERS

The method toLowerCase( ) converts all the characters in a string from uppercase to lowercase. Nonalphabetical characters, such as digits, are unaffected.

```
String toLowerCase( )

String toUpperCase( )

e.g. : String S = "This is a test";

String upper = S.toUpperCase( ); // THIS IS A TEST

String lower = S.toLowerCase( ); // this is a test
```

15.10 STRING BUFFER

String represents fixed length, immutable character sequences. StringBuffer represents growable and writeable character sequences.

```
String message = "Hello World";

message = message + "!!";
```

A StringBuffer is like the String datatype but it has methods for modifying the contents of the StringBuffer. When the Java compiler encounters code in the preceding example, it rewrites it as follows :

```
message = new StringBuffer(message).append ("!!").toString( )
```

The compiler converts the original message into a StringBuffer, appends the exclamation points to the StringBuffer, and then converts the StringBuffer back into a String.

Constructors

StringBuffer defines the following three constructors:

```
StringBuffer( ), StringBuffer(int size), StringBuffer (String str)
```

The default constructor reserves room for 16 characters. The second constructor explicitly sets the size of the buffer. Third constructor sets the initial content of StringBuffer object and reserves room for 16 more characters.

StringBuffer Class Method

length( ) returns current length of a StringBuffer.

```
e.g.: StringBuffer Sb = new StringBuffer("Hello");

      System.out.println ("Length =" + Sb.length( ));

Output :      Length = 5
```

capacity( ) method finds out total allocated characters.

```
System.out.println ("Capacity =" + Sb.capacity( ));

Output :      Capacity = 21
```

because room for 16 additional characters is automatically added.

setLength( ) method sets the length of the buffer.

```
void setLength(int len)
```

Here, len specifies length of the buffer.

```
e.g. : StringBuffer sb = new StringBuffer("Hello");

       sb.setLength(2);
```

```
        System.out.println("sb = " +sb);
Output :      sb = He
```

If you call setLength( ) with value less than the current length( ) then characters beyond the new length will be lost.

charAt( ) method obtains a single character from a StringBuffer.

```
char charAt (int)
e.g. : StringBuffer sb = new StringBuffer("Hello");
       System.out.println ("CharacterAt(1) = "+sb.charAt(1));
Output :      CharacterAt(1) = e
setCharAt( ) sets the value of a character within stringBuffer.
sb.setCharAt (1, "i");
System.out.println ("Character at(1) =" sb.charAt(1));
Output :      CharacterAt(1) = i
```

getChars( ) is used to copy substring of a String Buffer into an array.

void getChars (int sourceStart, int sourceEnd, char target[ ], int targetStart)

sourceStart	-	Index of beginning of substring.
sourceEnd	-	Index of one past the end of the desired substring.
target [ ]	-	Array which receives output.
targetStart	-	Index within target at which the substring will be copied.

append( ) method concatenates the string representation of any other type of data to the end of invoking StringBuffer object. This method is overloaded for all the built-in types and object.

```
e.g. : StringBuffer append(String str)
       StringBuffer append(int num)
```

insert( ) method one string into another. It is overloaded to accept values of all types, object.

```
StringBuffer insert(int index, String str)
StringBuffer insert(int index, char ch)
```

index specifies the index at which point the string or char will be inserted into the invoking StringBuffer object.

```
e.g. : StringBuffer sb = "I Java!";
       sb.insert (2, "like");
Output :      I like Java !
```

reverse( ) method reverses the characters within a StringBuffer object, and returns the object on which it was called

```
StringBuffer reverse( )
e.g. : StringBuffer S = new StringBuffer ("abcdef");
       s.reverse( )
       System.out.println (s);
Output :      fedcba
```

replace( ) method replaces one set of characters with another set inside a StringBuffer Object.

StringBuffer replace (int startIndex, int endIndex, String str);

The substring being replaced is specified by the indexes startIndex and endIndex.

```
e.g.:  StringBuffer sb = new StringBuffer ("This is a test");
       sb.replace (5, 7, "was");
```

Thus, the substring at startIndex through endIndex-1 is replaced.

Student Activity 3

1. Describe the function of the following methods :
   (a) IndexOf( )
   (b) Last IndesOf( )
   (c) Substring( )
   (d) Concat( )
   (e) Replace( )
   (f) Trim( )
   (g) ValueOf( )
2. What is StringBubber? Write various characters available with StringBuffer?
3. Describe various methods available with String Buffer.

15.11 SUMMARY

- A string is a sequence of characters. It can contain as many characters as you want.
- A string is a read-only object; its value cannot be changed after creation.
- The String class and StringBuffer classes are part of java.lang package which is automatically imported into all Java classes.
- Different operations can be performed on String object such as length( ), trim( ), toUpperCase( ), toLowerCase( ) etc.
- You can compare two strings using equals( ) method.
- Primitive values can be converted into string using valueOf( ) function.
- You can change the contents of existing string using StringBuffer class.
- StringBuffer provides a number of overloaded insert( ) methods for inserting various types of data at a particular location in the stringBuffer.
- Some of the methods of StringBuffer class are length( ), capacity( ), charAt( ), append( ), reverse( ) etc.
- String and StringBuffer objects deal with 16 bit unicode characters to support international alphabets.

15.12 KEYWORDS

String : A sequence of characters.

to String( ) : Method that returns a descriptive string.

CharAT( ) : Method used to extract a single character from a string.

getChars( ) : Method used to extract more than one character at a time.

getBytes( ) : Method used to store the characters in an array of bytes.

equals( ) : Method that compares two strings for equality.

equals IgnoreCase( ) : Method that compares two strings for equality, that ignore case difference.

startsWith( ) : Determines whether a given string begins with a specified string.

compareTo( ) : Method used to final whether the string is less than, equal to, or greater than the next.

concat( ) : Method used to concatenate two Strings.

replace( ) : Method used to replace all occurances of one character with other.

valueOf( ) : A static method that is overloaded within string class to convert data from other type to string.

StringBuffer : Represents growable and writeable character sequences.

15.13 REVIEW QUESTIONS

1. How many String objects are created in the following code ?

```
String x, y, z;
x = "UPTEC";
y = x;
z = x + y;
```

2. What is the output of each code ?

a.
```
String S = new String ("UPTEC");
if (s = = "UPTEC")
        System.out.println ("Equal A");
if (s.equals ("UPTEC")
        System.out.println ("Equal B");
```

b.
```
int num = 1234567;
system.out.println (String.valueOf(num).charAt(3));
```

c.
```
String s1 = "Sunday";
String s2 = "Monday";
System.out.println (s1.concat(s2).subString(4, 8));
```

d.
```
String s3 = " Monday  ";
System.out.println (s3.trim( ).indexOf("day"));
```

3. How does String class differ from the StringBuffer class?

4. Which of the following will equate to true ?

a. `s1 = = s2` (b) `s1 = s2` (c) `s3 = = s2` (d) `s1.equals(s2)` (e) `s3.equals(s1)`

5. Suppose that s1 and s2 are two strings. Which of the statements or expressions are correct?

a. `String s3 = s1 + s2;`

b. `String s3 = s1 - s2;`

c. `s1 < = s2`

d. `s1.compareTo(s2);`

e. `int m = s1.length( );`

6. Write a program to reverse a string using StringBuffer class.

7. Write a method called

   delete (String str, int m) that returns the input string with the m^{th} element removed.

8. Write a program which will count all occurences of a particular word.

9. Write a program which will rewrite the string in the alphabetical order. For example, the word "UPTEC" should be written as "CEPTU".

10. Fill in the blanks :

    a. ..... represents fixed length, immutable character sequences.

    b. ..... represents growable and writeable character sequences.

    c. ..... method replaces all occurences of one character with other.

    d. ..... and ..... methods are used for changing case of characters.

    e. To creat an empty string, ... constructor is called.

Answers to Review Questions :

1. 2 4. (d) and (e) 5. (a), (d) and (c)

10. (a) String (b) StringBuffer (c) Replace( )

    (d) to Lower Case( ), to Upper Case( ) (e) default.

15.14 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

UNIT

16

EXPLORING JAVA.LANG

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Desire wrapper classes and simple type wrappers
- Describe abstract process class
- Describe runtime class and memory management
- Describe System Class
- Understand Environment Properties
- Define Clone( ) and Clonable( ) Interface
- Define Class and Classloader
- Describe Math Class
- Describe Thread, ThreadGroup and Runnable Interface
- Describe Security Manager

U N I T S T R U C T U R E

16.1 INTRODUCTION

The java.lang package is a collection of classes which gets special treatment because some of its classes are so low level that they are considered part of the Java language. Since Java adheres to 'C' philosophy of keeping a language as simple as possible, it too relies on an external collection of methods for anything beyond the simplest data processing or flow controls. The following types of classes are contained in java.lang package.

- Type wrapper classes
- A math library class
- Low level system-access classes
- String support classes
- Multithreading support classes
- Error and exception classes

In the previous unit you studied String support classes, Multithreading classes and Exception classes. In this unit you should be able to use Type Wraper classes such as Boolean, Character, Integer etc. to convert simple primitives to object and object to primitives. You will also learn to create and execute other heavy weight processes. You will also learn to manipulate date and time objects using "System" class, "ThreadGroup" class "Math" class and its different methods.

16.2 WRAPPER CLASSES AND SIMPLE TYPE WRAPPERS

Java uses simple type, such as int, char for performance reasons. They are passed by value and cannot be passed by reference. Also there is no way for two methods to refer to same instance of int. To address this need, Java provides classes that correspond to each of the simple types. These classes are known as type Wrappers. Wrapper classes provide variety of functions related to primitives. The static and final variables of these classes provide useful constants, such as MAX_VALUE, MIN_VALUE for Integer, Byte, Character, Short, Long, Float, Double class. Another use of wrapper classes is to treat a primitive as an object. For example, Java defines a class called Vector that implements a resizable array of objects. You can not store primitive types in a vector; for this you need Wrapper class. For example, to store variable of type int in a vector, you need to create an Integer object for each int variable. Primitive data types may be converted into object types by using the wrapper classes contained in the java.lang package. For example :

Simple Type	Wrapper Class
boolean	Boolean
char	Character
double	Double
int	Integer
long	Long
float	Float

Each wrapper class provides a static method to convert a string to the corresponding primitive type.

Number

The abstract class Number defines a superclass that is implemented by the classes that wrap the numeric type bytes, short, int, long, float and double. Number has abstract methods that return the value of the object in each of the different number formats. E.g. doubleValue( ) returns the value as double.

Methods –	byte	byteValue( )
	double	doubleValue( )
	float	floatValue( )
	int	intValue( )
	long	longValue( )

Number class has six concrete subclasses that hold explicit values of each numeric type, Double, Float, Byte, Short, Integer and Long.

Double and Float

These are wrappers for floating-point values of type double and float respectively. The constructors for float are:

```
Float(double num)

Float(float num)

Float(String str) throws NumberFormatException

Double(double num)

Double(String str) throws NumberFormat Exception

e.g.: Double D = new Double (10.0);

      Float F = new Float (5.3);
```

There are number of methods defined in the Double and Float wrapper classes.

```
e.g.: int compareTo (Float f)

String toString( )

float floatValue( )

double doubleValue( )

double d = DoubleVal.doubleValue( ) // object to primitive double

float f = Floatval.floatValue( ) / / object to primitive float

int i = FloatVal.intValue( ) // Object to primitive int

float f = Float.parseFloat (STring str) // Returns the float
equivalent of the number contained in string.
```

Byte, Short, Integer and Long

These are the wrappers for byte, short, int, long integer types respectively.

Their constructors are:

Byte	(byte num)	
Byte	(string str)	throws NumberFormat Exception
Short	(short num)	
Short	(String str)	
Integer	(int num)	
Integer	(string str)	
Long	(long num)	
Long	(String str)	

Some common methods are :

parseInt( ),	parseLong( ),	toString( ),	floatValue( ),
doubleValue( ),	ValueOf( ),	intValue( ),	longValue( )

Conversion from String to Object

```
e.g.:  Integer IntVal = Integer.ValueOf (str); / / string to Integer Object
       Long LongVal = Long.ValueOf (str) / / string to Long Object
```

Conversion from Object to corresponding primitives :

```
int i = I.intValue( ) // where I is IntegerObject
float f = F.floatValue( ) // where F is Float object
```

Conversion of numeric string to Primitive Number :

```
int i = Integer.parseInt (Str)
long l = long.parseLong (Str)
```

Other Methods defined by Integer Class

String S = Integer.toBinaryString(int num)	Returns String that contains binaryequivalent of num
String S = Integer.toHexString(int num)	Returns a String that contains Hex equivalent of num
String S = Integer.toOctalString(int num)	Returns a string that contains octal equivalent of num

Other methods defined by Long class

String S = Long.toBinaryString (long num)	Returns a String that contains the binary equivalent of num
String S = Long.toHexString (long num)	Returns a String that contains the Hex equivalent of num
String S = Long.toOctalString (long num)	Returns a string that contains the octal equivalent of num

Commonly used Constants defined by Long, Double, Float, Integer, Byte, Short

MAX_VALUE	maximum positive Value
MIN_VALUE	minimum negative Value

Example:

```
import java.io.*;
class ABC
{
        public static void main (String args[ ])
        throws IOException
        {
        BufferedReader br = new BufferedReader (new Input StreamReader
                (System.in));
                String str;
                int i, sum = 0
                do
                {
                        str = br.readln( );
                        try
```

```
{
        i = Integer.parseInt (str);
}
catch (NumberFormatException   e)
{
        i = 0;
}
System.out.println ("String" + Integer.toString(i));
                System.out.println ("Binary String"
                + Integer.toBinary String(i))
                System.out.println ("Octal String" +
                Integer.toOctalString(i));
                System.out.println    ("HexString"
                + Integer.toHexString (i));
                Sum = Sum+i;
System.out.println ("Current Sum" + Sum);
                }
                while (i ! = 0);
        }
}
```

The above program gets the input. If the input is a number then it displays string equivalent of number's binary, octal and Hex equivalent. Then it accumulates the number into sum and displays current sum. If input is not a number, it raises an exception and the number is considered as zero and execution continues.

Character

Character is a simple wrapper around a char. The constructor for character is:

```
Character(char ch)
```

Here ch specifies character data. To obtain the char value contained in character object call charValue( ) as follows

```
char CharObj charValue( );
```

Some commonly used methods are:

Method	Description
static boolean isDigit(char ch)	Returns true if ch is a digit otherwise it returns false.
static boolean isLetter(char ch)	Returns true if ch is a letter otherwise it returns false.
static boolean isLowerCase(char ch)	Returns true if ch is a lowercase letter otherwise it returns false.
static boolean is SpaceChar(char ch)	Returns true if ch is space character otherwise it returns false.
static char isTitleCase(char ch)	Returns true if ch is title case character otherwise it returns false.
static char isUpperCase(char ch)	Returns true if ch is uppercase character otherwise it returns false.
char toLowerCase(char ch)	Returns lowercase equivalent of ch.
char toUpperCase(char ch)	Returns uppercase equivalent of ch.
char toTitleCase(char ch)	Returns title case equivalent of ch.

Boolean

Boolean is a wrapper class for boolean which is mostly useful when you want to pass a boolean variable by reference. It contains two constants 'TRUE' and 'FALSE' which define true and false Boolean objects.

Constructors are

- Boolean(boolean boolValue) // boolValue must either be true or false.
- Boolean(String boolString) // boolString either contains "true" or "false".

Some Common methods in Boolean Class are

```
boolean equals(Object boolObj)
```

Returns true if invoking object is equivalent to boolObj.

String toString( ) Returns the string equivalent of the invoking object.

boolean booleanValue( ) Returns boolean equivalent of Boolean object.

16.3 VOID

It is a "type Wrapper" for primitive type void. The void class has one field, TYPE, which holds a reference to the class object for type void. You do not create instance of this class.

Student Activity 1

1. What is a wrapper class?
2. List some simple type wrappers.
3. How will you convert.
   a. a string to object
   b. a numeric string to primitive number
   c. an object to corresponding primitives
4. Describe some commonly used constants.
5. Describe the following :
   a. Character
   b. Boolean
   c. Void

16.4 ABSTRACT PROCESS CLASS

Class Process provides access to stdin stdout, stderr for executing program's process; as well as return value of the program. It encapsulates a process. It is used primarily as a superclass for the type of objects created by exec( ) in Runtime class. Some common methods are

void destroy( )	Terminates the process.
int exitValue( )	Returns an exit code obtained from a subclass.
int waitfor( )	Returns the exit code returned by the process. This method does not return until the process on which it is called terminates.

16.5 RUNTIME CLASS AND MEMORY MANAGEMENT

The system and Runtime classes are designed to encapsulate the necessary connection to underlying OS. The Runtime class encapsulates the run time environment. You cannot instantiate a Runtime Object. You can get a reference to the current Runtime object by calling the static

method Runtime.getRuntime( ). Once you obtain a reference to the current Runtime object, you can call several methods that control the state and behavior of JVM.

Some methods defined in Runtime class are

- Process exec(String progName)

  Process exec (String ComlineArray [ ]);

  Executes the program specified by progName or given by Command line array.

- long freeMemory( )

  Returns the approximate number of bytes of free memory available to Java Runtime system.

- void exit(int exitcode)

  Halts the execution and returns the value of exitcode. '0' indicates normal termination, all other values indicate some form of error.

- void gc( )

  Initiates garbage collection.

- void loadLibrary(String libname)

  Loads the dynamic library whose name is associated with libraryName.

- long totalMemory( )

  Returns the total number of bytes of memory available to the program.

Although Java provides automatic garbage collection, sometimes you will want to know how large the object heap is and how much of it is left. To obtain these values, use totalMemory and freeMemory methods. Java's garbagecollector runs periodically to recycle unused objects. However, sometimes you will want to collect discarded objects prior to the collector's next round. You can run garbage collector by calling gc( ) method.

16.6 OTHER PROGRAM EXECUTION

You can use Java to execute other heavyweight processes on your multitasking operating system. The exec( ) method allows you to name the program you want to run.

```
class ExecDemo
{
        public static void main (String args[ ])
        {
                Runtime r = Runtime.getRuntime( );
                Process p = null;
                try
                {
                        p = r.exec ("notepad");
                }
                catch (Execption e)
                {
                System.out.println ("Error executing notepad");
                }
        }
}
```

16.7 SYSTEM CLASS

The system class automatically creates a standard input, standard output and standard error output stream, named system.in, system.out, and system.err. System class holds a collection of static methods and variables. Some common methods of System Class are :

- static void arraycopy (object source, int sourceStart, Object target, int targetst, int size)
- static long currentTimeMillis( )

  Returns current time in terms of milliseconds since midnight January 1,1970
- static int exit(int exitcode)

  Halts execution and returns the value of exitcode to parent process. It stops the JVM and by convention it returns nonzero status that indicates that an error has occurred.
- static void gc( )

  Initiates garbage collection.

  System.gc( ) actually calls the Runtime object's gc( ) method.
- static void runFinalization( )

  Initiates calls to the finalize( ) method of unused but not yet recycled objects.

NOTE The exit method should not be called in a Java applet. It is Webbrowser's responsibility to halt the JVM as needed.

CurrentTimeMillis( )

It returns the current time in terms of milliseconds since midnight January 1, 1970. Use this method to find out how long various parts of your program take to execute.

```
class  Elapsed
{
        public  static  void  main  (String  args[ ])
        {
                long  start,  end;
                System.out.println  ("Timing  a  for  loop  for  0  to
1000000");
                Start = System.currentTimeMillis( ); // get startingtime
                for  (int  i  =  0;  i  <  1000000;  i++);
                end  =  System.currentTimeMillis( );  //  get  endingtime
                System.out.println  ("Elapsed  time:"  +  (end-start));
        }
}
```

arraycopy( )

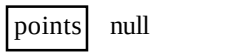
static void arraycopy(Object source, int sourceStart, Object target, int targetStart, int size)

Here the array to be copied is passed in "source" and the index at which the copy begins is passed as "sourceStart". The array "target" will receive the copy at the index passed as targetStart, size is the number of elements that are copied.

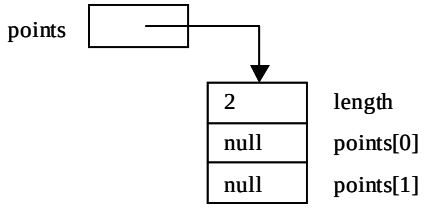
Example:

```
class point
        {
                int x; int y;
                point (int a, int b)
                {
                        x = a;
                        y = b;
                }
        }
```

MemoryModel after an array declaration, point points [ ];

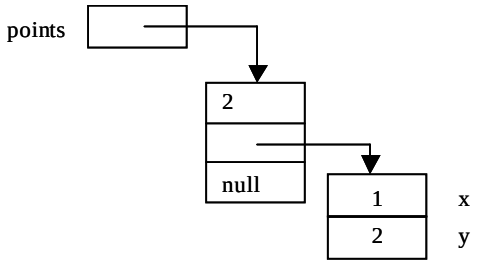


Memory Model after points = new point [2];



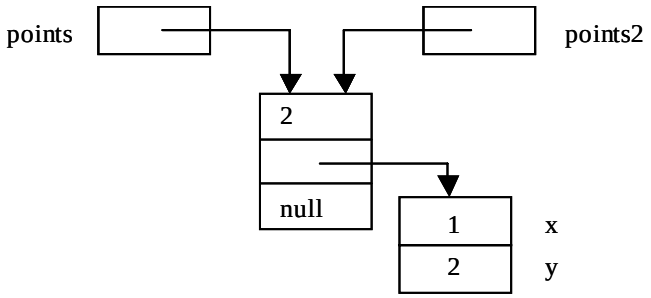
Memory Model after

points [0] = new point (1, 2)



Memory Model after

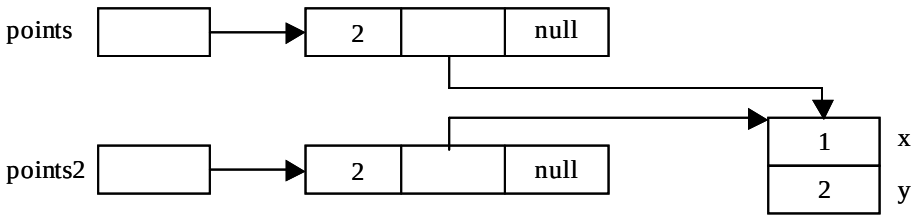
point points2[ ] = points;



To actually copy the values stored in an array into another array use -->

points2 = new point [points.length];

System.arraycopy(points, 0, points2, 0, points.length);



This method can be used to copy quickly an array of any type from one place to another. For example:

```
class ACDemo
{
        Static byte a[ ] = {65, 66, 67, 68, 69, 70};
        Static byte b[ ] = {77, 77, 77, 77, 77, 77,};
        public static void main (String args[ ])
        {
                System.out.prinln ("a=" + new String (a));
                System.out.prinln ("b=" + new String (b));
                System.arraycopy (a, 0, b, 0, a.length);
                System.out.prinln ("a=" + new String (a));
                System.out.prinln ("b=" + new String (b));
                System.arraycopy (a, 0, a, 1, a.length-1);
                System.arraycopy (b, 1, b, 0, b.length-1);
                System.out.prinln ("a=" + new String (a));
                System.out.prinln ("b=" + new String (b));
        }
}
Output :        a = ABCDEF
                b = MMMMMM
                a = ABCDEF
                b = ABCDEF
                a = AABCDE
                b = BCDEFF
```

- void setErr(Print Stream eStream)

  the method sets the standard err stream to Stream

- void setIn(InputStream iStream)

  the method sets the standard in stream to iStream

- void setOut(PrintStream oStream)

  the method sets the standard out stream to oStream

16.8 ENVIRONMENT PROPERTIES

You can obtain the value of various environment variables by calling the System.getProperty( ) method.

To display the path to the current user directory.

```
System.out.println(System.getProperty ("user.dir"));
```

Other commonly used Properties are:

java.class.path	file.separator
java.home	line.separator
java.version	os.name.
user.name	os.version
user.home	pat.separator

Student Activity 2

1. Describe some of the methods available with Abstract process class.
2. What is the function of Runtime class?
3. Describe some methods available in Runtime class.
4. What is the function of System class?
5. Describe the function of the following methods :
   a. Current Time Millis( )
   b. Arraycopy( )

16.9 USING CLONE( ) AND CLONABLE( ) INTERFACE

A clone is simply an exact copy of the original. The Clone( ) method generates a duplicate of the object on which it is called. Only classes that implement cloneable interface can be cloned. They include Vector, HashMap and BitSet class. If you call clone( ) on a class that does not implement cloneable, a CloneNotSupportedException is thrown. Cloning is potentially dangerous when object being cloned contains a reference variable. For example, if an object opens an Input/ Output stream and is then cloned, two objects will be capable of operating on the same stream. If one of these objects closes the stream, the other object might still attempt to write to it, causing an error.

For an array of objects, the value stored in array are references to the objects. To duplicate an array with its component object, elements use clone( ) method of object class in conjunction with arraycopy( ) method.

```
Vector val1[ ] = new Vector [10];

Vector val2[ ] = new Vector [val1.length];

System.arrayCopy (val1, 0, val2, 0, val1.length);

for (int i = 0; i < val1.length; i++)

val2.[i] - (Vector) val1[i].clone( );
```

For the classes which do not implement cloneable interface, you need to give the following command to fully duplicate the array:

```
points2[0] = new point(points[0].x, points[0].y);
```

16.10 CLASS AND CLASSLOADER

Class encapsulates the runtime state of an object or interface. You can not declare a Class object, they are created automatically when classes are loaded. Class methods are useful when run-time information about an object is required. Some common methods defined by Class are:

- static Class ForName(String name)

  Returns a Class object given its complete name.

- Field[ ] getFields( )

  Returns a Field object for all the public fields of this class.

- Method[ ] getMethods( )

  Returns a Method object for all the public methods of this class.

- String getName( )

  Returns the complete name of the class or interface of the invoking object.

- Class getSuperClass ( )

  Returns the superclass of the invoking object. The return value is null if invoking object is of type Object.

Example:

```
class X
{
        int a;
        int b;
}
class y extends x
{
        double c;
}
class ABC
{
        public static void main (String args[ ]
        {
                X       x = new X( );
                Y       y = new Y( );
                Class obj;
                obj = y.getClass( );
                System.out.println ('Y is of
                  type"+obj.getName( ));
                obj = obj.getSuperclass( );
                System.out.println ("Y's superclass
                    is"+obj.getsuperclass( ));
        }
}
```

Classloader

To execute Java bytecodes, the JVM uses a classloader to fetch the bytecodes from a disk or a network. A Java-enabled browser then starts up the JVM and passes the location of the applet class file to the classloader. Each classfile knows the names of any additional class files that it requires. This may require the classloader to make a number of additional classloading operations before the applet starts. If you create a subclass that extends ClassLoader, it allows you to load

classes in some way other than the way they are normally loaded by the Java run-time system. Some common methods defined by classloader are :

abstract Class loadClass(String name, boolean callResolveClass)

- An implementation of this abstract method must load a class given its name and call resolveClass( ) if callResolveClass is true.

final void resolveClass(Class obj)

- The class referred to by obj is resolved i.e., its name is entered into the class name space.

Normally the JVM loads class from the local file system in a platform dependent manner from the directory defined by classpath variable. Some classes may originate from other sources such as network. Method defineclass( ) converts array of bytes into instance of class "Class". Instance of newly defined class can be created using newInstance method.

16.11 MATH CLASS

This class contains all the floating point functions that are used for geometry and trigonometry as well as some general purpose methods.

- static double sin (double arg)

  Returns the sin of the angle specified by arg in radians.

- static double Cos (double arg)

  Returns the cosine of the angle specified by arg in radians.

- static double tan (double arg)

  Returns the tangent of the angle specified by arg in radians.

- static double exp (double arg)

  Returns e to the arg.

- static double log (double arg)

  Returns the natural logarithm of arg.

- static double pow (double y, double x)

  Returns y raised to the x.

- static double sqrt (double arg)

  Returns square root of arg.

- static int abs (int arg)

  Returns the absolute value of arg.

- static double ceil (double arg)

  Returns the smallest whole number greater than or equal to arg.

- static double floor (double arg)

  Returns the largest whole number less than or equal to arg.

- static int max (int x, int y)

  Returns the maximum of x and y.

- static int min (int x, int y)

  Returns the minimum of x and y.

- static int round (double arg)

  Returns arg rounded up to nearest int.

- static double IEEEremainder (double dividend, double divisor)

  Returns remainder of divided/divisor.

- static double random( )

  Returns a random number between 0 and 1.

- static double to Radians (double angle)

  Converts angle in degree to radians

- static double to Degrees (double angle)

  converts angle in radians to degree.

- Math defines two double constants

  E = 2.72 and PI = 3.14

java.math package also contains BigDecimal and BigInteger classes for arbitrary precision arithmetic. The BigDecimal methods are used primarly for financial calculations in which the rounding method needs to be specified exactly. BitInteger is mostly of interest to programmers doing cryptography.

16.12 THREAD, THREADGROUP AND RUNNABLE INTERFACE

These classes support multithreaded programming.

- Runnable Interface must be implemented by any class that will initiate a separate thread of execution. Runnable only defines one abstract method called run( ), which is the entry point to the thread.

```
abstract void run( )
```

- Thread class creates a new thread of execution. It has 7 constructors defined in it. e.g.,

```
Thread( )

Thread(Runnable threadOb)

Thread(ThreadGroup groupOb, Runnable threadOb, String threadName)
```

where threadOb is an instance of a class that implements the Runnable interface and defines where execution of the thread will begin. The name of the thread is specified by threadName, groupOb specifies the thread group to which the new thread belongs. Some common methods defined by thread class are :

- static int activeCount( )

  Returns the number of threads in the group.

- Thread currentThread( )

  Returns a Thread object that encapsulates the thread that calls this method.

- void destory( )

  Terminates the thread.

- Static int enumerate(Thread threads[ ])

- Puts copies of all Thread objects in the current thread's group into thread. The number of threads are returned.

- final String getName( )

  Returns thread's name

- final int getPriority( )

  Returns the thread's Priority settings.

- void run( )

  Begins execution of a thread

- final void setPriority(int priority)

  Sets the priority of the thread to that specified by priority.

Thread Group

Creates a group of threads. It defines two constructors

```
ThreadGroup(String groupaName)

ThreadGroup(ThreadGroup parentOb, String groupName)
```

In the first form a new group is created that has the current thread as its parent. In the second form the parent is specified by parentOb. Thread group offers a convenient way to manage groups of threads as a unit. This is particularly valuable in situations in which you want to suspend and resume a number of related threads. Some common methods are:

- int activeCount( )

  Returns the number of threads in the group plus any groups for which this thread is a parent.

- int activeGroupCount( )

  Returns the number of groups for which the invoking thread is a parent.

- final void destroy( )

  Destroys the thread group on which it is called.

- int enumerate (Thread group[ ])

  The threads that comprise the invoking thread group are put into the group array.

- final int getMaxPriority( )

  Returns the maximum priority setting for the group.

- final String getName( )

  Returns the name of the group.

- final void setMaxPriority (int priority)

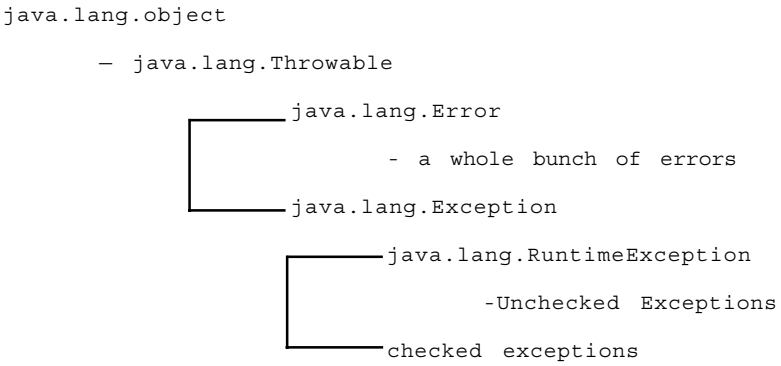
  Sets the maximum priority of the invoking group to priority.

- boolean isDestroyed( )

  Returns true if the group has been destroyed, otherwise it returns false.

16.13 THROWABLE CLASS

The Throwable Class supports Java's exception handling system and is the class from which all exception classes are derived.



A Throwable exception or error is said to be thrown at the point where it is generated and "caught" at the point where execution continues. The Throwable class provides methods that can trace the "stack" of method calls that a Thread has executed to get to the point at which the error occurred. This provides one of the main tools for debugging Java programs after an error or exception has occurred.

16.14 SECURITY MANAGER

Security Manager is an abstract class that your subclasses can implement to create a security manager. Generally you do not need to implement your own security manager. The Security manager's role is to check that individual calls to use protected resources, such as files, are acceptable at runtime. To make these decisions, the security manager has information about the calling stack frame, the actual arguments to the call and the security policy that the user has configured. The most fundamental job that the security manager must perform involves the loading of native libraries.

16.15 THE JAVA.LANG.REF AND JAVA.LANG.REFLECT PACKAGES

There are two subpackages of java.lang.

java.lang.ref

- The classes in this package provide more flexible control over the garbage collection process. For example, if you want to reuse numerous objects at some later time, you can continue to hold references to these objects, but that may require more memory.
- You can define soft references to these objects. An object that is "softly reachable" can be reclaimed by garbage collector if available memory runs low. In that case, the garbage collector sets the "soft" references to that object to null.

java.lang.reflect

- Reflection is the ability of a program to analyse itself. This package provides the ability to obtain information about the fields, constructors, methods and modifiers of a class.
- Class in the reflect package allows a program to discover the variables, methods, constructors and interfaces implemented. This capability is essential to the JavaBeans and to the Serialization API. Serialization refers to the ability to recreate the object to a file or over a network connection to another computer. It uses Reflection API to understand how to rebuild the object using object's class file. The reflect package has one interface called "Member" and seven classes. Some of the classes defined in reflect package are:

  Array, Field, Method, Constructor, Modifier etc.

Example:

```
import  java.lang.reflect.*

class  reflectdemo
```

```
public static void main (String args[ ])
{
        rc r = new rc( )
        r.disp( );
        class c = Class forName ("rc");
        Constructor c1[ ] = c.getConstructors( );
                Field f[ ] = c.getField( );
                Method m[ ] = c.getMethod( );
                for (int i = 0; i <c1.length; i++)
                {
                        System.out.println (" " + c1 [i]);
                }
                for (int i = 0; i<f.length; i++)
                {
                        System.out.println (" " + f [i]);
                }
                for (int i = 0; i < m.length;
                                        i++)
                {
                        System.out.println (" " + m [i]);
                }
        }
}
class rc
{
int a;
public rc( )
{
System.out.println ("rc constructor");
}
public void disp( )
{
        System.out.println ("disp method");
}
public void calc( )
{
        System.out.println ("calc method");
}
}
```

The above example will display all the constructors, methods and variables declared in class "rc" and class "Object" because each class is derived from super class "Object".

Student Activity 3

1. What is a clone ? Describe clone( ) method.
2. Define class and classloader.
3. Describe some methods available with math class.
4. Which classes support multithreaded programming?
5. Describe the following classes :
   a. Thread
   b. Thread Group
   c. Throwable Class
6. What is Security Manager ?
7. Describe the function of the following sub packages :
   a. java.lang.ref
   b. java.lang.reflect

16.16 SUMMARY

- The Wrapper classes let you use many Java utility classes that manipulate objects for manipulating primitive values.
- The static methods of these classes provide many convenient functions such as conversion to and from String representation.
- The static final variables of these classes provide useful constants, such as the MAX_VALUE constant of the Integer class.
- There is one wrapper class for each primitive type. For example :

  Integer wraps the int type

  Float wraps the float type
- Process class in conjunction with Runtime class is used to execute other heavy weight processes, (any executable program) such as notepad etc.
- Runtime class gives available system memory and memory usage.
- System class encapsulates stdin, stdout, stderr which allows you to write output to and read input from the console. It provides way for catastropic exit from the application through exit( ) method. It allows you to call garbage collector and initiate finalize( ) method.
- Information about current directory, home directory, login name of user etc. can be attained from getProperty( ) method of system class.
- Math Class provides functions that are used for geometric, trigonometric and arbitrary precision arithmetic.
- ThreadGroup class offers a convenient way to manage groups of thread as one unit.
- Class object provides methods to find out classname, superclass name, methods etc. for the invoking object.

16.17 KEYWORDS

java.lang package : A collection of classes which gets special treatment because some of its classes are so low level that they are considered part of the Java language.

Wrapper classes : Classes that let you use many Java utility classes that manipulate objects for manipulating prioritive values.

Number : A superclass that is implemented by the classes that wrap the numeric type bytes, short, int, long, float and double.

System class : Class that automatically creates a standard input, standard output and standard error output stream, named system.in, system.out, and system.err.

Clone : An exact copy of the original.

Clone( ) : A method that generates a duplicate of the object on which it is called.

Class loader : To execute java byte codes, the JVM uses a classloader to feteh the bytecodes from a disk or a network.

Math class : Class containing all the floating point functions that are used for geometry and trignometry as well as some general purpose methods.

Security manager : An abstract class that subclasses can implement to create a security manager.

16.18 REVIEW QUESTIONS

1. What are the applications of Wrapper Classes ?
2. Which one of these options is correct?

a.
```
double mySqrt (double val)
{
        return Math.sqrt (val);
}
```

b.
```
double mySqrt (double val)
{
        return Math.sqrt (Math.abs (val));
}
```

c.
```
double mySqrt (double val)
{
        Math myM = new Math( );
        return myM.sqrt (myM.abs (val));
}
```

3. Which of the following statements is correct ?
   a. You must initialize System with a path and filename if you want to use System.err output.
   b. The System.arraycopy method works only with array of primitives.
   c. The System.exit method takes an int primitive parameter.
4. Which of the following options will calculate the tangent of the angle ?

a.
```
double calc (double angle)
{
```

```
            return Math.tan (angle * Math.PI/180.0);
        }
b.  double calc (double angle)
        {
            return Math.tan (angle * PI/180.0);
        }
c.  double calc (double angle)
        {
            return Math.tan (angle);
        }
d.  double calc (double angle)
        {
            Math myM = new Math( );
            return myM.tan (angle * (myM.IP/180.0)).
        }
```

5. How do you get a Method object using java.lang.reflect.Method class?
   a. From any instance of the class to be examined.
   b. From a Class object created using the name of the class to be examined.
   c. From a Reflect object created using the name of the class to be examined.
6. Which of the following will produce a value of 22 if x = 22.9 ?
   a. ceil (x)
   b. round (x)
   c. abs(x)
   d. floor (x)
7. Which of the following are the Wrapper classes ?
   a. Random
   b. Byte
   c. Vector
   d. Integer
   e. Short
   f. Double
8. How does class loader load an applet ?
9. What is the use of java.ref package ?
10. Explain advantages of Thread Grouping ?
11. Write a program to input CustNo(String), ItemNo(String), Qty(int), Rate(Float class), discount Rate(Float class). Calculate NetValue and discount amount for each customer by calling a function. At the end of program display total collection and total discount amount given to all the customers.
12. Write a program to call Paintbrush program from within Java the application.
13. Write a program which does the following :

a. displays the memory available to the program.

b. displays free memory available to JavaRunTime.

c. initiates garbage collector.

d. displays free memory after initiating garbage collector.

e. Creates int array of 500 elements.

f. displays free memory after allocation.

g. discards 200 elements by putting null reference to these elements.

h. calls garbage collector.

i. displays free memory after garbage collection.

16.19 FURTHER READINGS

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT

17

JAVA.UTIL—THE UTILITY CLASSES

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe the enumeration interface
- Define vector
- Define stack
- Define dictionary
- Define hashtable
- Use store( ) and load( ) methods
- Define string tokenizer
- Describe bitset class
- Describe date and date comparison
- Describe time zones
- Describe random class
- Describe observer interface.

U N I T S T R U C T U R E

17.1 INTRODUCTION

The java.util package is a collection of classes and interfaces that support a broad range of functionality. Their applications include generating pseudorandom numbers, manipulating date and time, observing events, manipulating set of bits and tokenizing strings. Package java.util uses inheritance more than package java.lang. For example, the Properties class is an extension of class Hashtable, which itself is an extension of Dictionary. In previous unit you studied type Wrapper classes to convert simple primitives to object and object to primitivies. You also studied System, Math and Thread Group classes of java.lang package in detail. In this unit you will learn legacy classes such as Stack, Vector, Dictionary, Hashtable and Properties which are part of collection framework. You will also learn to tokenize Strings, work with dates, compute random numbers and observe events. This unit also talks about Enumeration Interface and Observable Interface.

17.2 THE ENUMERATION INTERFACE

It defines the method by which you can enumerate the elements in a collection of objects. This interface has two methods:

- boolean hasMoreElements( )
- Object nextElement( )

Enumeration interface requires any collection oriented class to implement these methods. HasMoreElements( ) returns true while there are still more elements to extract and false when all elements have been enumerated.

nextElement( ) returns the next object in the enumeration as a generic Object reference. The calling routine must cast that object into the object type held in the Enumeration. The Enumeration interface provides a way to iterate over all of the objects contained in a Vector, Hashtable or Stack. For example, if V is a Vector, to print every object contained in the vector, use the following code:

```
Enumeration e = V.elements( );
while (e.hasMoreElements( ))
{
        System.out.println (e.nextElement( ));
}
```

While an Enumeration is in use, nothing can be allowed to modify the underlying Vector data. If you need to continue using the vector, create the Enumeration from a clone of the Vector.

17.3 VECTOR

Vector implements a dynamic array, i.e., the Vector class can hold an array of object references that can automatically grow in size as needed. Because all Java classes inherit from Object, you can add any reference to a Vector. Vector contains many legacy methods. Various methods provide for retrieving objects by index and for searching. Vector extends **AbstractList** and implements **List** interface. Because Vector implements List, you can use a Vector just like you use an ArrayList instance (new collection API). You can also manipulate one using its legacy method. All of the Vector methods that alter the contents are synchronized to coordinate access by multiple threads.

Vector possesses a number of advantages over arrays.

- It is convenient to use Vectors to store objects.
- A Vector can be used to store a list of objects that may vary in size.
- We can add and delete objects from the list as and when required.

A major constraint in using vectors is that we can not directly store simple data types in a Vector; we can only store objects. Therefore, we need to convert simple types to objects using Wrapper classes.

Vector Constructors

```
Vector( ) // which has initial size of 10

Vector (int size) // initial capacity specified by size.
```

Vector (int size, int incr) // initial capacity specified by size and increment specified by incr. The increment specifies the number of elements to allocate each time that a vector is resized upwards.

```
Vector (collection c) // vector that contains the elements of
collection c.
```

All vectors start with an initial capacity. After initial capacity is reached, the next time you attempt to store an object in vector, the vector automatically allocates space for that object plus extra room for additional objects.

Vector defines three data members.

```
int capacityIncrement;          // increment value is stored

int elementCount;               // Number of elements in the vector

Object elementData[ ];          // Array that holds vector
```

Some common methods are given below.

Method	Description
final void addElement(Object elements)	Object is added to the vector.
final int capacity( )	Returns the capacity of the Vector.
final Object elementAt(int index)	Returns an enumeration of the elements in the vector.
final Object firstElement( )	Returns the first element in the vector.
final int size( )	Returns the number of elements currently in vector.
final void setSize(int size)	Sets the number of elements in the vector to size. If the new size is less than the old size, elements are lost. If new size is larger than old size null elements are added.
final int indexOf(Object element, int start)	Returns the index of the first occurrence of element at or after start.
final void insertElementsAt (Objecct	Adds element to the vector at the location specified element, int index) by index.
final void removeAllElements( )	Empties the vector.
final boolean removeElements (Object element)	Removes element from the vector. Returns true if successful and false if the object is not found.
final void removeElementAt (int index)	Removes the element at location specified by index.
final void setElementAt (object element, int index)	The location specified by index is assigned element.

Example :

```
class vectorDemo
{
public static void main (String args[ ])
        {
        Vector v = new vector (3,2);
        System.out.println ("Initial Size "+ v.size( ));
        System.out.println ("Initial Capacity "+ v.capacity( ));
        v.addElement (new Integer(1));
        v.addElement (new Integer(2));
        v.addElement (new Integer(3));
        v.addElement (new Integer(4));
        System.out.println ("Capacity after four additions:" +
v.capacity( ));
        v.addElement (new Double (5.45));
        System.out.println ("Current capacity" + v.capacity( ));
        v.addElement (new Double (6.08));
        v.addElement (new Integer (7));
        System.out.println (Current capacity" + v.capacity( ));
        v.addElement (new Float (9.4));
        v.addElement (new Integer(10));
        System.out.println ("First element" + (Integer)
v.firstElement( ));
        System.out.println("Last element"+(Integer) v.lastElement( ));
        if (v.contains (new Integer (3)))
        System.out.println ("Vector contains 3.");
        Enumeration vEnum = v.elements( );
        System.out.println ("\n Elements in vector :");
        while (vEnum.hasMoreElements( ))
        System.out.println (vEnum.nextElement( ) +" ");
        System.out.println( );
        }
}
Output :      Initial size : 0
              Initial capacity 3
Capacity after four additions : 5
Current capacity 5
Current capacity 7
First element 1
Last element 10
Vector element 3
Elements in vector
1      2      3      4      5.45   6.08   7      9.4    10
```

The above program uses a vector to store various types of numeric objects. It demonstrates several legacy methods to display various information about the vector.

Student Activity 1

1. What is java.util package ?
2. What is the role of the enumeration interface ?
3. Name the methods available in enumeration interface.
4. What is a vector ?
5. Write various vector constructors.

17.4 STACK

Stack is a subclass of vector that implements a standard last-in, first-out stack. Stack only defines the default constructor which creates an empty stack. Stack includes all the methods defined by vector and adds several of its own.

Method	Description
boolean empty( )	Returns true if the stack is empty and returns false if the stack contains elements.
Object peek( )	Returns the element on the top of the stack.
Object pop( )	Returns the element on the top of the stack and then removes it from the stack.
Object push(Object element)	Pushes element onto the stack and returns the element.
int search(Object element)	Searches for element in the stack. If found, its offset from the top of the stack is returned otherwise -1 is returned.

Example :

```
import. java.util *
        class StackDemo
        {
                static void showpush (Stack st, int a)
                {
                st.push (new Integer (a));
                System.out.println ("push ("+a +")");
                System.out.println ("Stack :" + st);
                }
        static void showpop (Stack st)
        {
        System.out.println ("pop >>");
        Integer a = (Integer) st.pop( );
        System.out.println ("stack:" + st);
        }
public static void main (String args[ ])
{
                Stack st = new Stack( );
```

```
                System.out.println ("stack" + st);
                showpush st, 42);
                showpush (st, 66);
                showpush (st, 99);
                showpop (st);
                showpop (st);
                showpop (st);
                try
                {
                        showpop (st);
                }
                catch (EmptyStackException e)
                {
                        System.out.println ("empty stack");
                }
        }
}
Output :        Stack:[ ]
                push (42)
                Stack : [42]
                push (66)
                Stack : [42, 66]
                push (99)
                Stack : [42, 66, 99]
                pop >> 99
                Stack : [42, 66]
                pop >> 66
                Stack : [42]
                pop >> 42
                Stack : [ ]
                pop >> empty stack
```

The above example creates a stack, pushes several Integer Objects onto it, and then pops them off. An EmptyStackException is thrown if you call pop( ) when the invoking stack is empty.

17.5 DICTIONARY

Arrays, vectors and stacks provide ordered access to your collection. To provide alternative collection methodology where order is not important, Java provides support for key-value pair collection. Dictionary is an abstract class that represents a key / value storage repository, given a key and a value, you can store the value in a Dictionary object. Value can be retrieved by using its key instead of looking up elements by an integer position.

Some common methods are given below.

Method		Description
Enumeration elements( )	:	Returns an enumeration of the values contained in dictionary.
Object get(Object key)	:	Returns the object that contains the value associated with key.
boolean isEmpty( )	:	Returns true if the dictionary is empty.
Object put(Object key, Object value)	:	Inserts a key and its value into the dictionary.
Object remove(Object key)	:	Removes key and its value.
int size( )	:	Returns number of entries in the dictionary.

NOTE The Dictionary class is now obsolete. You should implement the Map interface to obtain key/value storage functionality.

17.6 HASHTABLE

A Hashtable object is used extensively in Java to associate objects with names efficiently. A Hashtable object can store and retrieve "Value" objects indexed by "Key" objects. Hashtable extends Dictionary class and implements Map interface. When you use a Hashtable, you specify an object that is used as a key and the value that you want to link to that key. The key is then hashed and the resulting hashcode is used as index at which the value is stored within the table.

```
Hashtable pagetable = new Hashtable( );
String page = "http//www.coriolis.com";
String key ="coriolis";
pagetable.put (key, page);
```

A Hashtable can only store one "value" per "key" – if you add a new reference with a duplicate key, the old reference is dropped. Internally Hashtable stores object references according to hashcodes, so if you get an Enumeration of all of the stored "value" or "key" objects, the order is not predictable.

The Hashtable constructors are

- Hashtable( )
- Hashtable(int size) >> creates Hashtable with specified size.
- Hashtable(int size; float fillRatio) >> Creates Hashtable with specified size and specified fillRatio (0.0 and 1.0). FillRatio determines how full the hashtable can be before it is resized upwards. Default fill ratio is 0.75.

The operations you can do on Hashtable are mostly inherited from Dictionary.

Additional methods of Hashtable are

Method		Description
void clear( )	:	Resets and empties the hashtable.
object clone( )	:	Returns a duplicate of the invoking object.
boolean Contains (Object values)	:	Returns true if some value equal to "values" exists within the hashtable.
int size( )	:	Returns the number of entries in hashtable.
Enumeration keys( )	:	Returns the enumeration of keys contained in hashtable.
boolean isEmpty()	:	Returns true if hashtable is empty.

Example :

```
import java.util.*;
class hashdemo
        {
                public static void main (string args[ ])
                {
                Hashtable Student = new Hashtable( );
                Enumeration names;
                String str;
                float fee;
                // put data in hashtable
                Student.put ("Amit", new Float (3000.00));
                Student.put ("Jeet", new Float (2500.00));
                Student.put ("Akshay", new Float (2500.00));
                // show content of hashtable
                names = student.keys( );
                while (names. hashMoreElements( ))
{
str = (String) names.nextElements( );
System.out.println (Str +" : " + Student.get(str));
}
// deposit fine of Rs 50 into Amit's Account
fee = (( Float) Student.get ("Amit")).floatValue( );
Student.put ("Amit", new Float (fee+50));
System.out.println ("Total Money collected from Amit:" + Student.get
("Amit"));
}
}
```

17.7 PROPERTIES

When the objects to store in your hashtable are strings, you should consider using the Properties class. Properties is a subclass of hashtable. It is used to maintain lists of values in which the key is a string and value is also a string. Instead of having to cast return values to String class, the Properties class does this for you with its supporting methods. Property defines the following instance variable

```
Properties defaults;
```

This variable holds default property list associated with a properties object. Properties defines these constructors :

- Properties ( )
- Properties (Properties propDefault)

The first version is a default constructor. The second creates an object that uses propDefault for its default values. In both cases, property list is empty. In Addition to the methods that Properties inherits from Hashtable, Properties defines some more methods.

String getProperty(String key) Returns the value associated with key.

String getProperty (String key, String defaultProperty) Returns the value associated with key. "defaultProperty" is returned if key is neither in the list nor in the default property list.

void list(PrintStream streamOut) Sends the property list to the output stream linked to "streamOut".

Enumeration propertyNames( ) Returns an enumeration of the keys. This includes the keys found in the default property list too.

Object setProperty(String key, String value) Associates value with key. Returns the previous value associated with key.

void store(OutputStream streamOut, String description) After writing the string specified by description, the property list is written to the output stream linked to streamOut.

```
import  java.util.*;

class  PropDemo

{

        public  static  void  main  (String  args[ ]

        {

                properties  capitals  =  new  Properties( );

                Set  states;

                Set  str;

                capitals.put  ("M.P.",  "Bhopal");

                capitals.put  ("U.P.",  "Lucknow");

                capitals.put  ("Maharashtra",  "Mumbai");

                states  =  capitals.keySet( );

                Iterator  itr  =  states.iterator( );

                while  (itr  =  states.iterator( );

                while  (itkr.hasNext( ))

                {

                str  =  (String)  itr.next)( );

                System.out.println  ("The  capital  of"  +  str  +"is"  +
                                                capitals.  getProperty
                ((str)+  ".");

                }

                System.out.println  ();

                str  =  capitals.getProperty  ("Florida",  "Not.Found");

                System.out.println  ('The  capital  of  Florida  is"  +  str
                                                +  ".");

        }

}

Output  :       The  capital  of  M.P.  is  Bhopal

                The  capital  of  U.P.  is  Lucknow

                The  capital  of  Maharashtra  is  Mumbai

                The  capital  of  Florida  is  Not  Found
```

The above example demonstrates Properties. It creates a Property list in which the keys are the names of states and the values are the names of their capitals. Since Florida is not in the list, the defualt value is used.

Student Activity 2

1. What is a stack ?
2. Describe the function of following methods :
   a. Object peek
   b. Object( )
   c. int size( )
   d. boolean is Empty( )
3. What is a dictionary ?
4. What is a hashtable ?
5. Write various hashtable constructors.
6. Define properties.

17.8 USING STORE( ) AND LOAD( )

Information contained in the Property object can be easily stored to or loaded from disk with the store( ) and load( ) methods. This makes property lists specially convenient for implementing databases.

Example :

```
import java.io.*;
import java.util.*;
class Phonebook
{
        public static void main (String args[ ])
        throws IOException
                {
                Properties ht = new Properties( );
                BufferedReader br =
                new BufferedReader (new InputStreamReader System.in));
                String name, number;
                FileInputStream fin = null;
                boolean changed = false;
                try
                {
                fin = new FileInputStream ("Phonebook.dat");
                }
                catch (FileNotFoundException e)
                {
        }
        try
        {
```

```
                    if (fin ! = null)
                    {
                            ht.load (fin);
                            fin.close( );
                    }
            }
catch (IOException e) {System.out.println ("Enter reading file");}
do
{
System.out.println ("Enter new Name" + " ('quite toStop)":);
name = br.readLine( );
if (name.equals ("quit")) continue;
System.out.println ("Enter number:");
number = br.readLine( );
ht.put (name, number);
changed = true;
}
while (! name.equals ("quit"));
if (changed)
{
{
FileOutputStream fout = new FileOutputStream ("Phonebook.dat");
ht.store (fout, "TelephoneBook");
fout.close( );
}
do
{
name = br.readLine( );
if (name.equals ("quite")) continue;
number = (String) ht.get (name);
System.out.println (number);
}
while (! name.equals ("quit"));
}
}
```

The above program stores names and phone numbers in the Dictionary object from file called "Phonebook.dat". It also allows the user to add new details in the Dictionary object and to save the data in the file. Finally, it asks you to enter name and searches for the corresponding phone numbers.

The most common use of Properties is with a list of system properties. If you ask the system class for its list of properties, you can find out information like the

version of Java being used by the person running the program, to find out what directory to store temporary files in, just add

```
System.getProperties( ).list (System.out);
```

17.9 STRINGTOKENIZER

Class StringTokenizer extracts identifiable substrings and punctuation from an input stream according to user defined rules. This process is called tokenizing. The common application that uses text tokenizing is the compiler.

Compiler analyses a stream of tokens representing and extracted from your source file. Keywords, identifiers, punctuations, comments, strings and so on are first compressed into easy to manipulate tokens. Only after this lexical analysis stage does a compiler starts to check the complex grammar of any programming language. The Java compiler (javac.exe) uses the StringTokenizer class for this purpose.

The stringTokenizer class provides the first step in the parsing process called lexer. Parsing is the division of text into a set of discrete parts, or tokens. StringTokenizer implements the enumeration interface. StringTokenizer constructors are:

- StringTokenizer(String str) // default delimiters are used
- StringTokenizer(String str, String delimiters)
- StringTokenizer(String str, String delimiters, boolean delimAsToken)

str is the string that will be tokenized. delimiters is a string that specifies the delimiters (, ; :). If delimAsToken is true, then the delimiters are also returned. Delimiters are characters that separate tokens. The default set of delimiters consists of whitespace, space, tab, newline, carriage return.

Some common methods are:

Methods		Description
int countTokens( )	:	Returns the number of tokens left to be parsed.
boolean hasMoreElement( )	:	Returns true if one or more tokens remain in the string.
Object nextElement( )	:	Returns the next token as an Object.
String nextToken ( )	:	Returns the next token as a String.

Example :

```
import java.util.StringTokenizer;

class STDemo

{

        Static String in = "title = Java:The Complete
Reference;" + "author = Naughton and Schildt;" +

        "publisher = Osborne / McGrawHill;" + "copyright
                                        = 1999";

        public static void main (String args[ ])

        {

        StringTokenizer st = new StringTokenizer (in, "=;");

        while (st.hasMoreTokens( ))

        {

        String key = st.nextToken( );

        String val = st.nextToken( );
```

```
            System.out.println (key + "\t" + val);
            }
            }
            }
Output :    title Java : The Complete Reference
            author Naughton and Schildt
            publisher Osborne / McGraw-Hill
            copyright 1999
```

17.10 BITSET CLASS

Class BitSet implements a set of bits or a vector of boolean value. It creates a special type of array that holds bit values. This array can increase in size as needed.

BitSet operations include the following:

- Setting, clearing and getting single bits.
- ANDing, NOTing, ORing and XORing.
- Comparing bit set.

Bitset Constructors are:

- BitSet( )
- BitSet(int size)

The first version creates a default object. The second version specifies its initial size in bits. All bits are initialized to zero. BitSet implements the Cloneable interface. Some common methods are:

Method		Description
void and(BitSet bitset)	:	ANDs the contents of the invoking BitSet Object with those specified by bitset. The result is placed into the invoking object.
void clear(int index)	:	Zeros the bit specified by index.
object clone( )	:	Duplicates the invoking BitSet object.
boolean equals (object bitSet)	:	Returns true if the invoking bit set (object bitSet) is equivalent to the one passed in bitSet.
boolean get(int bitIndex)	:	Returns the current state of the bit at the specified index.
void or(BitSet bitset)	:	ORs the contents of the invoking BitSet object with that specified by bitSet. The result is placed into the invoking object.
void set(int index)	:	Sets the bit specified by the index.

Example :

```
import java.util.BitSet;
class BitSetDemo
{
public static void main (String args[ ])
{
```

```
                BitSet bits1 = new BitSet (16);
                BitSet bits2 = new BitSet (16);
                for (int i = 0; i < 16;  i++)
                {
                        if ((i%2) = = 0) bits1.set (i);
                        if ((i%5) ! = 0) bits2.set (i);
                        System.out.println (bits1);
                        System.out.println (bits2);
                        bits2.and (bits1);
                        System.out.println ("\nbit2 AND bits1");
                        System.out.println (bits2);
                        bits2.or (bits1);
                        System.out.println (bits2);
                }
        }
}
Output :        {0, 2, 4, 6, 8, 10, 12, 14}
                {1, 2, 3, 4, 6, 6, 8, 9, 11, 12, 13, 14}
                bits2 AND bits1
                {2, 4, 6, 8, 12, 14}
                bits2 OR bits1
                {0, 2, 4, 6, 8, 10, 12, 14}
```

The above program displays set bits in bit1 and bit2 in the form of their position in the bit set. Cleared bits are not shown.

Student Activity 3

1. What is the function of store( ) and load( ) methods?
2. What is a function of class string Tokenizer?
3. What is the function of Bitset class?
4. Describe the function of following methods :

   a. void clear (int index) b. object clone( )

   c. void set (int index) d. boolean get (int bitindex)

17.11 DATE AND DATE COMPARISON

The Date class encapsulates the current date and time. Date supports the following constructors:

- Date( )
- Date(long millisec)

The first constructor initialises the object with the current date and time. The second constructor sets the date and time that equals the number of milliseconds that have elapsed since midnight, January 1, 1970. Date implements the comparable interface.

Common methods are:

Method		Description
boolean after(Date date)	:	Returns true if the invoking Date object contains a date that is later than the one specified by date.
boolean before(Date date)	:	Returns true if the invoking Date object contains a date that is earlier than the one specified by date.
object clone( )	:	Duplicates the invoking Date object.
int compareTo(Date date)	:	Compares the value of the invoking object with that of date. Returns '0' if values are equal.
boolean equals(object date)	:	Returns true if the invoking Date object contains the same time and date as specified by date.
long getTime( )	:	Returns the number of milliseconds that have elapsed since January 1,1970.
void setTime (long time)	:	Sets the time and date as specified by time, which represents an elapsed time in milliseconds from midnight January 1, 1970.

Date features do not allow you to obtain the individual components of the date or time. You can only obtain the date and time in terms of milliseconds.

```
import java.util.Date;
class datedemo
{
public static void main (String args[ ])
        {
        Date date = new Date
        System.out.println (date);
        long sec = date.getTime( );
        System.out.println ("Millisecond since Jan 1, 1970" + sec);
        }
}
```

The above program displays date and time using the toString( ) method. Secondly, it displays time elapsed in milliseconds since January 1, 1970.

The abstract "Calendar" class provides a set of methods that allow you to convert a time in milliseconds to its year, month, day, hour, minute and second component.

```
import java.util.Calendar;
class Calendardemo
{
        public static void main (String args[ ])
        {
                Calendar cal = Calendar.getInstance( );
                System.out.print ("Date:");
                System.out.print (cal.get (Calendar.MONTH));
                System.out.print (cal.get (Calendar.DATE));
                System.out.println(cal.get (Calendar.YEAR));
                cal.set (Calendar.HOUR, 10);
```

```
                cal.set  (Calendar.MINUTE,  29);
                cal.set  (Calendar.SECOND,  22);
                System.out.print  ("Updated  time");
                System.out.print  (cal.get  (Calendar.HOUR)+":");
                System.out.print(cal.get  (Calendar.MINUTE)+":");
                System.out.print(cal.get  (Calendar.SECOND)+":");
            }
        }
```

The above program creates a calendar object initialized with the current date and time. It displays current month, date and year. It sets the new time and displays the updated time.

Date Comparison

There are ways to compare two Date objects.

1. Use getTime( ) to obtain number of milliseconds that have elapsed since, midnight January 1, 1970 for both objects and then compare two values.
2. Use before( ), after( ), equal( ) methods, e.g. : new Date(99, 2, 12). before(new Date (99, 2, 8) returns true.
3. Use CompareTo( ) method. Returns zero if two Date objects are equal, returns -ve value if invoking Date object is before the Date passed as parameter.

17.12 TIME ZONES

The TimeZone class allows you to work with time zone offsets from Greenwich Mean Time (GMT), also referred to as Coordinated Universal Time (UTC). It only has default constructor.

Method		Description
Static String[ ] getAvailableIDs( )	:	Returns an array of String objects representing the names of all time zones.
static TimeZone getDefault( )	:	Returns a TimeZone object that represents the default time zone used on the host computer.
abstract boolean inDaylightTime	:	Returns true if the date represented by d is in (Dated) daylight saving time in the invoking object.
static void setDefault(TimeZone tz)	:	Sets the default time zone to be used on this host. tz is a reference to the TimeZone object.

17.13 RANDOM CLASS

The Random class is a generator of pseudorandom numbers.

Random defines two constructors:

- Random( )
- Random(long seed)

The first version uses current time as seed value; in the second constructor you can supply seed value manually. If you give the same seed again, Random generates same sequence of numbers. The easiest way is to give the current time as the seed value so that it can generate different sequences. Some common methods are:

Method		Description
boolean nextBoolean( )	:	Returns the next boolean random number.
double nextDouble( )	:	Returns the next double random number.
float nextfloat( )	:	Returns the next float random number.
int nextInt( )	:	Returns the next int random number.
int nextInt(int n)	:	Returns the next int random number with the range zero to n.
void setSeed	:	Sets the seed value to that specified (long newSeed) by newSeed.
long nextLong( )	:	Returns the next long random number.

There are six tes of random numbers that you can extract from a Random object. nextBoolean( ), nextBytes( ), nextInt( ), nextLong( ), nextFloat( ), nextDouble( ) methods return a boolean, byte, integer, long, float, and double random numbers respectively.

Example :

```
import java.util.Random;
class randomdemo
{
        public static void main (String args[ ] )
        {
                Random r = new Random( );
                double d, sum = 0;
                for (int i = 0; i < 10; i++)
        {
                d = r.nextDouble( )
                System.out.println ("double Number :"+d);
                sum = sum+d;
        }
        System.out.println ("Sum = " Sum);
        }
}
```

17.14 OBSERVER INTERFACE

Sometimes, within an application, it is necessary for a change in an object to trigger changes in other objects; these changes may, in turn, trigger changes in yet other objects. In short, you have a number of objects that are in some way dependent on other objects. This is called dependency network.

In this mechanism any root object that needs to send some kind of notification to other objects should be subclassed from class Observable and any objects that need to receive such notifications should implement interface Observer.

To establish the dependency, any observer objects are added to the observable object. Whenever this observable object changes, it can then call its observable method notifyObservers( ).

The observable class is used to create subclasses that other parts of your program can observe. When an object of such a subclass undergoes a change, observing classes are notified. Observing classes implements Observer Interface, which defines update( ) method.

Common methods of Observable Class are:

Method		Description
void addObserver(Observer obj)	:	Adds obj to the list of objects observing the invoking objects.
int countObservers( )	:	Returns the number of objects observing the invoking objects.
void deleteObservers(Observer obj)	:	Removes obj from the list of objects observing the invoking objects.
void deleteObservers( )	:	Removes all observers for the invoking objects.
boolean hasChanged( )	:	Returns true if the invoking object has been modified.
void notifyObservers( )	:	Notifies all observers of the invoking object that it has changed by calling update( ).
protected void setChanged( )	:	Called when invoking object has changed.

Observer Interface defines only one method:

```
void update(Observable ObsOb, Object arg)
```

ObsOb is the object being observed

```
arg is the value passed by notifyObservers( )
```

Example :

```
import java.util.*;
class Watcher implements Observer
{
        public void update (Observable obj, Object arg)
        {
        System.out.println ("update( ) called, count is" + ((Integer)
arg)).intValue( ));
        }
}
class BeingWatched extends Observable
{
        void counter (int period)
        {
                for (; period > = 0; period - -)
                {
                        setChanged( );
                        notifyObservers (new Integer(period));
                        try
                        {
                                Thread.sleep (100);
                        }
                        catch (InterruptedException e)
                        {
```

```
                    System.out.println ("Sleep interrupted");
                        }
                    }
                }
            }
            class ObserverDemo
            {
                public static void main (String args[ ])
                {
                BeginWatched observed = new BeingWatched( );
                Watcher observing = new Watcher( );
                observed.addObserver (observing);
                observed.counter (5);
                }
            }
            Output :            Update( ) called, count is 5
                        Update( ) called, count is 4
                        Update( ) called, count is 3
                        Update( ) called, count is 2
                        Update( ) called, count is 1
                        Update( ) called, count is 0
```

The above program creates an observer class called "Watcher" that implements the observer interface. The class being monitored is called "BeingWatched". It extends Observable. counter( ) method in "BeingWatched" counts down from a specific value. Each time the count changes, notifyObservers( ) is called with the current count passed as its argument. This causes the update( ) method inside "Watcher" to be called which displays the current count. In the main( ), a Watcher and a BeingWatched object, called observing and observed, respectively are created. Then observing is added to the list of observers. Each time when counter( ) calls notifyObservers( ), observing.update( ) will be called.

Student Activity 4

1. What is the function of Date Class?
2. Write a program to display date and time of the system.
3. How will you compare two dates.
4. What is Random Class ? Write its constructors.
5. List some methods of observable class.
6. What is the function of observable class?

17.15 SUMMARY

In this lesson you should have learned the following:

- The primary, historic collection classes that provide ordered access are Vector and Stack.
- The Interface Enumeration offers the means to step through the contents.
- If you need to store primitive datatypes within a Vector, or any other collection besides an array, you need to use the Wrapper classes found in Java.
- The abstract Dictionary class describes the interface for working with key value maps.

- BitSet implements a set of bits or a vector of boolean values. Bitset operations include setting, clearing and getting single bits, Anding, NOTing, ORing, and XORing bitsets together.
- Date class can be used to store and manipulate dates. Calendar class provides a set of methods, that allow you to convert a time in millisecond to its year, month, day, hour, minute and second components.
- The Random class generates pseudo random numbers between 0 and 1.
- The Observable class is used to create subclasses that other parts of your program can observe. When an object of such a subclass undergoes a change, observing classes are notified. The update method is called when an observer is notified of a change in an observed object. For example, events generated in GUI are handled by delegating to objects that have registered an interest in that type of event. This model of event handling is called observer-observable design pattern in which the observable is the component that generates events and the observer is the object that has registered to receive events. Event observer objects are called listeners.

17.16 KEYWORDS

java.util package : A collection of classes and interfaces that support generating pseudorandom numbers, manipulating date and time, observing events, manipulating set of bits and tokenizing strings.

Vector : Vector implements a dynamic array.

Stack : A subclass of vector that implements a standard last-in, first-out stack.

Dictionary : An abstract class that represents a key / value storage repository.

Hashtable : Used to associate objects with names efficiently. It can store and retrieve “value” object indexed by “key” objects.

Properties : A subclass of hashtable used to maintain lists of values in which the key is a string and value is also a string.

Tokenizing : Extracting identifiable substrings and punctuation from an input stream according to user defined rules.

Random Class : A generator of pseudorandom numbers.

Observable Class : A class used to create subclasses that other parts of a program can observe.

17.17 REVIEW QUESTIONS

1. What is a vector ? How is it different from an array ?
2. What are legacy classes ?
3. What is the difference between Hashtable and String classes ?
4. What are the usages of Observer Interface ?
5. Which of the following statements about the java.util.Vector and java.util.Hashtable classes are correct ?
   a. A vector can hold object references or primitive values.
   b. A vector maintains object references in the order they were added.
   c. A Hashtable requires string objects as keys.
   d. A Hashtable maintains object references in the order they were added.
6. An object of the Hashtable class can store and retrieve object references based on associated "key" objects. Which of the following interfaces does Hashtable implement ?
   a. SortedMap b. Map

7. Write a program to create an array to store 12 months of a year. Using Stack, display these months in reverse order.

8. Write a program to store command line arguments in a vector object. Change the content of second element. Insert new element at a specific location. Delete an item at specific location. Convert vector into array of strings and display the Strings.

9. Write a program to input name and birthdate of a person. Find out his age.

17.18 FURTHER READINGS

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

UNIT

18

INPUT OUTPUT CLASSES

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe input output classes
- Describe file class in Java
- Describe a directory
- Describe the stream classes
- Describe input stream and output stream
- Describe file input stream and file output stream
- Describe byte array input stream and byte array output stream
- Describe print stream
- Describe random access file
- Describe stream tokenizer
- Understand stream benefits.

U N I T S T R U C T U R E

18.1 INTRODUCTION

A file is an abstraction used to represent any form of text or data in a permanent form. So far we have used variables and arrays for storing data inside programs. In this approach, firstly, the data is lost either when a variable goes out of scope or when the program is terminated. Secondly, it is difficult to handle large volumes of data using variables and arrays. We can overcome this problem by storing data on secondary storage devices such as floppy or hard disks. In this lesson, you will learn how to process data present in files in your java program. Java has a fairly clean interface to deal with the files. Files are just one more way in which your Java program obtains input or writes output. Other examples of external entities your program can deal with are console, network connection, MemoryBuffer. These can be manipulated by the Java Input/ Output classes. Although physically different, these devices are all handled by the same abstraction stream. Different files have different names and they are often stored in directories. This unit also discusses the aspect of file input and output.

18.12 INPUT OUTPUT CLASSES

To support the input output operations, the package used is java.io. The data retrieved from an input source are the results and are sent to an output destination. All devices are handled by same abstraction : the stream. A stream is a logical entity that either produces or consumes information. A stream is linked to a physical device by the Java I/O system. All streams behave in the same manner even if the actual devices that they link to may differ. Some of the important classes are:

- BufferedInputStream
- DataInputStream
- DataOutputStream
- FilterInputStream
- FileReader
- PrintWriter
- OutputStream
- StreamTokenizer
- File

18.3 FILE IN JAVA

File class in Java does not specify how information is retrieved from or stored in files. The File class is used to represent the name of a file or a set of files (a directory). A File object is used to obtain or manipulate the information associated with a disk file such as the permission, time, date and directory path. You can also get some more attributes for the file such as whether the file exists, whether the file is writeable or not and so on. The constructors used to create File objects are:

- File (String path of the Directory)

```
File f = new File ("/") ; // f refers to root directory
```

- File (String dirpath, String filename)

```
File f = new File ("/", "etc./passwd"); // f refers to file "/etc/
passwd"

File f = new File ("config.sys"); // f refers to config.sys in the
current directory
```

- File (File dirobj, String filename)

```
File f1 = new File (f, "a.dat"); // f refers to a.dat in the root
directory
```

Some common methods used in File are:

Method	Description
boolean isFile( )	Returns true if the invoking object is a file. This method returns false for some special files such as device drivers and pipes.
boolean isAbsolute( )	Returns true if the file has an absolute path and false if path is relative.
boolean renameTo (File newname)	It returns true if filename specified by "newname" becomes the new name of invoking File object and false if file cannot be renamed.
boolean delete( )	Returns true if it deletes the file and false if the file cannot be removed. It can also be used to delete directory provided the directory is empty.
String getName( )	Returns name of the file without path.
String getpath( )	Returns full path including filename.
String getParent( )	Returns full path of parent directory.
boolean exists( )	Returns true if file exists and false if file does not exist on the disk.
boolean canWrite( )	Returns true if file is writable and false if it is not writable.
boolean canRead( )	Returns true if file is readable and false if it is not readable.
boolean isDirectory( )	Returns true if the invoking object is a directory and false if it is a file.
int length( )	Returns file size in bytes.
boolean isHidden	Returns true if file is hidden and false if it is not hidden.
long lastModified	Returns time stamp of invoking file in millisec elapsed since Jan 1, 1970.
boolean setLastModified (long sec)	Sets the time stamp of invoking object to that specified by "sec".
boolean setReadOnly( )	Sets the invoking file to read only.
void deleteOnExit( )	Removes the file associated with the invoking object when the Java Virtual Machine terminates.

18.4 DIRECTORY

A directory is a file that contains a list of other files and directories. The method to check for directory is isDirectory( ). It returns true if invoking file is a directory. You can call list method on file object to extract the list of other files and directories inside the invoking file object, provided it is a directory.

```
Syntax :    String[ ] list( )
```

Example :

```
import java.io.File;
class Dircheck
{
        public static void main (String s[ ])
        {
                String dirname = "UPTEC";
                File fil = new File (dirname);
                if (fil.isDirectory( ))
                {
                        System.out.println ("Directory of" + dirname);
                        String s[ ] = fil. list( );
                        for (int i = 0; i < s. length;i++)
                        {
                        File file = new File (dirname + "/" + s[i]);
                                if (file.isDirectory( ))
                        {
                        System.out.println (s[i] + "is a directory")
                                }
                                else
                                {
                                System.out.println (s[i] +" is a file");
                                                }
                                        }
                                }
                                else
                                {
                        System.out.println (dirname + is not a
directory");
                                }
                        }
                }
```

18.5 FILE NAME FILTER INTERFACE

To limit the number of files returned by the list ( ) method we need to use the second form of list ( ).

```
String [ ] list (FilenameFilter fobj)
```

In this form, FileNameFilter defines only a single method accept( ).

```
boolean accept (File dir, String filename)
```

A sample code:

```
import java.io. *
public class try implements FilenameFilter
```

```
{
        String s;
        public try (String s)
        {
                this.s = "." + s;
        }
        public boolean accept (File dir, String name)
        {
                return name. endsWith (s);
        }
}
class dirfilter
{
        public static void main (String args[ ])
        {
                String dname = "/java/bin";
                FilenameFilter only = new try ("class");
                String s[ ] = f1.list (only);
                for (int i = 0; i <s.length; i++)
                System.out.println (s[i]);
        }
}
```

18.6 CREATING DIRECTORY

Two useful File utility methods are mkdir( ) and mkdirs( ). Both methods return either true or false. To create directory for which no path exists, use mkdirs( ) which creates directory and Parent directories.

Example:

```
File f = new file ("abc");
f.mkdir( );
```

Student Activity 1

1. What is a file in Java ?
2. What is a stream ?
3. List some methods used in file.
4. What is a directory ?
5. How can we limit the number of files returned by the list( ) method ?
6. How will you create a directory with no path ?

18.7 THE STREAM CLASSES

Java's input output streams are built upon four abstract classes:

- InputStream
- OutputStream
- Reader
- Writer

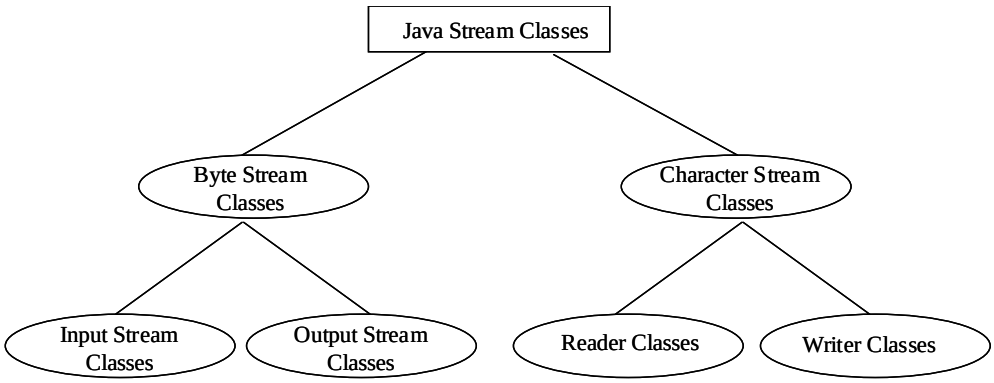


Figure 18.1 Classification of Java Stream Classes

InputStream and OutputStream are designed for byteStreams. You can use ByteStream classes when working with bytes or other binary objects. InputStream classes that are used to read 8 bit bytes include a super class known as InputStream and a number of subclasses for supporting various input related functions. InputStream class is an abstract class and we cannot create instance of this class.

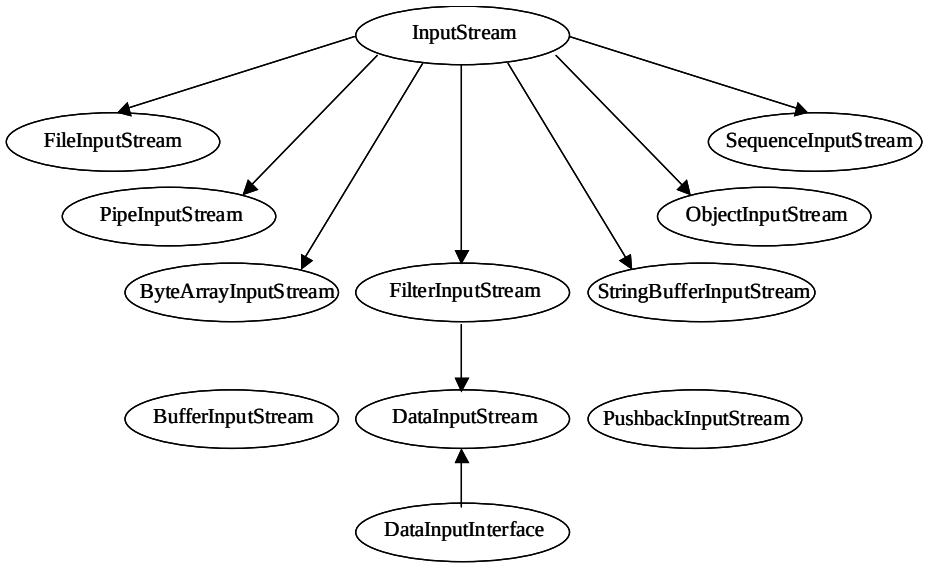


Figure 18.2 Hierarchy of InputStream Classes

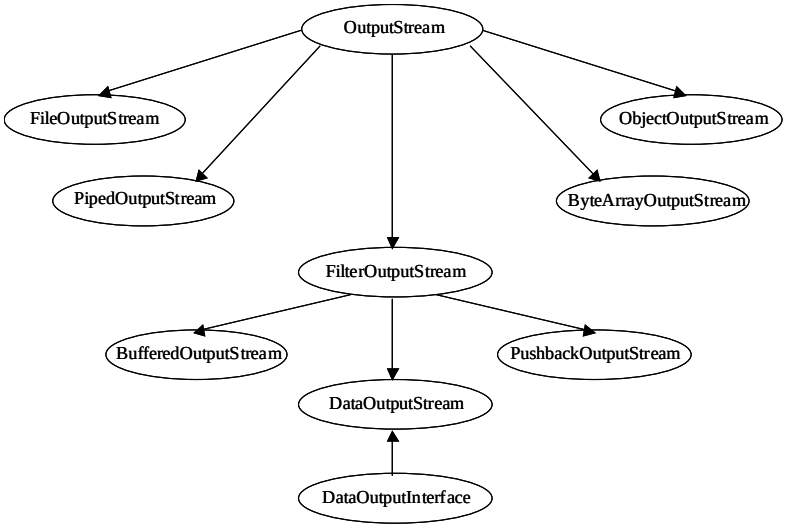


Figure 18.3 Hierarchy of OutputStream Classes

Reader and Writer are designed for Character Streams: You should use character streams when working with characters or strings. Basic unit of data for Reader and Writer stream is Unicode character.

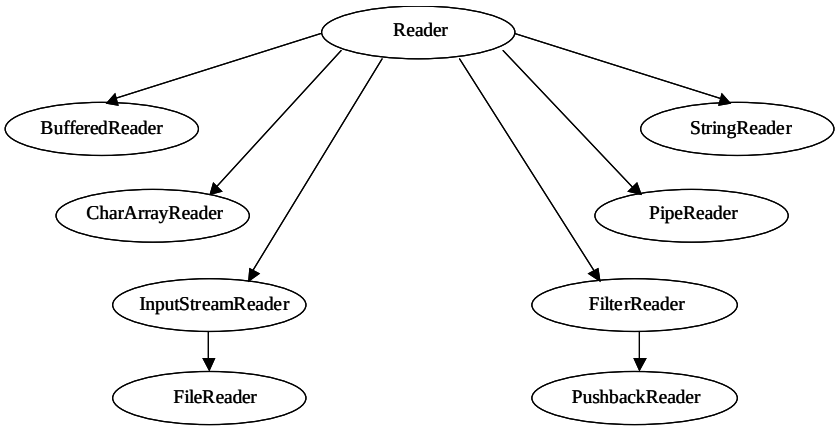


Figure 18.4 Hierarchy of ReaderStream Classes

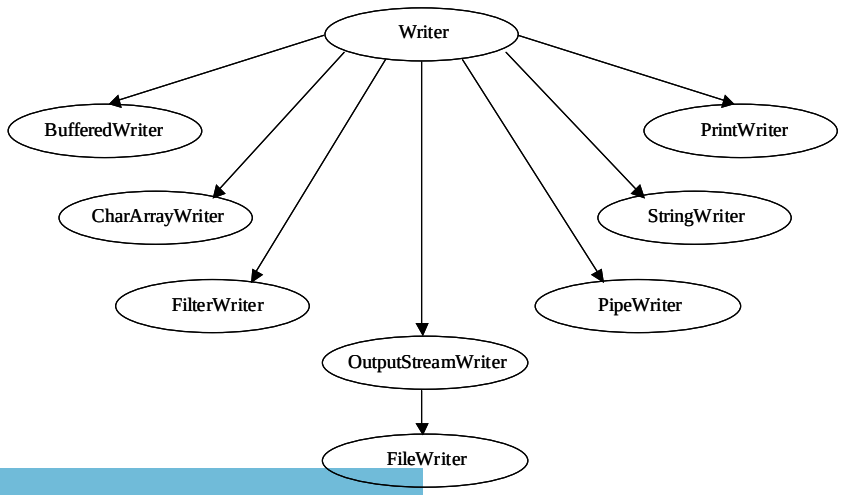


Figure 18.5 Hierarchy of WriterStream Classes

18.8 INPUTSTREAM AND OUTPUTSTREAM

It is an abstract class that defines Java's model of streaming byte input. All these methods throw an IOException on error conditions. Some important methods are:

InputStreamClassMethod	Description
int available( )	Gives number of bytes available in input.
void Close( )	Closes the input stream.
void mark(int numbytes)	Places a mark at the current position that will remain valid until numbytes are read.
boolean marksupported( )	Returns true if mark( )/reset( ) are supported.
int read( )	Returns int representation of next byte.
int read(Byte buffer[ ])	Reads buffer.length bytes into buffer. Returns -1 at the end of file.
int read(byte buffer[ ],	Reads numbytes into buffer starting from offset. Returns
int offset, int number)	-1 at the end of file.
void reset( ),	Resets the input pointer to previously set mark.
long skip(long numbytes)	Skips numbytes of input.

OutputStream Class is an abstract class that defines streaming byte output. Methods are:

OutputStream ClassMethod	Description
void close( )	Closes output stream.
void flush( )	It flushes/clears the output buffer.
void write(int b)	Writes a single byte to an output stream.
void write(byte buffer[ ])	Writes a complete array of bytes to an output stream.
void write(byte buffer [ ],	Writes numbytes bytes from array buffer, beginning at int
offset, int numbytes)	buffer [offset].

18.9 FILEINPUTSTREAM AND FILEOUTPUTSTREAM

FileInputStream This class creates an InputStream that can be used to read bytes from a file. The constructor used, takes a file name of File object and opens the file for reading.

- FileInputStream(String filepath)
- FileInputStream(File fileobj)

It throws FileNotFoundException.

FileOutputStreams This class creates an OutputStream that can be used to write bytes to a file. The constructors open an existing file or create a new file for writing data.

- FileOutputStream(String Filepath)
- FileOutputSteam(File Fileobj)
- FileOutputStream(String filepath, boolean append)

FileInputStream overrides six of the methods in abstract class InputStream. The mark( ) and reset( ) are not overriden.

FileOutputStream will create the file before opening it for output when you create the object. In case you attempt to open a read only file, an IOException will be thrown.

Note that the class DataInputStream extends FileInputStream implements the interface DataInput. Therefore, the DataInputStream class implements the methods and described in DataInput in addition to using the methods of InputStream class. The DataInput interface contains following methods to read different data types:

- readShort( ), readDouble( ), readInt( ), readLong( ),
- readLine( ), readLong( ), readChar( ), readFloat( ),
- readBoolean( ), readUTF( )

Similarly, DataOutputStream implements the interface DataOutput and therefore, implements the following methods contained in DataOutput interface:

```
WriteShort( ), WriteChar( ),
WriteFloat( ), WriteLong( ),
WriteDouble( ), WriteInt( ), WriteBoolean( ),
WriteUTF( ).
```

Example: FileInputStream and FileOutputStream

```
import java.io.*;
class fileio
{
        public static void main (String args[ ]) throws Exception
        {
        int size;
        InputStream f = new FileInputStream ("a.dat");
        OutputStream f0 = new FileOutputStream ("b.dat");
        OutputStream f1 = new FileOutputStream ("b1.dat");
        OutputStream f2 = new FileOutputStream ("b2.dat");
        size = f.available( );
        int n = size/4;
        for (int i = 0; i < n; i++)
                {
                        System.out.print ((char)f.read( ));
                }
                byte b[ ] = new byte [n];
                f.read(b);
                System.out.println (new String (b, 0, n));
                f.skip (n);
                f.read (b, n/2, n/2);
                System.out.println (new String (b, 0, b.length) );
                f.close( );
```

```
                String s = "some data . . . . ";
                byte buff[ ] = s.getBytes( );
                for (int i = 0; i < bufflength; i = i+1)
                {
                        f0.write (buff[i]);
                }
        f1.write (buff);
        f2.write (buff, bufflength-bufflength/4, buf.length/4);
                f2.close( );
                f1.close( );
                f0.close( )
        }
}
```

Student Activity 2

1. Describe various abstract classes upon which Java's input output streams are built.
2. Describe various methods available in input stream.
3. Describe various methods available in output stream.
4. Describe the role of file input stream.
5. Describe the role of file output stream.

18.10 BYTE ARRAY INPUT STREAM AND BYTE ARRAY OUTPUT STREAM

It is an implementation of an input stream that uses a byte array as the source. Constructors are:

```
ByteArrayInputStream(byte array[ ])
ByteArrayInputStream(byte arr[ ], int start, in Numbyte)
```

Example: ByteArrayInputStream and ByteArrayOutputStream

```
import java.io.*;
class ByteArrio
{
        public static void main (String args[ ])) throws IOException
        {
                String s = "abcdefg";
                byte b[ ] = s.getBytes( );
                ByteArrayInputStream in1 = new
                        ByteArrayInputStream(b);
                ByteArrayInputStream in2 = new
                ByteArrayInputStream (b, 0, 3);
                for (int i = 0; i < 2; i++)
                {
                        int c;
                        while (( c = in1.read( )) ! = -1)
```

```
                {
                        if (i = = 0)
                        {
                        System.out.print ((char) c);
                        }
                        else
                        {
                        System.out.print       (character.
                              toUpperCase (char)c));
                        }
                }
                int.reset( )
        }
        ByteArrayOutputStream       f       =       new
                ByteArrayOutputStreams( );
        f.write (b); // buffer to stream;
        System.out.println (f.toString( ));
        byte buff[ ] = f.toByteArray( );
        for (int i = 0; i <buff.length; i++)
        {
                System.out.println ((char) b[i]);
        }
        OutputStream f2 = new FileOutputStream ("a.dat");
        f.writeTo (f2); // to an output Stream
        f2.close( );
        f.reset( );
        for (int i = 0; i < 3; i++)
        f.write ('X');
        System.out.println (f.toString( ));
    }
}
```

ByteArrayOutputStream

It is an implementation of an output stream that uses a byte array as the destination. ByteArrayOutput Stream writer writes to memory. It should be used only when the amount of data to be written can safely be accomodated in memory as writing only a few megabytes will cause problems.

```
ByteArrayOutputStream( )
ByteArrayOutputStream (int numBytes)
```

This class provides for some extra methods above minimal write( ) method defined by OutputStream.

```
void writeTo(OutputStream out) throws IOException.
```

18.11 FILTEREDBYTESTREAM

void set (int index)Filtered streams are simply wrappers around underlying input or output streams and provide extended level of functionality such as buffering, character translation. The classes are FilterInputStream and FilterOutputStream.

The constructors are:

```
FilterOutputStream(OutputStreams OS)

FilterInputStream(InputStream is)
```

The classes provided in these classes are identical to those in InputStream and OutputStream.

18.12 BUFFEREDBYTESTREAM

For the byte oriented streams, a buffered stream extends a filtered stream class by attaching a memory buffer to the Input/Output streams. This allows Java to do Input/Output operations on more than a byte at a time, hence, increasing performance. You can use skipping, marking, resetting of the stream.

```
BufferedInputStream has two constructors:

BufferedInputStream (InputStream is) // creates buffered stream
                              using a default buffer size.

BufferedInputStream (InputStream, int buffsize) // creates buffered
                              stream with specified size.
```

BufferedInputStream allows you to wrap any input stream into buffered stream. In Buffer Output Stream the constructors used are:

```
BufferedOutputStream(OutputStream os) // default 512 bytes

BufferedOutputStream(OutputStream os, int Buffsize)
```

BufferedOutputStream is similar to any output stream with the exception of an added flush( ) method that is used to ensure that data buffers are physically written to the actual output device. Since the BufferedOutputStream is used to improve performance by reducing number of times the system actually writes data, you may need to call flush( ) to cause any data that is in the buffer to get written.

18.13 PRINTSTREAM

The PrintStream class provides all of the formatting capabilities which were used from the system file handle System.out.

System.out stream is a PrintStream. All outputs are in the form of 8 bit bytes. A PrintStream throws IOExceptions and sets an internal flag when one occurs. This class does not correctly handle unicode characters, hence programmers are encouraged to use the PrintWriter class instead of PrintStream. Java's PrintStream objects support the print( ), println( ) methods for all types including object. If an argument is not of a simple type, the PrintStream methods will call the objects' toString( ) method and then print the result.

PrintStream has two constructors:

```
PrintStream(OutputStream OutStream);

PrintStream(OutputStream OutputStream; boolean flushOnnewLine);
```

If FlushOnnewLine is true, flushing automatically takes place every time when (\n) character is output.

18.14 RANDOMACCESSFILE

Files are normally read and written sequentially. Sometimes you need the ability to go to a specific location in the file. RandomAccess class encapsulates a random access file. It is not derived from

InputStream or OutputStream. It implements DataInput or DataOutput which define the basic Input/Output methods. It supports positioning file pointer within the file. The constructors are:

```
RandomAccessFile(File fobj, String access) throws IOException.

RandomAccessFile(String filename, String access) throws IOException
```

In the first form, fobj specifies the name of the file to open as File object. In the second form, file is passed as file name. Access modes are "r" (the file can be read) and "rw" (the file can be opened in readwrite mode).

A RandomAccessFile has a file pointer that determines where in the file the next read or write operation will occur. When the file is initially opened, this pointer is at 0 (the beginning of the file). After that, it is positioned where last read or write ended. You can position this pointer with the seek method and write data at that position (Existing data is "pushed" to make space). A position is specified in bytes from beginning of file.

```
void seek(long newpos) throws IOException.
```

In addition to methods for reading an arbitrary number of bytes into a buffer, the RandomAccessFile class implements all the methods called by the DataInput and DataOutput interfaces. This means that there are methods for reading and writing all Java primitives plus String objects. In addition, one new method is added.

Void setLength(long len) throws IOException. This method sets the length of the invoking file to that specified by len. This method is used to lengthen or shorten a file.

Example:

```
public class randomfile
{
        public static void main (String args [ ])
        throws IOException
        {
        RandomAccessFile rf = new RandomAccessFile ("abc", "rw")
                for (int i = 0, i <10; i++)
                rf.writeDouble (i * 3.14);
                rf.close( );
                rf.new RandomAccessFile ("abc", "rw");
                rf.seek (5 *8);
                rf.writeDouble (2.71);
                rf.close( );
        }
}
```

18.15 STREAMTOKENIZER

Tokenizing some input means reducing it to a simpler stream of tokens. A stream can contain three types of entities:

- Words (multicharacter tokens)
- Single-Character tokens
- Whitespace including Java comments

Before you start processing a stream into tokens, you must define which ASCII characters should be treated as one of the three possible input types. You can get next token by calling nextToken( )

method of StreamTokenizer class. This method returns type of token it produced. It can either be multicharacter TT_WORD token or a single character token in which case return value holds the ASCII code for that character. If Stream is exhausted, nextToken( ) returns TT_EOF. If you have enabled end-of-line checking by invoking eolIsSignificant (true), TT_EOL will be returned each time the end of line is reached. TT_number will be returned each time numbers are encountered in the stream. Before starting to call nextToken( ), you should set up syntax table for the input stream using wordChars( ) and whitespaceChars( ) methods.

StreamTokenizer breaks up the InputStream into tokens that are limited by a set of characters.

Constructor

StreamTokenizer(Reader inStream)

Methods

- void eolIsSignificant(boolean eolflag) true the end of line is returned as tokens false the end of line char is ignored.
- void wordchars(int start, int end) specifies the range of characters that can be used in words.

  void range is 33 to 255
- void whitespaceChars(int start, int end)

  start-end is the range of whitespace characters.
- nextToxen( ) obtains next token from input string and returns type of token.
- type is the public variable returned by nextToken( ) and can keep following types.

TT_WORD	:	If token is a word
TT_NUMBER	:	If token is a character
TT_EOL	:	End of line encountered. this assumes that eolIs Significant( ) was invoked with 'true' flag
TT_EOF	:	If end of stream is encountered

Word Count with Stream Tokenizer

```
import java.io.*;
class Wrd
{
        public static int w = 0;
        public static int l = 0;
        public static int c = 0;
        public static void wc (Reader r) throws IOException
        {
        StreamTokenizer tok = new   StreamTokenizer(r);
                tok.resetSyntax( ).
                tok.wordChars (35, 255);
                tok.whitespaceChars (0, ' ');
                tok.eolIsSignificant (true);
                while (tok.nextToken( ) ! = tok.TT_EOF)
```

```
                {
                        switch (tok.ttype)
                        {
                                case tok.TT_EOL:
                                i++;
                                c++;
                                break;
                                case tok.TT_WORD;
                                w++;
                                default :
                                c + = tok.val.length( );
                                break;
                        }
        } }
        public static void main (String args[ ])
        {
                try
                {
                        wc (new FileReader ("a.dat"));
                        System.out.println ("a.dat :" + "lines
                        =" + l + "words = "+w + "chars = "+c);
                }
                catch (IOException c)
                {
                        System.out.println ("error");
                }
}}
```

18.16 STREAM BENEFITS

Streaming interface to Input/Output in Java provides a clean abstraction for a complex and often cumbersome task. Filtered stream classes allow you to dynamically build the custom streaming interface to suit your data transfer requirements.

Student Activity 3

1. What is the role of ByteArray Input Stream ?
2. What is the role of ByteArray Output Stream ?
3. What is the role of FilteredByte Stream ?
4. What is the role of BufferedByte Stream ?
5. Describe PrintStream Class.
6. What is the function of stream Tokenizer ?

18.17 SUMMARY

- All devices in Java are handled by same abstraction called stream. Stream is a logical entity which either produces or consumes information.
- You can send output to consol output through System.out.println. Here, "out" is an instance variable in System class. "out" is object of PrintStream class. The object is kept in memory so that you can refer to any of its method such as println( ).
- Standard input is represented by System.in. Here, "in" is an object of InputStream class. "in" does not enable you to read one line at a time or to read a java primitive. When you want to read an entire line from the console you must wrap System.in with InputStreamReader and BufferedReader.
- The file class in java represents tha name of the file, directory or set of files. A file object is used to retrieve different file attributes.
- InputStream and OutputStream classes are designed for byte streams.
- Reader and Writer classes are designed for character streams (Unicode characters).
- Methods in the stream class throw IOException.
- FileInputStream and FileOutputStream classes are used to handle disk files.
- to read a data file Use FileOutputStream. Wrap it with BufferedInputStream then with DataInputStream.
- RandomAccessFile class enables you to position the file pointer at any location in the file and allows reading and writing.

18.18 KEYWORDS

File : An abstraction used to represent any form of text or data in a permanent form.

Directory : A file that contains a list of other files and directories.

Input Stream : An abstract class that defines Java's model of streaming byte input.

FileInputStream : The class which creates an InputStream that can be used to read bytes from a file.

FileOutputStream : The class which creates an OutputStream that can be used to write bytes to a file.

PrintStream : The class that provides all of the formatting capabilities which were used from the system file handle system.out.

Random Access File Class : Enables you to position the file pointer at any location in the file and allows reading and writing.

18.19 REVIEW QUESTIONS

1. What is a file ? Why do we require files to store data ?
2. What is a stream ? What are the benefits of stream ?
3. Distinguish between
   a. InputStream and Reader Classes
   b. OutputStream and Writer Classes
4. Describe the function of File class.
5. Why do we need random access files ?

6. How will you check whether a file to be opened for writing already exists?
7. How is double type value read from the keyboard ?
8. The programmer must explicitly create the System.in and System.out objects. True / False.
9. If the file pointer points to a location in a sequential file other than the beginning, we must use the seek method to bring the pointer to the beginning, to read from the beginning of the file again. True / False.
10. To delete a file, we can use an instance of class File. True / False.
11. Reader class has a method that can read and return floats and doubles. True / False.
12. Unicode characters are all 16 bits. True / False.
13. File class can be used to rename a file. True / False.
14. DataOuputStream objects are used to write primitive data to a file. True / False.
15. DataInput is
    a. an abstract class
    b. used to read primitive datatype.
    c. interface that defines method to open files.
    d. interface that defines method to read primitive data types.
16. Which of the following statements are valid?
    a. new DataInputStream( );
    b. new DataInputStream (new File ("in.dat"));
    c. new DataInputStream ("in.dat");
    d. new DataInputStream (new FileInputStream ("in.dat")):
17. Which exception is thrown by read( ) method?
    a. Exception
    b. FileNotFoundException
    c. ReadException
    d. IOException
18. Given f is a File Object, which of the following are legal statements to create a new file?
    a. file.create( )
    b. new FileOUtputStream (file);
    c. new FileWriter (file);
    d. new FileInputStream (file);
    e. new RandomAccessFile (file);
19. Given below is the code for creating File object myFile. What happens when the given code is run ?

```
File createFile (String name)
{
File myFile = new File (name);
return myfile;
}
```

a. A new empty file with that name is created but not opened.

b. The current directory changes to specified name.

c. A new empty file with that name is created and opened.

d. None of the above

20. Which of the following are not abstract classes or interfaces ?

a. InputStream

b. PrintStream

c. Reader

d. DataInput

21. Write a program to echo every line entered by the user, unless the line is an empty line (in which case, the program terminates).

22. Write a program to copy the text file called "UPTEC.DAT" to "BACKUP.DAT". Also write the same program using the FileInputStream and FileOutputStream.

23. Write a program to create a datafile called temperature.dat. User enters temperature of the day after every few seconds and the data will be noted down in the data file. Also find out the highest temperature of the day.

24. Write a program to display all the subdirectories present on the harddisk.

25. Write a program to count number of characters, number of words and lines in the file.

26. Write a program to create a random access file that can store details of about five employees. Details include empno, name, address, salary. Calculate total salary given to all employees. Also display alternate records.

18.20 FURTHER READINGS

I.a dhotre, *Operating System*, Technical Publications.

Lang holx kandel Gideon, Abraham Kandel Joe L. Mott, *Foundation of Digital logic Design*, World Scientific.

UNIT

19

NETWORKING

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe the basics of networking
- Describe proxy server
- Describe domain name services
- Describe networking classes and interfaces
- Describe InetAddress class
- Describe TCP / IP sockets
- Describe datagram pocket
- Describe networth.

U N I T S T R U C T U R E

19.1 INTRODUCTION

This unit explores the java.net package which provides support for networking. Its creators have called it java "programming language that makes it more appropriate for writing networked programs than, say C++ or FORTRAN". What makes java a good language for networking are the classes defined in the java.net package.

19.2 BASICS OF NETWORKING

When two or more computers want to communicate with each other to share data or resources then the communications among them can be done through networking.

Overview of Socket

A Socket is a handle to a communication link with another application over the network. In other words a socket is a software interface that connects an application to the network. Sockets are often used in client/server applications. A centralised service waits for various remote machines to request specific resources, handling each request as it arrives. In order that clients know how to communicate with the server, standard application protocols are assigned well known ports.

Overview of Client/Server

You might be aware of the term Client/Server mentioned in the context of networking. A client is an entity that relies on another entity to accomplish some task. A server is an entity whose sole purpose is to serve clients by providing them with some kind of well-defined services such as searching a database or accepting mail messages. The interaction between client and server is just like the interaction between a lamp and an electrical socket. The power grid of the house is the server and the lamp is the power client. The server is a permanently available resource whereas the client is free to unplug after it has been served.

The notion of a socket allows a single computer to serve many different clients at once, as well as serving many different types of information. This is managed by the introduction of a port, which is a numbered socket on a particular machine. A server process can listen to a port number, although each session is unique. A session process must be multithreaded to manage multiple client connections.

Reserved Sockets

TCP/IP has 65,536 different TCP ports (or sockets) to which every machine can talk. The following table defines the standard TCP ports:

Port Name	Port Number	Description
echo	7	Displays whatever is sent by the user
discard	9	Discards whatever is sent by the user.
daytime	13	Gives the local time of the destination machine.
qotd	17	Give the "quote of the day" for the machine.
changer	19	Generates a test stream of characters.
ftp	21	FTP Port.
telnet	23	Telnet Protocol Port.
smtp	25	SMTP Port.
finger	79	Finger Protocol Port.
http	80	Web Server Port.
pop3	110	Pop Version 3 Port.
nntp	119	NNTP Port.

19.3 PROXY SERVER

A proxy server speaks the client side of a protocol to another server. This is often required when clients have certain restrictions on which servers they can connect to. Thus a client would connect to a proxy server which did not have such restrictions and the proxy server would in turn communicate for the client. A proxy server has the additional ability to filter certain requests or cache the results of those requests for future use. A caching proxy HTTP server can help reduce the bandwidth demands on a local network's connection to the Internet.

19.4 DOMAIN NAMING SERVICES

The Internet would'nt be a very friendly place to navigate if everyone had to refer to their addresses as numbers. Just as the four numbers of an IP address describe a network hierarchy from left to right, the name of an Internet address, called its domain name, describes a machine's location in a name space from right to left. For example, www.yahoo.com is in the com domain (commercial sites), it is called Yahoo (company name) and www is the name of the specific computer. www corresponds to rightmost number in the equivalent IP address.

19.5 NETWORKING CLASSES AND INTERFACES

The classes contained in the java.net package are listed here

Authenticator (java2)	JavaURLConnection (java2)	
ContentHandler	MultiCastSocket	URL
DatagramPacket	NetPermission	URLClassLoader
DatagramSocket	PasswordAuthentication	URLConnection
DatagramSocketImpl	ServerSocket	URLDecoder
HttpURLConnection	Socket	URLEncoder
InetAddress	SocketImpl	ContentHandlerFactory
	SocketPermission	
SocketImplFactory	URLStreamHandlerFactory	FileNameMap
SocketOptions		

Student Activity 1

1. What is a socket ?
2. Give an overview of client / server networking.
3. What is a Proxy Server ?
4. List some classes contained in java.net package.

19.6 INETADDRESS CLASS

As you know, when you want to estabish a connection between two machines across the Internet or any other kind of network, addresses are fundamental. The InetAddress class is used to convert the Domain Name (or textual Internet address) of an Internet address into an object which represents the address. The InetAddress class doesn't have any constructors but it has three factory methods that can be used to create instances of the InetAddress class.

NOTE According to Object Oriented Program, the term factory applies to a class that can construct instances of that class without invoking constructors. The instance of the class are constructed through static methods built into the class.

Methods (Factory/Instance) of InetAddress Class

The following are the factory methods of InetAddress Class.

static InetAddress getLocalHost( ) throws UnknownHostException.

static InetAddressgetByName (String hostname) throws UnknownHostException

static InetAddress [ ] getAllByName (String hostname)

thrown UnknownHostException

The getLocalHost( ) method returns the InetAddress object that represents the local host. The getByName (String hostname) method takes the hostname as a parameter and returns the InetAddress object.

The getAllByName (String hostname) method returns an array of InetAddress objects that includes all the addresses that a particular name (passed as a parameter to the method) resolves to.

Program 1:

```
import  java.net.*;
class  uptecdemoInetAddress1
{
        public  static  void  main  (String  args[ ])
        throws  UnknownHostException
        {
                InetAddress  idr;
                idr  =  InetAddress.getLocalHost( );
                System.out.println  (idr);
        }
}
```

The output produced by the program will be different from the one given below :

```
tr  7comph/132.147.168.17
```

Program 2

The following program displays the name and address of the machine whose host name is given by the user.

```
import  java.io.*;
import  java.net.*;
class  DemoInetAddress2
{
        public  static  void  main  (String  args[ ])
        throws  UnknownHostException,  IOException
        {
                InetAddress  adr;
                String  host;
                BufferedReader  str  =  new  BufferedReader
                (new  InputStreamReader  (System.in));
                System.out.print  ("Enter  the  Machines  |  host  name:");
                host  =  str.readLine( );
                adr  =  InetAddress.getByName  (host);
                System.out.println  (adr);
        }
}
```

NOTE BufferedReader and InputStreamReader are defined in the java.io package. The BufferedReader class includes methods which allow you to read text lines from a character InputStream. Since the standard input (System.in) is a byte stream, the Input StreamReader class converts the byte stream of the standard input, System.in, into a character stream. The method readLine( ) allows you to read text lines from an input stream.

The output produced by this program is as follows (the output will vary according to the input):

Enter the Machine's host name : Machine1 Machine1/130. 130. 15. 24

Enter the Machine's host name : Uptec

Uptec / 130. 130. 15. 25

InetAddress class also defines some non static methods which can be used on objects returned by the factory methods.

Methods	Purpose
String getHostName( )	Returns the string representation of the host name associated with the InetAddressObject.
byte[ ] getAddress( )	Returns an array of four bytes representing the numeric address of the host. This array of bytes cannot be printed directly by caching them to int. They have to be copied into an array of integers while ANDing with 255 to undo sign-extending.
String toString( )	Returns a string that consists of the host name and the IP Address, For example Uptec/130.130.15.25. This function can be overridden.
boolean equals(/InetAddress addr)	Returns true if the invoking object has the same Internet address as addr.

Let us see an example which demonstrates the usage of the getAddress( ) method and overriding of the toString( ) method to display the Internet address.

Program 4

```
import  java.net.*;
class  Uptec
{
        static  InetAddress  adr;
        byte[ ]  numaddr;
        byte[ ]  numadr;
        String  Addrmsg;
        Uptec( )  throws  UnknownHostException
        {
                adr  =  InetAddress.getLocalHost( );
                numadr  =  adr.getAddress( );
                Addrmsg  =  adr.getHostName( )  +"/";
        }
        public  String  toString( )
```

```
        {
                for (int i = 0; i < numadr.length; i++)
                Addrmsg + = (Numadr[i] & 255) +".";
                return "The local machine|'s address is :" + Addrmsg;
        }
}
```

19.7 TCP/IP SOCKETS

TCP/IP sockets are used to implement reliable, bidirectional, persistent, point to point, stream based between two hosts on the Internet. A socket can be used to connect Java's Input/Output system to other programs that may reside either on the local machine or any other machine on the Internet.

In Java, TCP sockets are classified as:

- TCP Server Socket
- TCP Client Socket

The java.net package provides two classes that allow you to create the two kinds of sockets. The classes are Socket and ServerSocket respectively. The Socket class, basically used to create client sockets, is designed to connect to server sockets and initiate protocol exchanges. The ServerSocket class, basically used to create server sockets, is designed to be a listener, which waits for clients to connect before doing anything.

When you create a Socket object, it implicitly establishes the connection between the client and the server. The Socket class provides some constructors to create client sockets.

Constructor	Description
Socket (String hostname, int port) throws UnknownHostExecption, IOException	This constructor is used to create a client socket and connect the local host to the specified hostName and port. It can throw an UnknownHostException or an IOException.
Socket (InetAddress ipAddress, int port) throws IOException	This constructor is used to create a client socket using a pre existing InetAddress object and a port. It can throw an IOException.

After you have created a socket, you can get the address and port details associated with the socket by the use of the following methods:

Method	Description
InetAddress getInetAddress ( )	Returns the address associated with the socket instance.
int getPort( )	Returns the remote port to which this socket is connected.
int getLocalPort( )	Returns the local port to which this socket instance is connected.

Since TCP is a stream based protocol, every TCP socket is associated with input and output streams.

NOTE Remember there should be two ports for all TCP connections, a port on the remote machine and a port on the local machine, through which the client communicates. The local port number need not be specified because the TCP/IP software allocates these ports dynamically. These ports are called ephemeral ports because they exist only for the duration of a volatile client/server transaction.

As discussed earlier, Java defines another class that must be used to create server applications. The ServerSocket class is used to create servers that listen for either local or remote client programs to connect to them on standard ports.

Constructor	Description
ServerSocket (Int port) throws IOException	This constructor is used to create a server socket on the specified port with a default queue length of 50.
ServerSocket (int port, int queue length) throws IOException	This constructor is used to create a server socket on the specified port with the specified queue length. It can throw an IOException.

Writing Client/Server Systems using TCP

To write a Server program:

Step 1 : Create the server socket and begin listening.

Setp 2 : Call the accept( ) method to get new connections.

Setp 3 : Create input and output streams for the returned socket.

Step 4 : Conduct the conversation based on the agreed protocol.

Step 5 : Close the Client Stream and the Socket.

Step 6 : Go back to step 2 or continue to step 7.

Step 7 : Close the Server Socket.

Program 1

For using TCP for a server program which sends the current data to the client upon its request and terminates when the client wants to quit.

```
import java.io.*;
import java.net.*;
import java.util.*;
class TCPServer
{
        public static void main (String args[ ])
        {
                ServerSocket server;
                Socket recSocket;
                PrintWriter on;
                BufferedReader i;
                String msg;
                server = new ServerSocket (8001);
                for (; ;)
```

```
                {
                        recSocket = Server.accept( );
                        i = new BufferedReader (new InputStreamReader
                                (recSocket.getInputStream( )));
                        on = new PrintWriter (new OutputStreamWriter
                                (recSocket.getOutputStream( )));
                        on.println (server);
                        on flush( );
                        msg = i.readline( );
                        if (msg.equals ("Date"))
                        msg = "Client asked for "+msg+" and the current
                               date is" + (new Date( )).toString( );
                        else
                        if (msg.equals ("Quit"))
                        System.exit(1);
                        else
                        msg = "Invalid Request";
                        on.println (msg);
                        on.flush( );
                        i.close( );
                        on.close( );
                        recSocket.close( );
                }
        }
}
```

To write a Client Program:

Setp 1 : Create the Client Socket Connection.

Step 2 : Acquire read and write streams for the socket.

Step 3 : Use the streams according to the server's protocol.

Step 4 : Close the streams.

Step 5 : Close the socket.

```
import java.io.*
import java.net.*;
class TCPClient
{
        public static void main (String args[ ]) throws
        IOException
        {
                Socket socket;
                BufferedReader in;
                PrintWriter out;
```

```
            String request;

            socket = new Socket ("Vandana", 8001);

            out = new PrintWriter (new OutputStreamWriter
                        (Socket.getInputStream( )));

            System.out.println ("Server says :"+ in.readLine( ));

            DataInputStream i = new DataInputStream (System.in);

            System.out.print ("Client can Request for Date or
                        quit :");

            request = i.readLine( );

            out.println (request);

            out.flush( );

            System.out.println ("server responds:" +
                        in.readLine));

            in.close( ).

            out.close( );

            socket.close( );

        }

    }
```

19.8 DATAGRAM PACKET

The java.net package provides two classes namely, DatagramPacket and DatagramSocket through which you can communicate with remote systems. A UDP Datagram is encapsulated in an instance of the class DatagramPacket.

The DatagramPacket object is the container for the data while the DatagramSocket is used to send or receive the datagram packets.

The class DatagramPacket defines two constructors which allow you to create datagram packet. The first constructor is used for receiving data over a datagram socket.

```
DatagramPacket (byte buffer[ ], int size)
```

The second constructor is used for transmitting datagrams. This constructor requires the destination machine's address and port number from the buffer and the size parameters.

DatagramPacket (byte buffer, int size, InetAddress destination, int port)

Other methods defined in the class DatagramPacket are

Method	Description
InetAddress getAddress( )	Returns the destination machine's address.
int getPort( )	Returns the portnumber.
byte [ ] getData( )	Returns data contained in the datagram.
int getLength( )	Returns the length of the valid data contained in the byte array returned from the getData( ) method.

DatagramPacket class does not provide the mechanism of sending and receiving datagrams. The companion class DatagramSocket provides this functionality.

Constructor	Description
DatagramSocket( ) throws SocketException	This constructor is used to create a socket at an unused port generally used for client applications.
DatagramSocket (int port) throws SocketException	This constructor is used to specify a particular port which is useful for server applications.
DatagramSocket (int port, InetAddress addr) throws SocketException	This constructor is used for machines having multiple IP addresses. You can use this constructor send and listen for datagrams from one of the IP addresses assigned to the machine.

The important methods are:

Method	Description
void send (DatagramPacket packet) throws IOException	This method sends the datagram's data to the previously defined host and port by taking a DatagramPacket object as a parameter.
void receive (DatagramPacket p) throw IOException	This method receives a datagram by taking a DatagramPacket object as a parameter. To extract data from the received datagram use the getData( ) method defined in the DatagramPacket Class.
void close ( )	This method closes the opened datagram socket.

Server and Client

Program 1

This program displays the local time at the specified web site where the web site's name is typed at the command line.

```
import java.net.*;
import jave.io.*;
class daytimedemo
{
        final static int DAY_TIME_PORT = 13;
        public static void main (String args[ ])
        {
                DatagramSocket socket;
                InetAddress Destination;
                DatagramPacket datagram;
                byte dataSend[ ] = new byte [256];
                String dataReceive;
                if (args.length = = 0)
                {
                        System.out.println ("Usage: Java Day time
                                                Demo");
                        System.exit (0);
                }
```

```
                try
                {
                        socket = new DatagramSocket( );
                        Destination = InetAddress.getByName (args[0]);
                        datagram = new DatagramPacket
                                    (datasend, datasend.Length,
                                    destination, DAY_TIME_PORT);
                        socket.send (datagram);
                        datagram = new DatagramPacket
                                    (datasend, datasend.length);
                        socket.receive (datagram);
                        datareceive = new String(datagram.getData( ));
                        System.out.println (The time at the
                        site"+destination.getHostName( ) + "is" +
                                    dataReceive);
                        Socket.close( );
                }
                catch (UnknownHostException h)
                {
                        System.out.println ("The website is unknown");
                }
                catch (IOException I)
                {
                        System.out.println ("Error in sending
                                            or receiving datagram");
                }
                catch (socketException s){System.out.println
                            ("Socket cannot be created");}
        }
}
```

19.9 NETWORTH

Given the five classes, InetAddress, Socket, ServerSocket, DatagramSocket and DatagramPacket, you can program any Internet protocol in existence. They also give you powerful low level control over Internet connections.

Student Activity 2

1. Describe InetAddress class.
2. Describe factory methods of InetAddress class.
3. What is the use of TCP/IP sockets ?
4. Describe some constructors provided by socket class.
5. How can you get the address and port details associated with the socket ?
6. Write an algorithm to write a server program.
7. Write an algorithm to write a client program.
8. What is a datagram pocket ?

19.10 SUMMARY

- Java is the first programming language developed with built in support for network programming. This is possible with the help of the classes defined in the java.net package.
- A socket is a handle to a communication link over the network, with another application. In other words, a socket is a software interface that connects an application to the network. Sockets are often used in client/server applications.
- A socket allows a single computer to serve many clients at once, as well as serving many different types of information.
- A server process can listen to a port only when a client connects to it. A server can accept multiple clients connected to the same port number.
- TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of communication protocols for communication between different types of machines and networks.
- TCP is a connection oriented data stream service. A connection is required between two computers before sending data via TCP.
- The InetAddress class is used to convert the DomainName (textual Internet address) of an Internet address into an object which represents that address.
- The java.net package provides two classes, namely DatagramPacket and DatagramSocket, through which you can communicate with remote systems.
- The java.net package provides two classes that allow you to create two kinds of TCP Sockets. The classes are Socket and ServerSocket.

19.11 KEYWORDS

Socket : A software interface that connects an application to the network.

Port : A numbered socket on a particular machine.

Client : An entity that relies on another entity to accomplish some task.

Server : An entity whose sole purpose is to serve by clients providing them with some kind of well-defined services such as searching a database or accepting mail messages.

Domain name : The name off an Internet address that describes a machine's location in a name space from right to left.

InetAddress Class : A class used to convert the Domain name of an Internet address into an object which represents the address.

Datagram Packet : A class used to communicate with remote systems.

19.12 REVIEW QUESTIONS

1. Fill in the blanks:
   a. Java provides in built support for network programming through classes defined in the _____ package.
   b. A _____ is a software interface that connects applications over a network.
   c. _____ is a numeric addressing scheme, while _____is a textual addressing method.
   d. _____ is the networking protocol supported by Java.
   e. _____ class is used to convert the domain name of an Internet address into an object, which represents that address.
2. State whether the following are true or false.
   a. A server can accept multiple clients connected to the same port.

b. A factory method is another name for a constructor and is used to create instances of a class.

c. The DatagramPacket class uses factory methods to create datagram instances.

d. As soon as a socket object is created, it implicitly establishes a connection between the server and the client.

3. Write a program to demonstrate the usage of the getAddress ( ) method and overriding the toString( ) method.

4. Write a program to show a datagram client. It reads user input strings and sends them to the echo server (already made), the echo server sends the data right back and the client prints the response to the console.

5. Write a program to display the local time at the specified web site where the web site's name is typed at the command line.

Answers to Review Questions :

1. (a) java.net (b) Socket (c) IP Address, Domain name system
   (d) TCP / IP (e) Inet Address

2. (a) False (b) True (c) True (d) True

19.13 FURTHER READINGS

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

UNIT

20

APPLET CLASS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe applet basics
- Understand applet life cycle
- Understand a simple banner applets
- Understand how to handle events.
- Describe applet context interface
- Describe audioclip and applet stub interface.

U N I T S T R U C T U R E

20.1 INTRODUCTION

As you have learned, an applet is a small program that runs embedded in a web browser's HTML page. Applet has a drawing or working area. When it starts running, it can immediately use its applet area. In the previous unit, you were introduced to general applets and steps necessary to compile and run one. In this unit we will learn about applets in detail.

20.2 APPLET BASICS

Traditionally, the word applet has come to mean any small application. In Java, an applet is any Java program that is launched from a web document, that is from an HTML file. They can be transported over the network from one computer to another. The size or complexity of a Java applet has no limit. Java has revolutionized the way Internet users retrieve and use documents on world wide network. Java has enabled them to create and use fully interactive multimedia web documents.

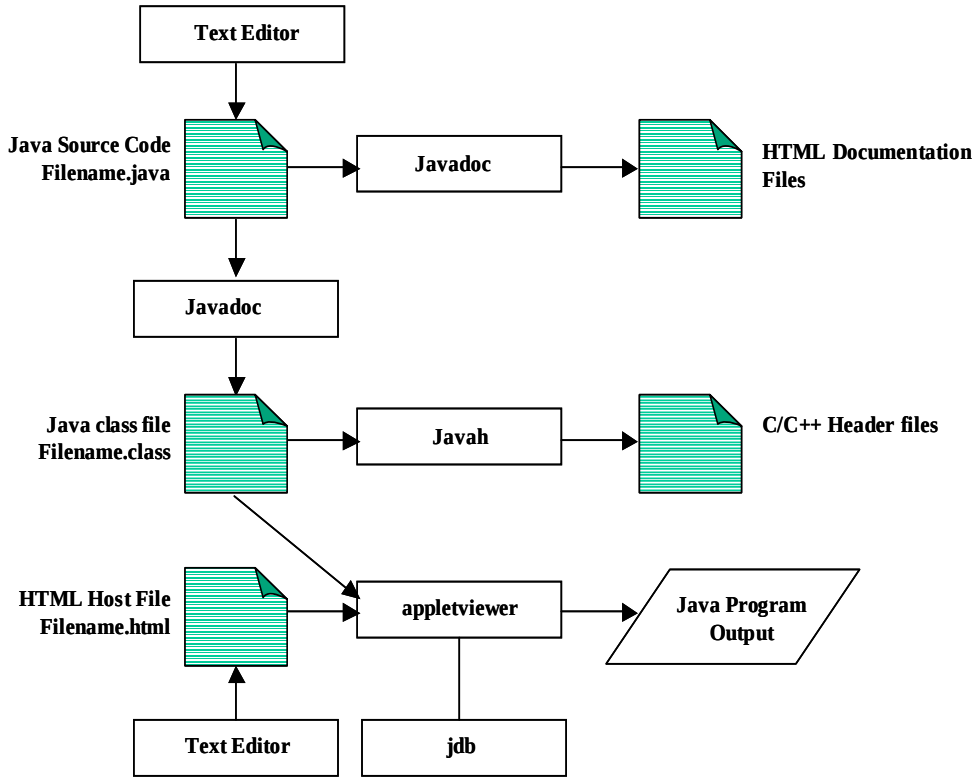


Figure 20.1 : How Java Applets are Built Using JDK

All applets must import java.applet. Since all applets run in a window, it is necessary to include support for that window. Hence all applets must also import java.awt. Execution of an applet is started and controlled with entirely different mechanism. Once an applet has been compiled, it is included in an HTML file using Applet tag. The applet will be executed by a Java enabled web browser when it encounters the Applet tag within an HTML file. The browser contains a JVM to interpret the Java bytecodes, load Java Classes and handle security issues.

The applet class is contained in Java.applet package. Applet contains several methods that give you control over the execution of your applet. In addition, java.applet also defines three interfaces: AppletContext, AudioClip and AppletStub.

Applet Class

The Applet class must start with the two statements–

- import java. applet.*;
- import java.awt.*;

The class used in the program should be extended from System defined Applet class. The Applet class provides a number of entry points such as init( ), start( ), stop( ) and destroy( ) that are called automatically by the browser or applet viewer. To build an applet you must extend applet and override one or more of the methods.

```
public class App extends Applet
{
}        // an example to show how to use Applet class.
```

All applets are subclasses of the Applet class in the java.applet package. When your applet is downloaded, the browser creates an instance of your applet class and calls various documented methods at strategic moments to tell your applet to do something.

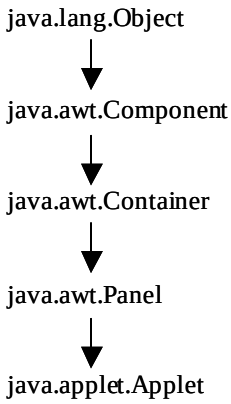
Applets must have two kinds of behaviour:

- Behaviour to work as part of a browser and handle occurrences such as the browser page being reloaded.
- Behaviour to present a graphical user interface and take inputs from users. They appear within the HTML page in the browser and can contain controls such as buttons, list boxes and so on.

All applets must be declared public because the Applet class is a public class.

Architecture

The Applet class is inherited from the Panel class which is inherited from the Container class. The abstract container class inherits many methods from the abstract component class.



An applet is a window based event driven program and it waits until an event occurs. The AWT notifies the applet about an event by calling an event handler that has been provided by applet. Applet should not enter a "mode" of operation in which it maintains control for an extended period. In these situations you must start an additional thread of execution.

The user initiates interaction with an applet. For example, when the user clicks a mouse inside the applet's window, a mouse-clicked event is generated. Applets can contain many controls such as push button check boxes. When the user interacts with one of these controls, an event is generated.

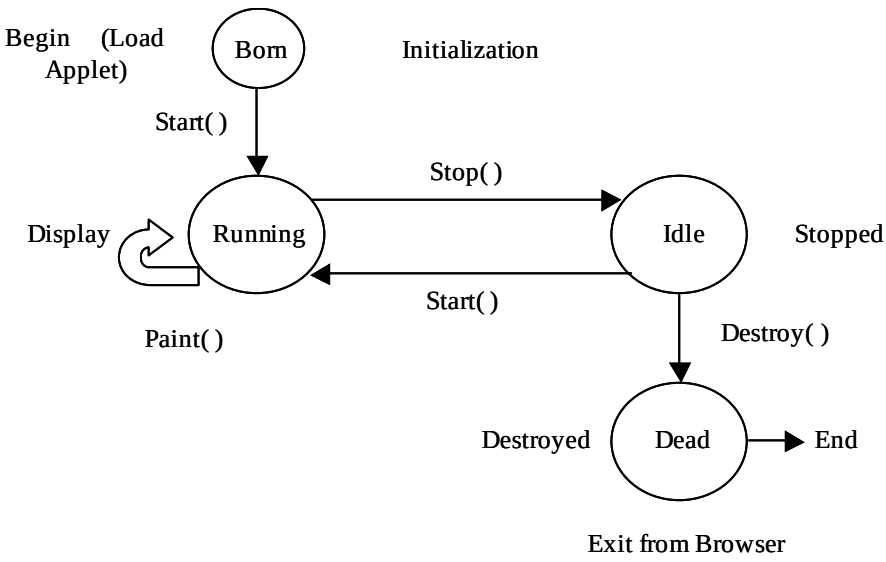


Figure 20.2

```
public class Appskel extends Applet
{
        public void init( )
        {
                // initialization
        }
        public void start( )
        {
                // start or resume execution
        }
        public void stop( )
        {
                // suspend execution
        }
        public void destroy( )
        {
                // perform shutdown activities
        }
        public void paint (Graphics g)
        {
                // redisplay contents of window
        }
}
```

This skeleton generates the blank applet window when viewed with an applet viewer. This skeleton does not do anything.

20.3 APPLET LIFE CYCLE

It can initialize itself. It can start running. It can stop running. It can perform a final cleanup in preparation for being unloaded. When an applet begins, the AWT calls following methods in sequence:

- init( ),
- start( ),
- paint( ).

When an applet is terminated, stop( ) and destroy( ) methods are called in sequence.

Initialization

When the browser downloads an HTML page containing applets, it creates an instance for each of the Applet classes, using the no arg constructor. Applet must have a no argument constructor otherwise it cannot be loaded. Initialization can be done through init( ). The init( ) method is the first method to be called. It is used to initialize the applet each time it is reloaded. Applets can be used for setting up an initial state, loading images or fonts, or setting parameters. For example:

```
public void init( )
{       // code here}
```

Immediately after calling init( ), the browser calls the start( ) method. start( ) is also called when user returns to an HTML page that contains the applet. So, if the user leaves a web page and comes back, the applet resumes execution at start( ). The paint( ) method is called each time your applet's output must be redrawn. This situation can occur for several reasons. For example, window may be minimized and then restored. Paint( ) method is also called when applet begins execution.

Termination

The stop( ) is called automatically whenever the user moves away from the HTML page that contains an applet. The stop( ) method is used to suspend the applet's execution so that it does not take up system resources when the user is not viewing the page or has quit the browser. For example, the processor intensive activities such as animation can be stopped which will not be visible to the user until the user returns to the page. destroy( ) is available for applets that need to release additional resources. The browser calls destroy( ) immediately before the applet is unloaded from memory and after calling its stop( ) method. If you have an applet which displays animation or sound continuously through its start( ) method, such an applet would be very wasteful of processor resources if it keeps on animating even after the user switches to a different page. The effective way would be to put animation in a separate thread. This way whenever the applet's page is displayed by new page, you can simply freeze the thread and when the applet's page is reloaded, you can let the thread run again. This is the purpose of start( ) and stop( ) methods of applet.

Update Method

This method is called when your applet has requested that a portion of its window be redrawn. Basically to use update( ) method, paint( ) method should be clear.

- paint( ) method is used to display on the screen.
- repaint( ) method is used to repaint or redisplay the same area and object but it calls update( ) method first.

update( ) method clears the screen of an existing content by filling it with the background color of the Applet. Then, the update( ) method calls the paint( ) method. So, we can say repaint( ) method calls update( ) which in turn calls paint( ).

Display Methods

Painting is how an applet displays something on screen– be it text, a line, a colored background, or an image. The paint( ) method is used for displaying anything on the applet. paint( ) method takes an argument, an instance of class graphics. The code given can be as follows

```
public void paint (Graphics g)
{ }
```

where Graphics class contains the member functions that can be used to display the output to the browser.

```
text String, the drawString( ) method is used.
g.drawString (String S, int x, int y)
```

where x and y are the coordinate positions on the screen and S is the string to be output.

Example :

```
g.drawString ("hello", 10,20);
The source code is :
import java.applet.Applet;
import java.awt.*;
public class rect extends Applet
```

```
{
public void init( )
{
resize (50, 600); // to set the size of the display window
}
        public void paint (Graphics g)
        {
                g.drawString ("plain text", 15, 48);
                g.setColor (Color.red);
                g.drawString ("colored text", 15, 68);
        }
}
```

drawLine( ) method is used to draw line from starting x and y coordinates to ending x1 and y1 coordinates.

```
Method : drawLine ( )
Syntax : drawLine (int x, int y, int x1, int y1);
```

To set the background color of an applet window and the color in which the text is shown, the following methods are used. These methods are defined in component class. A good place to set background and foreground color is init( ) method.

void setBackground (Color newcolor);

void setForeground (Color newcolor);

You can obtain color settings for background and foreground colors by calling getBackground( ) and getForeground( ) respectively. These are also defined in component class.

```
Syntax :      Color getBackground( )
              Color getForeground( )
```

A simple Applet :

```
import java.awt.*;
import java.applet.*;
public class colorApp extends Applet
{
String msg;
public void init( )
{
setBackground (Color.cyan); // color.cyanis a.constant
setForeground (Color.red); // defined in Color class
}
public void start( )
        {
                System.out.println ("inside Start");
        }
        public void paint (Graphics g)
```

```
        {
                msg + = "Inside paint";
                g.drawString (msg, 10, 30);
        }
}
```

20.4 A SIMPLE BANNER APPLET

Applet writes to its window when its update( ) or paint( ) method is called. How does an applet update its window when the information to be displayed changes ? For example, in a moving banner application, how does the applet use update method when banner scrolls ? Due to the fundamental constraint imposed on applet that it must return control to AWT run time system, we cannot write a loop inside paint( ) that repeatedly scrolls the banner. The solution is to call repaint( ) method defined by AWT. It causes the AWT run-time system to execute a call to your applet's update( ) method which, in its default implementation, calls paint( ).

Syntax of repaint( ) method:

void repaint( ) / / causes entire window to be painted

void repaint(int left, int top, int width, int height);

If you want to update only a small portion of the window, it is more efficient to paint only that region. Region may be defined by its top-left corner and height and width of the window.

NOTE It is possible for a method other than paint( ), update( ) to output to an applet's window. To do so, it must obtain a graphics context by calling getGraphics( ) defined by component class and then use this context to output to the window.

Example :

```
import java.awt.*;
import java.applet.*;
public class BannerApp extends Applet implements Runnable
{
        String msg = "HAPPY COMPUTING";
        Thread t = null
        int state;
        boolean flag;
        // set color and thread
        public void init( )
{
        SetBackground (Color.cyan);
        SetForeground (Color.red);
}
public void start( )
{
        t = new Thread (this);
        flag = false;
        t.start( );
}
```

```
public void run( )
{
        char ch;
        // display banner
        for (; ;)
        {
                try
                {
                        repaint( );
                        Thread.sleep (250);
                        ch = msg.charAt(0);
                        msg = msg.subString (1, msg.
                length( ));
                        msg = msg + ch;
                        if (flag)
                        break;
                }
                catch (InterruptedException c)
                {
                }
        }
}
public void stop( )
{
        flag = true; t = null;
}
        public void paint (Graphics g)
{
        g.drawString (msg, 50, 30);
}
}
```

Student Activity 1

1. What is an Applet ?
2. How Java Applets are built using JDK ?
3. Describe the behavioral attitude of applets.
4. Describe an Applet's life cycle.
5. When does an Applet write to its window ?

20.5 HANDLING EVENTS

One of the things that you learned when creating applets for the first time is that forces are at work behind the scenes as the program is running. Event Handling involves methods that are called

automatically when a user action causes an event to take place. An event is generated in response to just about anything that a user can do during the life cycle of a Java program. Every movement of the mouse, keypress, button click generates an event. Some events are:

```
Mouse Clicks – mouse down, mouse up, mouse clicked.

Key Press – keyPressed, keyReleased, keyTyped.
```

UserInterface event — Button clicked, scrollbar moved up and down, popup menu popped and so on. These events are supported by java.awt.event package.

The approach to handling events is based on the delegation event model which defines standard and consistent mechanism to generate and process events. Its concept is quite simple. A source generates an event and sends it to one or more listeners. In this scheme the listener simply waits until it receives an event. Once received, the listener processes the event and then returns. The advantage of this design is that the application logic that processes event is cleanly separated from user interface logic that generates these events. A user interface element is able to delegate the processing of an event to a separate piece of code.

Event is an object that describes a state change in a source. It can be generated–

- By user Interaction with GUI controls

E.g. pressing a button, entering characters via keyboard, selecting item in the list, clicking.

- Not by interaction with GUI controls.

E.g. when times expires, a counter exceeds a value, a S/W / H/W failure occurs.

Event Source is an object that generates an event. Sources may generate more than one type of event. A source must register listener in order to receive notifications about specific type of event.

```
public void addTypeListener (TypeListener el)

e.g. add.keyListener( ), addMouseMotionListener( )
```

When an event occurs, all registered listeners are notified and receive a copy of the event object. In all cases, notifications are sent only to listener that registers to receive them. Some sources may allow only one listener to register.

```
public addTypeListener (TypeListener el)

throws java.util.TooManyListenersException
```

A source may unregister the listener in a specific type of event.

```
public void removeTypeListener (TypeListener 1);
```

Event Listener is an object that is notified when event occurs. EventListener object must satisfy the following conditions:

- It must have been registered with one or more sources to receive notifications about specific events.
- It must implement methods to receive and process these notifications. For e.g. MouseMotionListener implements dragged and moved events.

Event Object

At the root of Java event class is EventObject. It has the following methods:

- getsource( ) returns source of the event
- goString( ) returns string equivalent of the event

AWTEvent It is a subclass of EventObject and superclass for all AWT events. getID( ) to determine type of the event is one of the methods available in AWTEvent class.

Important classes in java.awt.event package are ActionEvent, AdjustmentEvent, ComponentEvent, ContainerEvent, FocusEvent, ItemEvent, KeyEvent, MouseEvent, TextEvent, WindowEvent.

ActionEvent Class Action Event is generated when button is pressed, list is double clicked or menu item is selected. The following constants of ActionEvent class can be used to identify modifiers associated with an event:

ACTION_PERFORMED --> to identify action events.

ALT-MASK

CTAL-MASK

META-MASK Used to identity any modification associated with the event

SHIFT-MASK

ActionEvent class has following constructors:

ActionEvent (Object src, int type, String cmd)

ActionEvent (object src, int type, String cmd, int modifiers)

object src : reference to the object that generated the event

int type : type of event

String cmd : command string

object entry : specific item

int state : the current state

Methods Object getItem( )

It is used to obtain object reference that generated the event.

The KeyEvent Class The KeyEvent is generated when keyboard input occurs.

There are three types of Events - KEY_PRESSED, KEY_RELEASED, KEY_TYPED.

Constants VK_0, VK_2, VK_A . . . etc. defines the ASCII equivalents of the numbers and letters. Constants VK_ENTER, VK_ESCAPE, VK_UP, VK_LEFT, VK_SHIFT specify virtual key codes.

Constructors

KeyEvent (component src, int type, long when, int modifiers, int c)

Where component src is a reference to the component that generated the event,

long when : systemtime at which key was pressed,

int type : specifies type of the event,

int c : VK_A ... VK_UP } represents virtual key codes

int modifier : are passed here.

KeyEvent (component src, int type, long when, int modifier, int c, char ch)

ch contains character equivalent of key. If no valid character exists then VK_CHAR_UNDEFINED is returned.

Methods char getKeyChar( ) method returns the character that was entered. int getKeyCode ( ) method returns the key code.

MouseEvent Class There are seven events in this class.

MOUSE_CLICKED, MOUSE_DRAGGED, MOUSE_ENTERED, MOUSE_EXITED, MOUSE_MOVED, MOUSE_PRESSED, MOUSE_RELEASED

MouseEvent is subclass of ItemEvent.

Constructor MouseEvent (Component src, int type, long when, int modifier, int x, int y, int clicks, boolean trigpopup)

where,

Component src : specifies reference to the component that generated the event,

int clicks : specifies clickcount,

int type : specifies type of the event,

boolean trigpopup : flag indicates if this event causes a popup menu to appear,

int modifier : specifies which modifiers were pressed when a mouse event occurred.

Methods int getX( ), int getY( ), Point getPoint( )

int getClickCount( ), boolean isPopupTrigger( )

TextEvent Class The TextEvents, are generated by text fields and text areas when characters are entered by user.

constants - TEXT_VALUE_CHANGED

constructor - TextEvent (object src, int type)

Here src is a reference to the object that generated the event and type of the event is specified by the.

WindowEventClass This class will be discussed later in detail. Window getWindow( ) is the most import method which returns Window object that generated the event.

Event Listener Interface

The delegation event model has two parts : sources and listeners. Listeners are created by implementing one or more of the interfaces defined by java.awt.event. When an event occurs, event source invokes the appropriate method defined by the listener. The commonly used listener interfaces are:

```
ActionListener, AdjustmentListener, ComponentListener (4 methods)
ContainerListener (2 methods),
FocusListener (2 methods),
ItemListener (1 method),
KeyListener (3 methods),
MouseListener (5 methods),
MouseMotionListener (2 methods), TextListener
(1 method), WindowListener (7 methods).
Methods in respective interfaces are:
void actionPerformed (ActionEvent ae)
void adjustmentValueChanged (AdjustmentEvent ae)
void componentMoved (ComponentEvent ce)
void focusLost (FocusEvent fe)
void itemStateChanged (ItemEvent ie)
```

```
void keyPressed (KeyEvent ke)
void mouseClicked (MouseEvent me)
void mouseDragged (MouseEvent me)
void textChanged (TextEvent te)
void windowClosing (WindowEvent we)
```

Steps for event handling are:

- Implement appropriate interface for the listener so that it will receive the type of the event desired.
- Register the listener as a recipient for the event notifications.

Source may generate several types of events. Each event must be registered separately. Also an object may register to receive several types of events but it must implement all of the interfaces that are required.

Handling Mouse Events

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code = "MEvents" width = 300 height = 100>
</applet>
*/ public class MEvents extends Applet
implements MouseListener, MouseMotionListener
{
        String msg = " ";
        int x = 0, y = 0;
        public void init( )
        {
        addMouseListener (this);
        addMouseMotionListener (this);
}
public void mouseClicked (MouseEvent me)
{
        x = 0;
        y = 10;
        msg = "Mouse Clicked";
        repaint( )
}
public void paint (Graphics g)
{
        g.drawString (msg, x, y);
}
```

```
public void mouseEntered (MouseEvent me)
{
        x = 0; y = 10; msg = "mouse entered";
        repaint( );
}
public void mousePressed (MouseEvent me)
{
        x = mc.getX( ); y = me.getY( ); msg = "down";
        repaint( );
}
public void mouseDragged (MouseEvent me)
{
        X = megetX( ); y = me.getY( ); msg = "*";
        ShowStatus ("Dragging MouseAt" + x +", " + y);
        repaint( );
}
public void mouseMoved (MouseEvent me)
{
showStatus ("Moving mouseAt" + me.getX( ) + me.getY( ));
}
}
```

NOTE To handle mouse events, you must implement MouseListener and the Mouse Motion Listener interfaces. All the methods of these two interfaces must be implemented in your code. In the above example, you need to add remaining methods also.

Handling Keyboard Events

Each time user presses a key, at least two or often three events are generated in sequence.

- KeyPressed
- KeyReleased
- KeyTyped

Example to demonstrate key event handler is given below.

Program must request input focus before it can process keyboard events.Use requestfocus( ).

```
import java.awt.*;
import java.awt.event;*;
import java.Applet.*;
<applet code = "KEvents" width = 300 height = 300> </applet>
public class KEvents extends Applet implement KeyListener
{
        String msg = " ";
        int x = 10; y = 20;
```

```
        public void init ( )
        {
                addkeyListener (this);
                requestFocus( );
        }
        public void KeyPressed (KeyEvent ke)
        {
        showStatus ("Key Down"); // can also be used to
                        handle special keys.
        }
        public void KeyReleased (KeyEvent ke)
        {
                showStatus ("Key Up");
        }
        public void keyTyped (KeyEvent Ke)
        {
                msg = msg + getKeyCode( );
                repaint( );
        }
        public void paint (Graphics g)
        {
                g.drawString (msg, x, y);
        }
}
```

If you want to handle special keys such as function keys, you need to hande them in KeyPressed( ) method. Some of the function keys can be detected by following constants.

```
KeyEvent.VK_F1
KeyEvent.VK_PAGE_DOWN
KeyEvent.VK_LEFT
KeyEvent.VK_RIGHT
Partial code to check which function key was pressed.
int key = ke.getKeyCode( );
Switch (key)
{
case KeyEvent.VK_F1 : msg + = "<F1>"; break;
case KeyEvent.VK_PAGE_UP:
msg + = "<PageUP>";
break; }
```

Adapter Classes

- Can simplify the creation of event handlers.

- Provide empty implementation of all methods.
- Useful when you want to receive and process only some of the events that are handled by a particular listener interface.

Example :

Adapter Class	Listener Interface
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListner

Example to demonstrate the use of adapter is given below.

```
import . . . . . .
appletCode . . . . . .
public class adapt extends Applet
{
public void init( )
{
        addMouseListener (new MymouseAdapter (this));
        addMouseMotionListener (new MymousemotionAdapter (this));
}
}
class MymouseAdapter extends MouseAdapter
{
        adapt adaptdemo;
        public mymouseAdapter (adapt adaptdemo)
        {
                this.adaptdemo = adaptdemo;
        }
        public void mouseClicked (MouseEvent me)
        {
                adapdemo.showStatus ("Mouse Clicked");
        }
}
class MymousemotionAdapter extends MouseMotionAdapter
{
```

```
        adapt adaptdemo;
        public MymouseMotionAdapter (adapt adaptdemo)
        {
                this.adaptdemo = adaptdemo;
        }
        public void mouseDragged (MouseEvent me)
        {
                adapdemo.showstatus ("Mouse Dragged");
        }
}
The HTML Applet Tag
<HTML> <HEAD> <TITLE> Hello </TITLE> </HEAD>
<BODY>
<Applet Code = "Hello.Class" WIDTH = 300
        HEIGHT = 50>
</APPLET> </BODY> </HTML>
```

To run Applet file in Java we need to make an html file where the Applet Code tag will specify which Java class file has to be displayed. In this case Hello.class is the file. Width and Height property of applet tag specifies how much area in Internet Explorer window the Applet should cover.

Parameters to Applets

Applet parameters come in two parts, a user defined name which indicates the parameter, and a value which determines the value. For each parameter we use the <PARAM> tag which has two attributes for the name and value, called NAME and VALUE.

```
<APPLET CODE = "Hello.class" WIDTH = 100 height = 100>
<PARAM NAME = font VALUE = "Times Roman">
<PARAM NAME = SIZE VALUE = "36">
</APPLET>
```

An example to pass the parameter

```
import java.awt.*;
import java.applet.Applet;
public class param extends Applet
{
        String nam, nam1;
        public void init ( )
        {
                nam = getParameter ("str");
                if (nam = = null)
                nam = " ";
                nam1 = getParameter ("str1");
        }
```

```
public void paint (Graphics g)
{
g.drawString ("hello!" + nam, 10, 50);
g.drawString ("Welcome to the world of Java", 10, 58);
g.drawString (nam1, 100, 100);
}
}
```

To run the above applet, parameters should be passed from the html file. The content of html file —

```
<html>
<Applet code = "param" width = 300 height = 300>
<param name = str value = "UPTEC LTD">
<param name = str1 value = "Kanpur">
</applet>
</html>
```

Example for Displaying Numeric Values

```
import java.awt.*;
import java.applet.*;
public class num extends Applet
{
        public void paint (Graphics g)
        {
                int val1 = 10;
                int val2 = 20;
                int sum = val1 + val2;
                String s = "sum:" + String.valueOf (sum);
                g.drawString (s, 100, 100);
        }
}
```

Example for Getting input from User:

```
import java.awt.*;
import java.applet.*;
public class Input extends Applet
{
        TextField text1, text2;
        public void init( )
        {
text1 = new TextField (8);
text2 = new TextField (8);
add (text1);
add (text2);
```

```
text1. setText ("0");
text2. setText ("0");
}
public void paint (Graphics g)
{
int x = 0, y = 0, z = 0;
string s1, s2, s;
g.drawString ("Input number in each box", 10, 50);
try
        {
        s1 = text1.getText( );
        x = Integer.parseInt (s1);
        s2 = text2.getText( );
        y = Integer.parseInt (32);
        }
}
catch (Exception e)
{
        z = x+y;
        s = String.valueOf (z);
        g.drawString (s, 100, 75);
}
public boolean action (Event event, Object ob)
{
        repaint( );
        return true;
}
}
```

The HTML APPLET tag

<Applet

[codeBase = codebase URL]	:	URL directory for class file
code = Classfilename		
[ALT = alternatetext]	:	Used to specify a short text message that should be displayed if browser can not run Java applet currently.
[WIDTH = pixels]		
[HEIGHT = pixels]		
[Align = alignment]	:	Specifies the alignment of the applets, For e.g. LEFT, RIGHT, TOP, BOTTOM etc.
[VSpace = pixel]	:	Vspace specifies the space in pixels, above and below the applet. HSpace specifies the space in pixels on each side of the applet.
[HSpace = pixel]		
[<param name = Attrname		
value = Attrvalue>]		

[HTML text displayed in absence of Java]

</Applet>

Student Activity 2

1. What is an event ?
2. Define event source.
3. Define event listener.
4. Which methods are included with Event Object ?
5. Define AWT Event.
6. Define Action EventClass.
7. Give the constructor of various event classes.
8. Give the methods of various event classes.
9. What are listeners ? Name some common listner interfaces.
10. Give on example to demonstrate the use of adapter.

20.6 GETDOCUMENTBASE( ), GETCODEBASE( ), SHOWDOCUMENT( ), APPLETCONTEXT INTERFACE GETDOCUMENTBASE( )

- To load anything into the applet we need to pass the relative path of that. For example, if we want to load Image then the code given can be

  Image img = getImage (getDocumentBase( ), "Images / book.gif")

- Here, the base address of the current HTML file is produced by getDocumentBase( ) method.
- So, getDocumentBase( ) method returns a URL object that represents the folder containing the web page presenting the applet.

  getCodeBase( )

- The getCodeBase( ) method returns a URL object that represents the folder where the applets main class file is located.
- getCodeBase( ) depends on whether your images are stored in subfolders of your Java applet or subfolders of the applet's web page.

```
Syntax :      URL url = getCodeBase( ); // main class file
              URL url = getDocumentBase( );  // html file or images
```

Example to display code and document Bases:

```
import java.awt.*;
import java.applet.*;
import java.net.*;
public class codeBaseApp extends Applet
{
        public void paint (Graphics g)
        {
                String msg;
                URL url = getCodeBase( )
                msg = "Code Base:" + url.toString( )
                g.drawString (msg, 10, 20);
```

```
                url = getDocumentBase( );
                msg = "Document Base:" + url.toString( );
                g.drawString (msg, 10, 40);
        }
}
```

The above example displays the directory name holding the HTML file and full path of the file holding the object code of .java file.

- It is also possible for the applet to display the result of the CGI program as a web page, just as if it were running in normal HTML mode.
- To allow your applet to transfer control to another URL, use the showDocument(u) method where 'u' in the parameter is the URL or Uniform Resource Locator i.e., the whole path.
- AppletContext is an interface which has details of Applet. It has got some methods which help in accessing information about an Applet's execution environment.
- getAppletContext( ) is one of the methods which is used to get access to the currently executing applet in the browser. Once you get Applet context you can call ShowDocument( ) to bring another document in view.

Partial code for ShowDocument( ) and getAppletContext( ):

```
public void start( )
{
        AppletContext ac = getAppletContext( );
        URL url = getCodeBase( )
        try
        {
        ac.showDocument (new URL (url + "text.html"));
        }
        catch (MalformedURLException e)
        {
                ShowStatus ("URL Not FOUND");
        }
}
```

20.7 AUDIOCLIP AND APPLETSTUB INTERFACE

Inside the package java.applet we have the interfaces AppletContext, AppletStub and AudioClip. AppletStub interface supports communication between an applet and its browser's environment and is used to develop custom applet viewers. AudioClip interface provides methods that support the playing of audio clips such as play( ), stop( ), loop( ). After you load audio clips using getAudioClip( ) you can use these methods to play it.

Student Activity 3

1. Give an example to display code and document bases.
2. Define AppletContext.
3. What is the role of AppletStub Interface ?
4. What is the role of AudioClip Interface ?

20.8 SUMMARY

- Applets are small graphical applications generally accessed from Internet Servers and run within a web browser or Applet Viewer.
- Applets have limited access to local resources, so they can be used without risk of introducing viruses or violating data security.
- Applets must extend the Applet class and override certain methods such as init( ), start( ), stop( ) destroy( ), paint( ), update( ).
- Applets may exist in different states such as New Born state, Running State, Idle state and Dead state.
- Parameters can be passed to applets to make them more flexible.
- Applets can show another document from different URL using showDocument( ) method available in AppletContext interface.
- Audio clips and video clips can be played from within an applet.
- Applet can respond to events generated by users such as keypress, mouseclicked etc.

20.9 KEYWORDS

Applet : Small graphical applications generally accessed from Internet Servers and run within a web browser or Applet Viewer.

Event Handling : Involves methods that are called automatically when a user action causes an event to take place.

Event : An object that describes a state change in a source.

Event Source : An object that generates an event.

Event Listeners : An object that is notified when event occurs.

Action Event : In event generated when button is pressed, list is double clicked or menu item is selected.

KeyEvent : An event generated when keyboard input occurs.

TextEvent : An event generated by text fields and text areas when characters are entered by user.

AppletStub Interface : An interface that supports communication between an applet and its browser's environment and is used to develop custom applet viewers.

AudioClip Interface : An interface that provides methods that support the playing of audioclips such as play( ), stop( ), loop( ).

20.10 REVIEW QUESTIONS

1. What is an applet ?
2. Write the difference between an applet and an application.
3. Discuss the steps involved in running an applet.
4. Why does applet class need to be declared as public ?
5. Discuss different stages in the life cycle of an applet.
6. Distinguish between init( ) and start( ).
7. Frames and applets cannot be used together in the same program. True / False.
8. The CODE value in an <applet> tag must name a class file that is in the same directory as the calling HTML page. True / False.

9. If getParameter ( ) returns null, then assigning the return value to a variable of type string may cause an exception to be thrown. True / False.

10. When we invoke repaint( ) for a component, the AWT invokes which of the following methods?

    a. draw( )

    b. show( )

    c. update( )

    d. paint( )

11. The setBackground( ) method is part of the class

    a. Graphics

    b. Applet

    c. Component

    d. Container

    e. Object

12. If you want to assign a value to the variable year, then which of the following lines can be used within an <applet> tag ?

```
a.  number = getParameter (99)
b.  <number = 99)
c.  <param = radius value = 99>
d.  <param name = number value = 99>
e.  <param number = 99>
```

13. Which of the following applet tags is legal to embed an applet class named Test into a web Page ?

```
a.  <applet class = Text width = 200 height = 100>
b.  <applet code = "Test.class" width = 200 height = 100>
c.  <applet code = Text.class width = 200 height = 100 </applet>
d.  <applet code = Text.class width = 200 height = 100> </applet>
```

14. Which of the following HTML code contains an error ?

```
a.  <applet
b.  width = 400 height = 200
c.  code = Hello.class>
d.  <param
e.  name = "string" value = "Hello">
f.  </applet>
```

15. Which of the following statements about hierarchy of the class java.awt.Applet is incorrect?

    a. An applet is a kind of container

    b. An applet is a kind of window

    c. An applet is a kind of Component

    d. An applet is a kind of panel.

16. When an Applet uses an InputStream created by URL, which network protocol is involved ?

    a. UDP

    b. ISO8859-1

c. HTTP

d. Multicast

17. Develop an applet that receives three numeric values as input from the user and then displays the largest of the three on the screen.

18. Develop an applet that will display a filled rectangle changing to a filled ovals, changing to a filled circle alternatively while moving across the screen.

19. Develop an applet using multithreading to move applet window up and down continuously.

Answers to Review Questions :

10.	(c)	11.	(c)	12.	(d)	13.	(b)
14.	(c)	15.	(b)	16.	(a)		

20.11 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased.

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

UNIT

21

AWT : WINDOWS, GRAPHICS AND TEXT

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe various AWT classes
- Describe window fundamentals
- Work with frame windows
- Understand event handling in a frame window
- Understand how to display information while working with graphics and color
- Work with fonts
- Manage text output using fontmetrics
- Explore text and graphics

U N I T S T R U C T U R E

21.1 INTRODUCTION

This unit covers the classes and interfaces of the Abstract Windows Toolkit (AWT). Here you will learn how the AWT classes and interfaces are used to create the Graphical User Interface (GUI) of applets and stand alone applications.

21.2 AWT CLASSES

The classes and interfaces of the Abstract Windows Toolkit (AWT) are used to develop stand alone applications and to implement the GUI controls used by applets. Some of the important classes are:

AWTEvent	It encapsulates AWT events.
BorderLayout	This manager uses five components North, South, East, West and Center.
Button	It creates a push button control.
Canvas	A blank semantics free window.
CardLayout	It emulates index cards. Only one card is shown on top at a time.
Checkbox	It creates a checkbox control.
CheckboxGroup	It creates a group of checkbox control.
CheckboxMenuItems	It create an on/off menu item.
Choice	It creates a popup list.
Color	It manages color in a portable, platform independent fashion.
Component	This is an abstract superclass for various AWT components.
Container	This is a subclass of component that can hold components.
Dialog	It creates a dialog window.
Dimension	It specifies the dimensions of an object, the width is stored in width and the height is stored in height.
Event	It encapsulates event.
FileDialog	It creates a window from which a file can be selected.
FlowLayout	It positions components left to right, top to bottom.
Font	It encapsulates a type font.
FontMetrics	Encapsulates the information related to fonts.
Frame	It creates a standard window that has a title bar, resize corners and a number.
Graphics	Encapsulates the Graphics Context.
GraphicsDevice	Such as a screen or printer.
GridbagConstraints	Various constraints related to Gridbag layout.
GridbagLayout	Layout manager displays components subject to constraints.
GridLayout	Two dimensional grid.
Image	Encapsulates Image.
Insets	Borders of a container.
Label	An item used only for display.
List	Collection of items.
Menu	Creates a pulldown menu.

Menubar	Creates a menubar.
MenuItem	Creates a menu item.
Panel	Concrete subclass of Container.
Point	Cartesian coordinate pair stored in X and Y.
PopupMenu	Encapsulates popup menu.
Rectangle Scrollbar	An item used in a window to scroll.
ScrollPane	Class implements a container with scrollbars so that a large component can be viewed through a small viewport.
TextArea	Class which gives you an area to type the text along with which you can specify the size by rows and columns.
TextField	A class which let you create an area to type the text.
TextComponent	Superclass for TextArea and TextField.
ToolKit Window	With no frame, no menubar, and no title.

21.3 WINDOW FUNDAMENTALS

The AWT defines windows according to a class hierarchy that adds functionality and specificity with each level. The two most common windows are those derived from Panel which is used by applets and those derived from Frame, which creates a standard window.

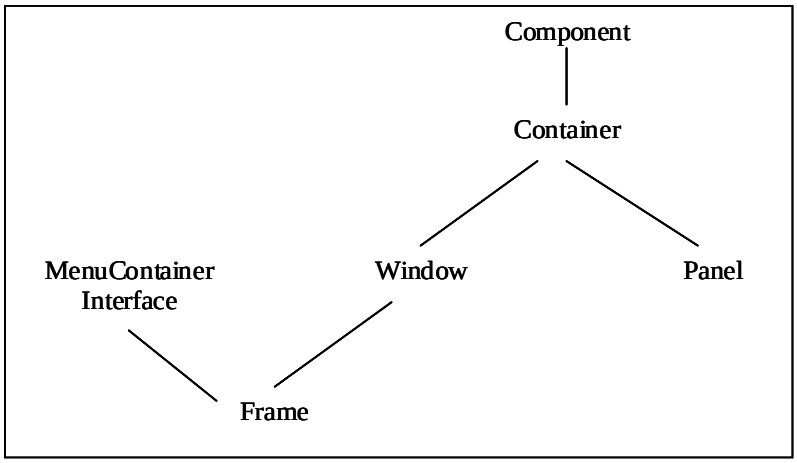


Figure 21.1

The Window class creates a top level window. A top level window is not contained within any other object, it sits directly on the desktop. Generally, you won't create window objects directly. Instead, you will use a subclass of Window called Frame.

Frame

Frame encapsulates what is commonly thought of as a "Window". It is a subclass of window and has a title bar, menubar, borders and resizing corners. If you create a Frame object from within an applet, it will contain a warning message, such as "Warning : Applet Window," to the user that an applet window has been created.

Canvas

Although it is not part of the hierarchy for applet or frame windows, there is one other type of window that you will find valuable. Canvas encapsulates a blank window upon which you can draw.

Panel

The panel class is a concrete subclass of container. It does not add any new methods, it simply implements container. A panel may be thought of as recursively restable, concrete screen component.

Student Activity 1

1. What is the role of AWT classes and interfaces ?
2. What is the role of following AWT classes :

   (a) AWTEvent (b) Button (c) Choice (d) Component

   (e) Event (f) Frame (g) Graphics (h) Image

   (i) List (j) Menu (k) Panel (l) Point
3. Give the class hierarch of windows.
4. Define the following :

   (a) Frame (b) Canvas (c) Panel

21.4 WORKING WITH FRAME WINDOWS

After the applet, the type of window you will most often create is derived from Frame. You will use it to create child windows within applets and top level or child windows, for applications.

Frame supports these two constructors:

```
Frame ( )

Frame (String title)
```

Setting the Window's Dimensions:

setSize( ) method is used to set the dimensions of the windows. Its signature is:

```
void setSize (int newwidth, int newheight)

void setSize (Dimension newSize)
```

Dimension getSize( ) returns the current size of the window contained within the width and height field of a Dimension object.

Hiding and Showing

After a frame window has been created, it will not be visible until you call setVisible( ).

```
Void setVisible (boolean visibleFlat)
```

The component is visible if the argument to this method is true, otherwise it is hidden.

Setting a Window's Title

You can change the title in a frame's window using setTitle( ).

```
void setTitle (String newTitle)
```

Here, newTitle is the new file for the window.

Closing the Frame Window

When using frame window, your programs must remove that window from the screen when it is closed by calling setVisible (False). To intercept a window close event you implement the windowClosing( ) method of the WindowListener interface. Inside windowClosing( ) you must remove the window from the screen.

21.5 FRAME WINDOW IN AN APPLET

While it is possible to simply create a window by creating an instance of Frame, you will seldom do so because you will not be able to do much with it. For example, you will not be able to receive or process events that occur within it or easily output information to it. Most of the time you will create a subclass of Frame.

Program

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
class sampleframe extends Frame
{
        sampleframe (string title)
        {
                super (title);
                MyWindowAdapter adapter = new MyWindowAdapter (this);
                addWindowListener (adapter);
        }
        public void paint (Graphics g)
        {
        g.drawString ("This in frame window", 10, 40);
        }
}
class MyWindowAdapter extends WindowAdapter
{
        sampleframe sp;
        public MyWindowAdapter (sampleframe sp)
        {
                this.sp = sp;
        }
public void windowClosing (WindowEvent we)
        {
                s.setVisible (False);
        }
}
// Create frame window
public class AppletFrame extends Applet
{
        Frame f;
        public void init( )
        {
                f = new SampleFrame ("A frame Window");
```

```
                f.setSize (250, 250);
                f.setVisible (true);
        }
        public void start( )
        {
                f.setVisible (true);
        }
        public void stop( )
}
f.setVisible (false);
{
        public void paint (Graphics g)
        {
        g.drawString (This is an applet window", 10, 20);
        }
}
```

21.6 EVENT HANDLING IN A FRAME WINDOW

Since frame is a subclass of component, it inherits all the capabilities defined by component. This means that you can use and manage a frame window that you create just like you manage applet's main window. Whenever an event occurs in a window, the event handlers defined by that window will be called. Each window handles its own events. The events are:

```
MouseEntered
MouseDragged
MouseClicked
MousePressed
MouseMoved
```

21.7 A WINDOWED PROGRAM

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class Appwindow extends Frame
{
        String keymsg = " ";
        String mousemsg = " ";
        int mouseX = 30; mouseY = 30;
        public appwindow( )
        {
        addkeyListener (new MyKeyAdapter (this));
        addMouseListener (new MyMouseAdapter (this));
        addWindowListener (new MyWindowAdapter( );
```

```
        }
        public void paint (Graphics g)
        {
                g.drawString (keymsg, 10, 40);
                g.drawString (mousemsg, mouseX, mouseY);
        }
        public static void main (String args[ ])
        {
                appwindow ap = new appwindow( );
                ap.setSize (new Dimension (300, 200));
                ap.setTitle ("An awt based application");
                ap.setVisible (true);
        }
}
class myKeyAdapter extends KeyAdapter
{
        appwindow ap;
        public MyKeyAdapter (appwindow ap)
        {
                this.ap = appwindow;
        }
        public void keyTyped (keyEvent me)
        {
                ap.keymsg + = ke.getKeyChar( );
        ap.repaint( );
        }
}
class MyMouseAdapter extends MouseAdapter
{
        appwindow app;
        public MyMouseAdapter (appwindow ap)
}
{
        this.ap = ap;
}
public void mousePressed (MouseEvent me)
{
        ap.mouseX = me.getX( );
        ap.mouseY = me.getY( );
        ap.mousemsg = "Mouse down at" + ap.mouseX + ","+ ap.mouseY;
```

```
        ap.repaint( );
}
}
class MyWindowAdapter extends WindowAdapter
{
        public void windowClosing (WindowEvent me)
        {
                System.exit(0);
        }
}
```

21.8 DISPLAYING INFORMATION WHILE WORKING WITH GRAPHICS AND COLOR

The AWT supports a rich assortment of graphics methods. All graphics are drawn relative to a window. This can be the main window of an applet, a child window of an applet or a stand alone application window. The origin of each window is at the top left corner and is 0, 0. Coordinates are specified in pixels. All output to window takes place through a graphics context. A graphics context is encapsulated by the Graphics class and is obtained in two ways:

- It is passed to an applet when one of its various methods, such as paint( ) or update( ) is called.
- It is returned by the getGraphics( ) method of Component.

Drawing Lines

```
void drawLine (int startX, int startY, int endX, int endY)
Drawing Rectangles
        drawRect (int top, int left, int width, int height)
        fillRect (int top, int left, int width, int height)
        void drawRoundRect (int top, int left, int width, int height,
                                    int xdrain, int ydrain)
        void fillRoundRect (int top, int left, int width, int height,
                                    int xdrain, int ydrain)
        void drawOval (int top, int left, int width, int height)
        void fillOval (int top, int left, int width, int height)
```

Drawing Arcs

```
        void drawArc (int top, int left, int width, int
                            height, int startAngle int sweepAngle)
        void fillArc (int top, int left, int width, int height, int
                            startAngle, int sweepAngle)
```

Working With Color

```
Color (int red, int green, int blue)
Color (int rgbValue)
Color (float red, float green, float blue)
```

The first constructor takes three integers that specifies the color as a mix of red, green and blue. These values must be between 0 and 255.

```
new Color (255, 100, 100);
```

```
an int newRed = (0Xff 000000|+ 0XC0<<16) | (0X00<<&);
Color darkRed = new Color (newRed);
```

You can obtain the red, green and blue components of a color independently.

```
int getREd( ) int getRed( )
int getBlue( ) int getBlue( )
int getGreen( ) int getGreen( )
int getRGB( ) int getRGB( )
```

Setting the Current Graphics Color:

```
void setColor (Color newColor)
Color getColor( )
```

Setting the Paint Mode

The paint mode determines how objects are drawn in a window. By default, new output to a window overwrites any preexisting contents. However, it is possible to have new objects XORed onto the window by using setXORMode( ) as follows:

```
void setXORMode (Color XorColor)
```

Here, XorColor specifies the color that will be XORed to the window when an object is drawn. The advantage of XOR mode is that the new object is always guaranteed to be visible no matter what color the object is drawn over.

To return to overwrite mode, call setPaintMode( ) void setPaintMode( )

21.9 WORKING WITH FONTS

The AWT supports multiple type fonts. Fonts have emerged from the domain of traditional typesetting to become an important part of computer generated documents and displays.

Beginning with Java2, fonts have a family name, a logical font name and a face name. Fonts are encapsulated by the Font Class.

Methods

String getFamily( ) returns the name of the font family.

```
static Font getFont(String property)
static String getFontName( ) returns the face name
int getSize( )
int getStyle( )
boolean isBold( )
boolean isItalic( )
boolean isPlain( )
String toString( )
import java.applet.*;
import java.awt.*;
public class showfonts extends Applet
{
public void paint (Graphics g)
{
```

```
String msg = " ";
String FontList [ ];
Graphic Environment.getLocalGraphicsEnvironment( );
FontList = g.getAvailableFontFamilyNames( );
for (int i = 0; i < FontList.length; i++)
msg + = FontList [i] + " ";
g.drawString (msg, 4, 16);
}
}
```

21.10 MANAGING TEXT OUTPUT USING FONTMETRICS

Java supports a number of Fonts. For most fonts, characters are not all of the same dimensions – most fonts are proportional. Also the height of each character, the length of descenders, the amount of space between horizontal lines vary from font to font. The AWT includes FontMetrics class. which encapsulates complete information about a font.

Height	Baseline	Ascent
Descent	Leading	

FontMetrics defines several methods that help you manage text output.

Method

`int bytesWidth(byte b[ ], int start, int numBytes)`	
`int charWidth(char ([ ], int start, int numChars)`	
`int getAscent( )`	
`int getDescent( )`	Returns the descent of font.
`Font getFont( )`	Returns the font.
`int getHeight( )`	Used to output multiple lines of text.
`int getLeading( )`	Returns the space between lines of text.
`int getMaxAdvance( )`	Returns the width of the widest character. -1 is returned if the value is not available.
`String toString( )`	

Program

```
import java.applet.*;
import java.awt.*;
public class Multiline extends Appllet
{
        int curX = 0, cury = 0;
        public void paint (Graphics g)
        FontMetrics fm = g.getFontMetrics( );
        nextLine ("This is an Line one", g);
        sameLine ("This is on line two", g);
        sameLine ("This is on same line ", g);
        nextLine ("This is on Line three", g);
}
void nextLine (String s, Graphics g()
{
```

```
FontMetrics fm = g.getFontMetrtics( );
cury + = fm.getHeight( ); // advance to the next line
curX = 0;
g.drawString (S, curX, curY);
curX = fm.StringWidth(s);
}
void sameLine (String s, Graphics g)
{
        FontMetrics fm = g.getFontMetrics( );
        g.drawString (S, curX, curY);
        curX + = fm.stringWidth(s);
}
```

21.11 EXPLORING TEXT AND GRAPHICS

Exploration in text and graphics part is still going, and further refinements and enhancements are expected. The Java 2D supports enhanced control over graphics, including such things as coordinate translations, rotation and scaling.

Student Activity 2

1. Write the constructors supported by frame.
2. Write a method to :
   (a) Show a frame window
   (b) Set a window's title
   (c) Close the frame window
   (d) Set the paint mode
3. How can you obtain a graphics context ?
4. What is the role of fontmetrics ? Describe some of its methods.

21.12 SUMMARY

- Java has a complete Application Program Interface (API). The "java.awt" package is an Abstract Windows Toolkit that can add graphics to applets and applications.
- The java.awt package includes classes for font, color and graphics.
- The Color class contains methods that control the color settings of User Interface Components.
- The Font class contains fonts including their names, sizes and styles. The graphics class can be used with the font (and FontMetrics) classes to display text in different font styles,
- The Graphics class can be used to draw figures and images using X-Y coordinate system.

21.13 KEYWORDS

Frame : Encapsulates what is commonly thought of as a "window". It is a subclass of window and has a title bar, menubar, borders and resizing corners.

Canvas : Encapsulates a blank window upon which you can draw.

Panel : A concrete subclass of container which recursively restable concrete screen components.

Answers to Review Questions

1. (a) AWT (b) drawoval( ) (c) drawrect( )

2. (a) True (b) True (c) false (d) true

21.14 REVIEW QUESTIONS

1. Fill in the Blanks.
   a. The _____ package provides graphics capability in Java.
   b. _____ method is used to draw a Circle.
   c. Graphics can be drawn in any method of an applet except the _____ method.
   d. The default font in Java is generally _____ .
   e. The _____ method can draw a rectangle without round edges.
2. State whether the following are true or false.
   a. The setColor( ) method sets the background color of the canvas.
   b. The new keyword is used to create a graphics object.
   c. Graphics class methods should be used in the init( ) method of an applet.
   d. The getFont( ) method returns the current font.
3. Answer the following questions:
   a. What is the difference between checkbox and checkbox group classes ?
   b. Define canvas, panel, and container. Find out the similarities and dissimilarities, if any.
4. Write an applet that displays a string in blue against a green background. Then reset the original background and foreground colors and display the string again. Also determine the amount of red, green, and blue colors in the foreground color.
5. Determine the font name, size and style of the string and modify font to TimesNewRoman, 28, bold and italicized.
6. Create the following figures
   a. A line from (10, 10), to (50, 50)
   b. A rectangle of width and height 40 pixels respectively from (10, 10)

21.15 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT

22

AWT : CONTROLS, LAYOUTS AND MENUS

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe various control fundamentals
- Describe various layouts
- Describe menus
- Describe Dialog class

U N I T S T R U C T U R E

22.1 INTRODUCTION

User Interface (UI) refers to dynamic and interactive communication between a program and its users. The package "java.awt" (Abstract Windows Toolkit) of JDK gives a portable set of nested components, starting from the outermost window all the way down to the smallest UI components.

22.2 CONTROL FUNDAMENTALS

The major User Interface elements of AWT that support the development of GUI are:

- **UI Components** These include Labels, Text fields, Menus, Buttons and other typical elements of UI Interface.
- **Layout Managers** Layout Managers control the appearance of the display i.e., they can be used to present the components of the container which can be positioned as desired.
- **Containers** Containers are generic AWT components that can contain other components. including containers e.g. FRAME, DIALOG, MENUBAR.
- **Events** The events system enables responding to the interaction between the components and the containers for the application.

The basic UI Components include Label, TextField, TextArea, Button, Checkbox, RadioButton, Choice, List, Scrollbar, ScrollPane.

Label

Component Classes

- The Label Class

Labels are text strings that can be used to label other UI Components.

- Constructor Methods

To create labels use one of these constructors

a. Label( ) // creates an empty label.

b. Label(String) // creates a left aligned label with the given text.

c. Label(String, int) // creates a label with the given text and the specified alignment. The available alignments are stored in the class Label as Label.LEFT, Label.CENTER, Label.RIGHT.

Labels can be created with or without any object of the class.

```
add (new Label ("Label"));
add (new Label ("label - center aligned", Label.CENTER));
add (new Label ("label - right aligned", Label.RIGHT));
Label lbl = new Label ("Label");
add(lbl);
```

The TextField Class

A TextField is a subclass of the TextComponent class. It can hold a single row of text and its width is either set to accommodate a string used in the constructor or set by an int parameter specifying the number of columns.

Constructor Methods

To create a textfield use any of the following methods:

- TextField( ) creates an empty textfield.
- TextField(String) creates a textfield initialized with the given string.
- TextField(int) : creates a textfield with the given width in characters and initialized with the given string.

TextField can be created with or without any identifier of the class but since they have events associated with them, they preferably should be created with an identifier.

```
TextField tf = new TextField ("First Name", 20);
add (tf);
add (new TextField ("Last name", 20));
```

Here's a complete program

```
import java.applet.Applet.*;
import java.awt.*;
public class textandLabel extends Applet
{
public void init( )
```

```
        {
                add (new Label ("FirstName", Label.CENTER));
                add (new TextField (30));
                add (new Label ("LastName", Label.CENTER));
                add (new TextField(30));
        }
}
```

Other TextField Methods

```
getText( )          // returns the text entered.
isEditable( )       // returns true or false based on whether the
                    field is editable or not.
setEditable( )      // enables you to control whether the contents
                    of a text field may be modified by the user.
setEchoChar(char)   //obscures the characters typed into the field.
                    Used to create password textfield.
select(int, int)    //returns the text between the two int positions.
```

TextArea Class

TextArea also is a subclass of Text Component class. It is a multiline input area that includes scrollbars so that large amount of text can be entered and displayed.

Constructor Methods

TextArea (int rows, int cols) creates a textarea of the specified number of rows and columns with horizontal and vertical scrollbars.

TextArea (Strings, int row, int cols) creates a textarea of the specified number of rows. TextArea can be created as add (new TextArea ("personal info", 10, 50);

```
TextArea ta = new TextArea (10, 50);
add (ta);
```

Other Methods

All these methods can be invoked only by the objects of the class.

```
append (String S) // appends text to the text area.
insert (String S, int pos) // inserts text at the specified position
within the text area.
```

Button Class

Buttons are simple UI components that perform actions when they are clicked i.e., they require instructions to handle user events such as mouse clicks etc.

Constructor Methods

Button (String S) // creates a button with a label.

To create and add a button to the applet. Button `btn = new Button ("Click Me");`

```
add (btn);
```

Checkbox Class

Checkboxes are user interface components that have two states : Checked and Unchecked (true or false). Checkboxes can be of two types:

Non Exclusive Checkboxes

These checkboxes enable you to select any number of checkboxes within a given series.

Constructor Methods

Checkbox( )	:	Creates an empty and unselected checkbox.
Checkbox(String)	:	Creates a checkbox with a given string as label.
Checkbox(String, null, boolean)	:	Creates a checkbox with a given string as Label. Null specifies the group name, if any (used in case of radio buttons) and boolean specifies the state of a checkbox that is either selected or unselected based on whether the boolean argument is true or false.

To create and add a checkbox:

- ```
  Checkbox cb = new Checkbox ("BTECH");
  add (cb);
  ```
- ```
  Checkbox cbl = new Checkbox ("MCA");
  add (cb1);
  ```
- ```
  Checkbox gr = new Checkbox ("Coord", null, true);
  ```

Exclusive Checkboxes

Only one checkbox can be selected from within a group. These types of checkboxes are also known as radio buttons. They have the appearance of checkboxes.

Constructor Methods

To create a series of radio buttons, first create an instance of CheckboxGroup, then create the button. Each button requires three parameters to be specified– label, group name, and whether it is initially selected (true) or not (false). Finally add the group.

```
CheckboxGroup cbg = new CheckboxGroup( );
Checkbox cb1 = new Checkbox ("read', cbg, True);
Checkbox cb2 = new Checkbox ("blue", cbg, false);
Checkbox cb3 = new Checkbox ("Green", cbg, false);
```

Other Methods

`getState( )` // to check current state of the checkbox i.e., whether selected or not selected.

`setState( )` // to set the state i.e., true or false setState (true);

Choice Pull Down List

A choice list appears like a menu and enables selection of an item from that menu. The advantage of choice list over radio buttons is, that it occupies less space on the window. To create a choice list, create an instance of the class Choice.

```
Choice dessert = new Choice ( );
```

To add items into the list

```
dessert.addItem ("Chocolate");
dessert.addItem ("Vanilla");
// to add the choice list to the Panel.
add (dessert);
```
```

Other Methods

getItem(int) : returns the label of the specified item number.

CountItems( ) : returns the count of items in the list.

getSelectedIndex( ) : returns the index position of the item that's selected.

getSelectedItem( ) : returns the selected item as a string.

select (String) : selects the item of the specified string.

ScrollBars

Scrollbars are used to select continuous values between a specified minimum and maximum. Scrollbar may be oriented horizontally or vertically. Each end has an arrow that you can click to move the current value of the scroll bar one unit in the direction of the arrow. The constructors are as follows

```
Scrollbar (int style)

Scrollbar (int style, int initial, int thumbsize, int min, int max)
```

Here style specifies orientation of the scrollbar. The number of units represented by the height of the thumb is passed in thumb size.

int getValue( ) : It returns current value of the scrollbar.

void setValue (int newval) : Sets the new value for the scrollbar and sliderbox will be positioned to reflect the new value.

int getMinimum( ) } : Returns minimum and maximum values.
int getMaximum( ) }

Student Activity 1

1. Name various user interface components.
2. What is the role of layout manager ?
3. What are containers ?
4. Define events.
5. Define the following :

   (a) Labels (b) Texfield (c) Textarea (d) Button

   (e) Checkbox (f) Choice list (g) Scrollbars

6. Write the constructors of the above components.

22.3 LAYOUTS

Java programs are designed for use on multiple platforms. At run time, they need to adapt to the Operating System as they use the native windows and controls on that system, which is important for the customer and also for the marketability of the product.

Standard Layout Managers

You must have observed that as the components are added to the panel, they do not come properly formatted. To format the output and to give it a better look, AWT has provided five standard layout managers and each of them provides different rules.

1. FlowLayout
2. GridLayout

3. BorderLayout
4. CardLayout
5. GridbagLayout

To change to any of the layouts use the following methods

```
setLayout (new layoutmanager( ));
```

For example,

```
setLayout (new GridLayout( ));
```

FlowLayout

The FlowLayout is the default layout for applet and panel. This layout adds objects to a container in rows, from left to right. When the first row fills the container, the components are wrapped to the next row automatically. With FlowLayout, only the alignments and the horizontal and vertical spacing between the components can be changed.

```
setLayout (new FlowLayout( ));
setLayout (new FlowLayout (FlowLayout.LEFT));
setLayout (new FlowLayout (FlowLayout.LEFT), 10, 15);
```

This sets the components left aligned with a gap of 10 pixels between components on the same row and 15 pixels between rows.

Example :

```
import java.applet.Applet;
import java.awt.*;
public class lays extends Applet
{
        public void init( )
        {
        setLayout (new FlowLayout (FlowLayout.LEFT));
        add (new Label ("Branch", Label.CENTER));
        add (new Label ("Branch Code", Label.CENTER));
        add (new TextField (10));
        }
}
```

GridLayout

The GridLayout class divides the screen into equal sized rows and columns producing a spreadsheet arrangement. In the constructor function of GridLayout rows, columns and horizontal and vertical gaps (in pixels) can also be specified, but these are optional. These components are added cell by cell from left to right, filling the top row and then moving down.

To divide the window in three rows and columns:

```
setLayout (new GridLayout (3, 3));
```

To divide the window into 3 equal rows and columns with 10 pixels between columns and 15 pixels between rows:

```
setLayout (new GridLayout (3, 3, 10, 15));
```

Example :

```
import java.applet.Applet;
import java.awt.*;
public class lays extends Applet
{
        public void init( )
{
                setLayout (new GridLayout (3, 12));
                add (new Label ("First Name"));
                add (new TextField (10));
                add (new Label ("Second Name"));
                add (new TextField (10));
                add (new Label ("password"));
                add (new TextField (10));
        }
}
```

BorderLayout

BoderLayout Manager is a simple layout that places control so that they fill their container. Its a default layout for frame and dialog. When you add a component to the container that has BorderLayout, you can also specify one of the five constraints in the form of a string, "North", "East", "South", "West", and "Center". If you do not specify a constraint the default constraint is "center". A maximum of five components can be displayed in the container. The BorderLayout manager is the default layout for the Window Class so frames and dialog boxes use the BorderLayout manager if none is specified.

```
add (new Button ("West"), "West");
add (new Button ("Center"));
```

CardLayout

The cardlayout manager is an interesting layout because it does not attempt to resize components to display them all in the container. Instead it displays one at a time.

An example for this:

```
import java.awt.*;
import java.awt.event.*;
```

public class threepagesapplet extends java.applet.Applet implements MouseListener.

```
TextArea ta;
Button pgstop;
Button pg3bottom;
CardLayout Layout;
public void init( )
{
        layout = new CardLayout( );
        setLayout (layout);
```

```
        add (page1Button = new Button ("Buttonpage"), "page1Button");
        page1Button.addMouseListener (this);
        add (page2label = new label ("label page"), "page2label");
        page2label.addMouseListener (this);
        Panel panel = new Panel( );
        panel.setLayout (new BorderLayout( ));
        panel.add (page3text = new TextArea ("Compositepage"));
        page3text.addMouseListener (this);
        panel.add (page3buttom = new Button ("Bott button"), "South");
        page3bottom.addMouseListener (this);
        add (panel, "panel");
    }
    public void mouse clicked (MouseEvent e)
    {
        layout.next (this);
    }
    public void mouseEntered (MouseEvent e) { }
    public void mouseExited (MouseEvent e) { }
    public void mousePressed (MouseEvent e) { }
    public void mouseReleased (MouseEvent e) { }
```

When the user clicks on any of the components in the applet, the threepagesApplet shows the next component in its layout. This is done by calling the next( ) method on the Layout.

- first (Container)
- next (Container)
- previous (Container)
- last (Container)

layout.show (this, "page2label") : jumps directly to this page.

Student Activity 2

1. List various standard layout managers.
2. Describe flowlayout.
3. Describe gridlayout.
4. Describe borderlayout.
5. Describe cardlayout.

22.4 MENUS

Menus are one of the primary User Interface constructs in almost any windowed application. Each user created new window can have its own menu bar at the top of the screen. A menu bar can have number of drop down menus which in turn can have menu items or submenus. The most basic menu in an application consists of three main elements : menubar, popups and menu items. Java also supports some advanced menu features such as separators, disabled menu items, checked menuitems and submenus.

Steps to Create a Menu:

1. Create an instance of the MenuBar Class.

```
Menubar mb = new Menubar ( );
```

2. Attach it to the Frame using the setMenuBar( ).

```
frame.setMenuBar (mb);
```

3. To create a popup create an instance of Menu class.

```
Menu m = new Menu ("background");
```

4. Add the pop up to the menu bar using add( ).

```
mb.add(m);
```

5. Create and add menu items to the pop up using add( ) and MenuItem( ).

```
m.add (new MenuItem ("default"));
```

6. To have separators, use addSeparator( ).
7. To disable or enable any MenuItem use disable( ) or enable( ).

Example :

```
import java.applet.Applet;
import java.awt.*;
public class lays extends Frame
{
        Menubar mb = new MenuBar( );
        Menu main = new Menu ("Main");
        Menu quit = new Menu ("Quit");
        MenuItem thebwitem = new MenuItem ("BW");
        MenuItem theColoritem = new MenuItem ("Color");
        MenuItem thereditem = new MenuItem ("Red");
        MenuItem theblueitem = new MenuItem ("BLUE");
        MenuItem theexititem = new MenuItem ("Exit");
        public static void main (String args [ ])
        {
                lays theapp = new lays ( );
        }
        public lays( )
        {
                super ("A Menu example");
                setMenubar (mb Bar);
                mb.add (main);
                add (quit);
                main.add (thebwitem);
                main.add (thecoloritem);
                main.addSeparator( );
                main.add (thereditem);
                main.add (theblueitem);
                quit.add (theexititem);
                coloritem.setEnabled (false);
```

```
                setSize (320, 240);
                show( );
        }
}
```

22.5 DIALOG CLASS

Dialogs are separate popup windows that accept input from the user. Unlike frames they have several restrictions such as,

- They cannot have menus or be resized.
- They must be opened by a parent window in order to exit.

Since dialogs are designed for user input, they can be designated either as modal or non modal at creation time. Modal dialogs are to be explicitly closed as they do not allow simultaneous navigation in two different windows whereas nonmodal dialogs do.

Generic Dialogs

Dialog (Frame, string, boolean);

This creates an initially invisible dialog attached to the current frame which is either modal (true) or non modal (false).

FileDialog

File Dialog provides a basic file open/save dialog box that enables accessibility to the file system. The file dialog is system independent but depending on the platform, the standard Open/Save file dialog is brought up.

Steps to Create FileDialog:

1. To Create a file Dialog use the following constructor:

```
FileDialog (Frame, String, int);
```

2. The int argument specifies the mode i.e., either FileDialog.LOAD for file open or FileDialog. SAVE for file save.
3. Use the show( ) method to display it.

Example :

```
import java.awt.*;
import java.applet.Applet.*;
public class lays extends Applet
{
        public void init( )
        {
        myframe frame = new myframe ("my first window");
        frame.resize (200, 200);
        frame.init( );
        frame.show( );
        }
}
class myframe extends Frame
myframe (String s)
{super (s); }
public void init ( )
{
```

```
        Dialog dl = new Dialog (this, "first", true);
        dl.setLayout (new FlowLayout (FlowLayout.CENTER));
        TextField tf = new TextField (30);
        dl.add (tf);
        dl.add (new Button ("OK"));
        dl.resize (200, 175);
        dl.show( );
    }
    }
```

22.6 OTHER CONTROLS

- ComboBox—it is a pull down list component.

  methods >> getSelectedItem( ), getSelectedIndex( ), getSelectedObjects( )

- Slider >> Allows GUI users to enter numerical values that can range continuously from a minimum value to a maximum value. There are two sliders in Java – vertical slider and horizontal slider.

Student Activity 3

1. What are Menus ?
2. Write a step-wise procedure to create menus.
3. What are dialogs ? How are they different from frames ?
4. What is a filedialog ? How will you create them ?
5. How will you create combobox ?

22.7 SUMMARY

- User Interface refers to communication between a program and its users.
- The development of a Graphical User Interface is supported by basic UI Components – Layout Manager, containers and events.
- The package java.awt provides a set of basic UI components such as Labels, Textfields and Checkboxes.
- Components can be created using their constructor methods and added to the desired window using the add( ) method.
- The layout managers control the placement of components and ensure consistency of the screen's appearance across different platforms. The default layout is "FlowLayout".
- Containers are placeholders for components. They include frames, windows and dialogs. Unlike frames, dialogs cannot have menus or be resized and they must be opened by a parent window.
- Menus are primary UI construct in any standalone application. They are an instance of the menu class. They consist of a menubar which can further have a number of drop-down menus.

22.8 KEYWORDS

Layout Manager : Controls the appearance of the display.

Container : Generic AWT components that can contain other components including containers.

Label : Text string that can be used to label Other User Interface components.

TextField : A subclass of the Text component class which can hold a single row of text.

TextArea : A subclass of the Textcomponent class which is a multiline input area including scrollbars.

Scroll Bar : Used to select continuous values between a specified minimum and maximum.

Dialog : Separate popup window that accepts input from the user.

ComboBox : A pull down list component.

22.9 REVIEW QUESTIONS

1. Fill in the blanks:
   a. Labels, TextFields, Menus etc., are examples of _____ components.
   b. The default Layout Manager for an applet is _____.
   c. _____ User Interface component can be used to enter multiple lines of text.
   d. The _____ class divides the screen into equal sized rows and columns.
   e. Exclusive checkboxes are also known as _____ .
2. State whether the following are true or false:
   a. The Windows class is an abstract class.
   b. User Interface components can be added to the desired window using the add( ) method.
   c. Buttons can be labeled using the label component only.
   d. Exclusive checkboxes are also known as radio buttons.
3. Write an applet program to generate a form in a new window that accepts the following data from the user : name, parent's name and address. The user can select his age group from under-20, between 20 and 50 and above 50 years and select working or non working from a Checklist. Add buttons to open and close the window.
4. Create an applet containing three buttons – red, green and blue. The background color is initially set to white. Depending on the button pressed, the background color should change accordingly. If the user presses the button labeled red, the background color should change to red.

Answers to Review Questions :

1. (a) User Interface (b) Flow Layout (c) TextArea (d) Grid Layout (e) Radio buttons
2. (a) True (b) True (c) True (d) True

22.10 FURTHER READINGS

Herbert Schildt ; *Java: The Complete Reference, J2SE TM* ; 2005, Mc Graw-Hill Professional.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT

23

IMAGES

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe File formats
- Describe Image fundamentals
- Define Image observer
- Describe Media Tracker

U N I T S T R U C T U R E

23.1 INTRODUCTION

Java has extensive image manipulation features. The Abstract Window Toolkit (AWT) provides support for applets, windows, fonts, graphics, GUI components and imaging system and animation. In this unit you will examine the AWT's image class and the java.awt.Image package. Together, they provide support for imaging i.e., processing and manipulation of graphical images. The essence of all animation is moving pictures, – a sequence of still images displayed at a fast enough rate to give a moving effect. Movies use a frame rate of 24 frames per second. Normal televisions use 50 Hz frequency. Faster update rates mean less image flickering and consequently more realistic looking animation.

In this unit you will learn to create Image Object, get the data in the image object and to display image object on the screen. You will use image observer which can perform actions such as indicating progress to download. You will also learn a technique to eliminate flickering in the animation which will lead to exploration of the complex Image AWT subsystem.

23.2 FILE FORMATS

Web browser can display image files in GIF and JPEG. GIF format means Graphics Interchange Format. Originally, web images could only be in GIF format. Which was created by CompuServe in 1987. GIF images can have only up to 256 colours each. JPEG format was created by a group of photographic experts in 1995. It stands for Joint Photographic Expert Group. JPEG stores full

colour spectrum. JPEG formats are more highly compressed than GIF format. Most Browsers also support animated and transparent GIF image also known as GIF89a format. The Java image classes abstract the differences behind the formats. All of the above formats are supported by core API. In future, other graphics formats will be supported via Extension APIs.

23.3 IMAGE FUNDAMENTALS

Java supports an Image class, along with other classes for the purpose of animation and creation of large sequence of video frames. In Java, the image class is used to create, load and display images.

Image class encapsulates a platform independent image structure. This approach shields you from the profusion of hardware or software dependence. The methods provided by class Image allow you to query the image dimensions, to know image's properties such as image format, copyright information and so on. The java.awt.Image class is abstract so you cannot create Image instances from it. This package is only devoted to image processing. Other classes must create instances of image for you. Methods that construct and return objects are called factory methods.

There are three common operations that occur when you work with images : creating an image, loading an image and displaying an image. In Java, image class is used to refer to images in memory and to images that must be loaded from external sources. Thus, Java provides ways to create new image object and ways to load one, and also ways to display them.

Creating an Image

The image class does not have enough information about its environment to create proper data format for the screen. The component class in java.awt is used to create the image. Component class has a factory method called createImage( ).

Syntax is

Image createImage (int width, int height); Image createImage (ImageProducer imgProd). This command returns an image produced by imgProd which is an object of a class that implements the ImageProducer interface.

```
Canvas c = new Canvas( )

Image test = c.createImage (200, 100)
```

This creates a blank image of specified size. Later, data can be written on to it.

Loading an Image

The other way to obtain image is to load one. To do this, Use getImage( ) method defined by Applet class.

```
Image getImage (URL url, [String imageName])
```

The above method returns a image object that encapsulates the image found at the location specified by url and having the name specified by ImageName.

Displaying an Image

Once you have an image object, you can use drawImage( ) method defined by Graphics class to display it.

```
boolean drawImage (Image imgobj, int left, int top, ImageObserver imgOb)
```

This displays the image passed in imgobj with its upper left corner specified by left and top.

Example :

```
import java.awt.*;
        import java.applet.*;
        public class SimpleImageLoad extends Applet
        {
                Image img;
                public void init( )
        {
        img = getImage (getDocumentBase( ),        "ABC.GIF");
        }
        public void point (Graphics g)
        {
                g.drawImage (img, 0, 0, this);
        }
}
```

The image ABC.GIF will be loaded from a URL that is relative to the result of getDocumentBase( ) which is the URL of the HTML page this applet tag was in Image observer calls the paint( ) method every time more image data arrives.

Student Activity 1

1. Describe various imagefile formats.
2. What is the role of Image class ?
3. How will you create an Image ?
4. How will you load an Image ?
5. How will you display an object ?

23.4 IMAGE OBSERVER

Image observer is an interface used to receive notification as an image is being generated. This interface provides for notifying a component about changes in image information. This is necessary because Java can accomplish image loading as a background process that may not have a complete image ready when a component is ready to be painted. The method that is required by this interface is image Update; this method is called by the process loading an image to report status changes. It defines only one method : imageUpdate( ).

```
boolean imageUpdate (Image imgObj, int flags, int left, int top, int width,
int height)
```

Where, imgObj is the image to be loaded–flags is an integer which specifies the status of the update report and remaining four integers, left, top, width, height, represent a rectangle that contains different values depending on the value passed in flags. imageUpdate( ) method should return false if it has completed loading and true if there are more images to process.

Different flags:

WIDTH	:	The width parameter is valid and contains the width of the image.
HEIGHT	:	The height parameter is valid and contains the height of the image.

PROPERTIES	:	Properties associated with image can now be obtained using imgObj.getProperty( ).
SOMEBITS	:	More pixels needed to draw the image.
FRAMEBITS	:	A complete frame that is part of multiframe image.
ALLBITS	:	The image is now complete.
ERROR	:	The image is incomplete and cannot be displayed.
ABORT	:	An image that was being tracked asynchronously was aborted before it was complete. However, if an error has not occurred, accessing any part of the image's data will restart the production of the image.

Example of imageUpdate( ):

```
public boolean imageUpdate (Image img, int flags, int x, int y, int
w, int h)
{
        if ((flags & ALLBITS) == 0)
        {
        System.out.println ("Still processing the image');
        return true;
        }
        else
        {
        System.out.println ("Done processing the image");
        return false;
        }
}
```

Image observer can be used to load an image without flicker. The default implementation of imageUpdate( ) repaints entire image each time any new data arrives. This causes flashing between the background colour and image. repaint( ) method causes the system to only repaint the image every tenth of a second or so. This causes a jerky feel as the image is painting. Default implementation knows nothing about images that may fail to load properly. To overcome the above three problems, the following method can be used to load the image without flickering, without jerk and finally it handles the error caused by the desired file not being found by examining the flags parameter for ABORT bit.

Example :

```
import java.awt.*;
import java.applet.*;
public class ImageObserveApp extends Applet
{
        Image img;
        boolean error = false;
        String imgname;
        public void init( )
        {
                SetBackground (color.blue);
```

```
                imgname = "UPTECLOGO.jpg";
                img = getImage (getDocumentBase( ), imgname);
        }
        public void paint (Graphics g)
        {
                if (error)
                {
                        Dimension d = getSize( );
                        g.SetColor (Color.red);
                        g.fillRect (0, 0, d.width, d.height);
                        g.SetColor (Color.black);
                        g.drawString ("Image not found" + imgname, 10,
100);
                }
        else
        {
                g.drawImage (img, 0, 0, this);
        }
}
public void update (Graphics g)
{
        paint (g);
}
public boolean imageUpdate (Image img, int flags, int x, iny y, inw
w, int h)
{
        if (( flags & SOMEBITS) ! = 0)   // new partial data
        {
                repaint (x, y, w, h);   // new partial pixels
        }
        else if ((flags & ABORT) ! = 0)
        {
                error = true;   // file not found
                repaint ( );
        } // paint whole applet
        return (flags & (ALLBITS 1 ABORT)) = = 0;
        }
}
```

Double Buffering

Use of an offscreen image to reduce flickering is called double buffering. This allows you to render any image, including text and graphics, to an off screen buffer that you can display at a later time. The advantage is that the image is seen when it is complete. Double buffering is used

to reduce flickering while drawing a complicated image. It is called double buffering because the screen is considered a buffer for pixels and the offscreen image is the second buffer, where you can prepare pixels for display.

Example :

```
import java.awt.*
import java.awt.event.*;
import java.applet.*;
public class DoubleBuffer extends Applet
{
        int gap = 3;
        int mx, my;
        boolean flicker = true;
        Image buffer = null;
        int w, h;
        public void init( )
        {
                Dimension d = getSize( );
                w = d.width;
                h = d.height;
                buffer = createImage (w, h);
addMouseMotionListener (new MouseMotionAdapter( )
{
        public void mouseDragged (MouseEvent me)
        {
                mx = me.getX( );
                my = me.getY( );
                flicker = false;
                repaint( );
        }
        public void mouseMoved (MouseEvent me)
        {
                mx = me.getX( );
                my = me.getY( );
                flicker = true;
                repaint( );
        }
        });
}
public void paint (Graphics g)
{
```

```
        Graphics screengc = null;
        if (!flicker)
        {
                screeng = g;
                g = buffer.getGraphics( );
        }
        g.setColor (Color.blue);
        g.fillRect (0, 0, w, h);
        g.setColor (Color.red);
        for (int i = 0; i < w; i+ = gap)
        g.drawLine (i, 0, w-i, h);
        for (int i = 0; i < h; i+ = gap)
        g.drawLine (0, i, w, h-i);
        g.setColor (Color.black);
        g.drawString ("Press mouse button", 10, h/2);
        g.fillOval (mx-gap, my-gap, gap *2+1, gap*2+1);
        if (!flicker)
        {
                screengc.drawImage (buffer, 0, 0, null);
        }
}
public void update (Graphics g)
{
paint (g);
}
}
```

The above applet will show different pictures when mouse is moved with/without pressing the button. When a mouse button is pressed, image is always complete and clean due to double buffering. The technique can be summarized as follows:

- Do all drawing off screen.
- Copy the finished product to screen.
- Repeat previous steps as quickly as possible.

23.5 MEDIATRACKER

The MediaTracker class keeps track of images loaded from server. To load multiple images synchronously, and without having to worry about imageUpdate( ), Sun Microsystems added a class called MediaTracker in java.awt. This is an object that will check the status of an arbitrary number of images in parallel.

MediaTracker defines addImage( ) method.

void addImage (Image imgObj, int imgID, int width, int height)

imgObj is the image being tracked.

imgID is identification number.

width, height specify the dimensions of the object when it is displayed.

To check the status of an image, call checkID( ) method.

```
boolean checkID (int imgID)
```

The method returns true when all images that have the specified ID have been loaded.

Example :

```
import java.util.*;
import java.applet.*;
import java.awt.*;
public class TrackedImageApp Extends Applet implements Runnable
{
        MediaTracker tracker;
        int tracked;
        int curing = 0;
        Thread t;
        static final int max = 10;
        Image img[ ] = new Image [max];
        String name[ ] = new String [max];
        boolean stopflag;
        public void init( )
        {
        tracker = new MediaTracker (this);
        StringTokenizer st = new StringTokenizer
        ("UPTEC+LOGIN+ORACLE+NCSE", "+");
        while (st.hasMoreTokens( ) && tracked < = max)
        {
        name [tracked] = st.nextToken( );
        img [tracked] = getImage (getDocumentBase(), name
[tracked]+".jpg");
        tracker.addImage (img [tracked], tracked);
        tracked++;
        }
}
public void paint (Graphics g)
{
        int document = 0;
        for (int i = 0; i<tracked; i++)
        if (tracker.checkID (i, true))
        document + +;
}
}
If (document = = tracked)
```

```
{
        Image i = img [curimg ++];
        g.drawImage (i, 10, 10, null);
}
}
public void start( )
{
        motor = new Thread (this);
        stopflag = false;
        motor.start( );
}
        public void stop( )
{
        stopflag = true;
}
public void run( )
{
        motor.setPriority (Thread.MIN_PRIORITY);
        while (true)
        {
                repaint( );
                try
                {
                        Thread.sleep (100);
                }
                catch (InterruptedException e)
                {
                };
                if (stopflag)
                return;
                }
        }
}
```

This example creates a new MediaTracker in init( ) method and then adds each of the named images. In paint( ) method all the images are loaded and displayed.

Student Activity 2

1. Define Image Observer.
2. Describe the method available in Image Observer.
3. What are the features of Image Observer.
4. Define double Suffering.
5. What is media Tracker ? Describe its methods.

23.6 SUMMARY

- The key requirements for any good animation are, absence of flickering and high frame rates.
- Web images can be of GIF and JPEG formats. JPEG images can be of higher fidelity as well as more highly compressed than, GIF images.
- CreateImage( ) method of "Component" class is used to create image object. Use getImage( ) method defined in Applet class to load the image specified by URL. Finally, display it with drawImage( ) method of Graphics class to display it on the screen.
- ImageObserver Interface is useful when loading an image from network to display progress of downloads.
- Double buffering is one of the most basic computer graphics techniques for smooth animation. The technique can be summarized as follows :
- Do all your drawing off-screen.
- Copy the finished product to the screen.
- Repeat the previous two steps as quickly as possible.
- MediaTracker class keeps track of images loaded from server.

23.7 KEYWORDS

Image Class : Encapsulates a platform independent image structure.

Image Observer : An Interface used to receive notification as an image is being generated.

Double Buffering : Use of an off screen image to reduce flickering.

MediaTracker : A class that keeps track of images loaded from server.

23.8 REVIEW QUESTIONS

1. How do you display an image in Java application and in Java applet ?
2. What is the use of ImageObserver Interface ?
3. What are the advantages of double buffering ?
4. How does MediaTracker work ?
5. What is the significance of flags in imageUpdate( ) method of ImageObserver interface ?
6. Write a program to create an applet which will display Mc Donald's logo on the screen.
7. In the above program, also display the loading status in the status bar using image observer.
8. Fill in the blanks :
   a. Web Browser can display image files in ..... and ..... formats.
   b. ..... encapsulates a platform independent Image Structure.
   c. ..... can be used to load an image without flicker.
   d. ..... is a buffer for pixels and ..... is the second buffer.

Answers to Review Questions :

8. (a) CIF, JPEG (b) Image class (c) Image Observer
   (d) The Screen, Offscreen Image

23.9 FURTHER READINGS

E. Balaguruswami, *Programming with Java,* Tata McGraw-Hill.

Davis, Stephen R., *Learn Java Now,* Microsoft Press.

Naughton, Patrick, *The Java Hand Book,* OSborne McGraw-Hill.

Sams.net, Java unleased

Herbert Schildt, *The Complete Reference Java 2,* Tata McGraw-Hill.

Jim Farley ; *O'Reilly, Java Distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.

UNIT

24

JDBC

L E A R N I N G O B J E C T I V E S

After completion of this unit, you should be able to

- Describe ODBC as an Industry Standard
- Describe JDBC classes and methods
- Describe various JDBC components
- Describe various JDBC methods
- Understand how to establish a session
- Understand how to execute a query
- Understand how to close the session.

U N I T S T R U C T U R E

24.1 INTRODUCTION

In the current information age, a database is the tool used to collect and manipulate data. The database forms the foundation of the infrastructure in many companies. While the database system is well suited to the storage and retrieval of data, people need some sort of visual front end application to see and use the data stored.

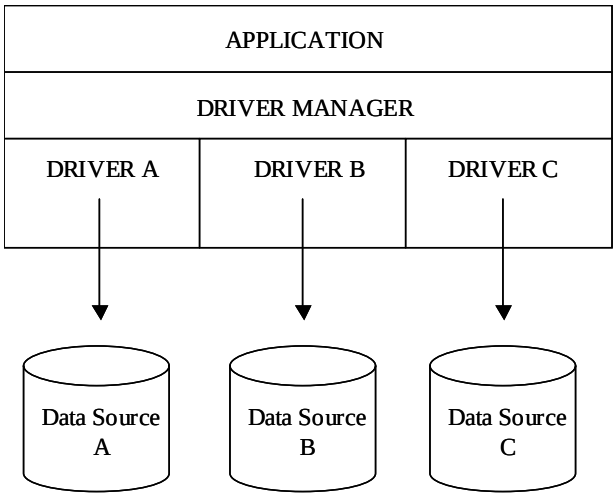
Almost 100 percent of today's enterprise applications use a database. These databases are often managed by a Relational Database Management System. Whichever database management system is used, its role in the corporate information system is predomiant.

The Java Database Connectivity (JDBC) interface allows Java applets, servlets and applications to access data in popular database management systems. The standard for accessing data is SQL which permits maximum interoperability. Of course, SQL is the language used with JDBC. JDBC is

a software layer that allows developers to write real client-server projects in Java. DDBC does not concern itself with specific DBMS functions.

An Industry Standard : ODBC

ODBC is a microsoft's implementation of a CLI (Call level Interface). It allows the programmer to develop, compile and deploy an application without targeting a specific DBMS. Modules called drivers link the application to the database of their choice. For this reason, and because it is independent of the network layer protocols, ODBC permits maximum interoperability. The DDBC mechanism is very close to the ODBC, but are adapted for Java.



The four ODBC Components

24.2 JDBC INTRODUCTION OF CLASSES AND METHODS

JDBC is not a derivative of Microsoft's Open Database Connectivety specification (ODBC). JDBC is written entirely in Java and ODBC is a C interface. While ODBC is usable by non C languages like Visual Basic, it has the inherent development risks of C, like memory leaks. However, both JDBC and ODBC are based on the X/oPen SQL Command Level Interface (CLI). Having the same conceptual base allowes work on the API to proceed quickly and makes acceptance and learning of the API easier. Sun provides a jdbc-odbc bridge that translates JDBC - ODBC. This implementation when done with native methods, is very small and efficient.

JDBC API Components

Application

The user application calls ODBC function to send SQL statements to the database and retrieve results. It performs the following tasks:

- Requests a connection with a data source.
- Sends SQL statements to the data source.
- Defines storage areas and datatypes of the result sets.
- Requests results.
- Processes errors.
- Controls transactions; requests commit or rollback operations.
- Closes the connection.

Driver Manager

Driver Manager's primary purpose is to load specific drivers on behalf of the user application.

- Perform a lookup in an ODBC configuration file or system registry to map the ODBC Data Source Name (DSN) to a specific DBMS driver.
- Process ODBC unitialization calls.
- Provide entry points and ODBC functions for each specific driver.
- Perform parameter and sequence validation for ODBC calls.

Driver

The Driver processes ODBC function calls, sends SL statements to a specific data source and returns results back to the application. When necessary, the driver translates and or/optimizes requests so that the request conforms to the syntax supported by the specific DBMS. The driver:

- establishes a connection to a data source.
- sends requests to the data source.
- performs translations when requested by the user application.
- returns results to the user application.
- formats errors in standard ODBC error codes.
- manipulates cursors if necessary.
- initiates transactions, if they are explicitly required.

There are two types of ODBC Drivers:

1. Single tier, which processes ODBC calls and SQL statements.
2. Multiple tier which processes ODBC calls and sends SQL statements to the data source.

Data Source

The data source consists of the data the user application wants to access and its associated parameters that is the type of operating system. DBMS and Network Layer (if any) are used to access the DBMS.

JDBC Components

Application

The user application invokes JDBC methods to send SQL statements to the database and retrieve results. It performs the following tasks

- Requests a connection with a data source.
- Sends SQL statements to the data source.
- Defines storage areas and data types for the result sets.
- Requests results.
- Processes errors.
- Controls transactions : requests commit or rollback segments.
- Closes the connection.

Driver Manager

Its primary purpose is to load specific drivers for the user application. It may also perform the following:

- Locate a driver for a particular database.
- Process JDBC Initialization calls.
- Provide entry points to JDBC functions for each specific driver.
- Perform parameter and sequence validation for JDBC calls.

Driver

The Driver processes JDBC methods invocations, sends SQL statements to a specific data source and returns results back to the application. When necessary the driver translates and / or optimizes requests so that the request conforms to the syntax supported by DBMS. The driver:

- establishes a connection to a data source.
- sends requests to the data source.
- performs translations when requested by the user application.
- returns results to the user application.
- formats errors in standard JDBC error code.
- manipulate cursors if necessary.
- initiates transactions if explicitly required.

JDBC Interfaces

java.sql.DriverManager A class that provides methods to load drivers and to support the creation of database connetions using methods expressed in the java.sql.Driver Interface.

java.sqlConnection Represents a particular connection on which further actions will be allowed.

java.sql.Statement Associated to a connection, it allows SQL statement to be sent to the database.

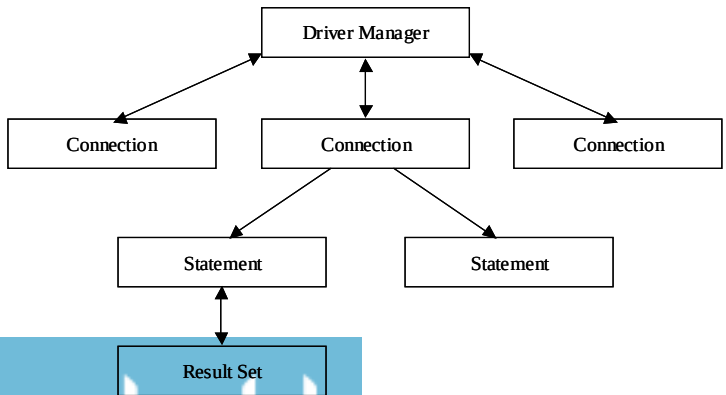
java.sql.Collable Statement It has the same role as java.sql.statement but in the context of database stored procedures.

java.sql.PreparedStatement It also has the same role as java.sql. Statement, but in the context of precompiled SQL.

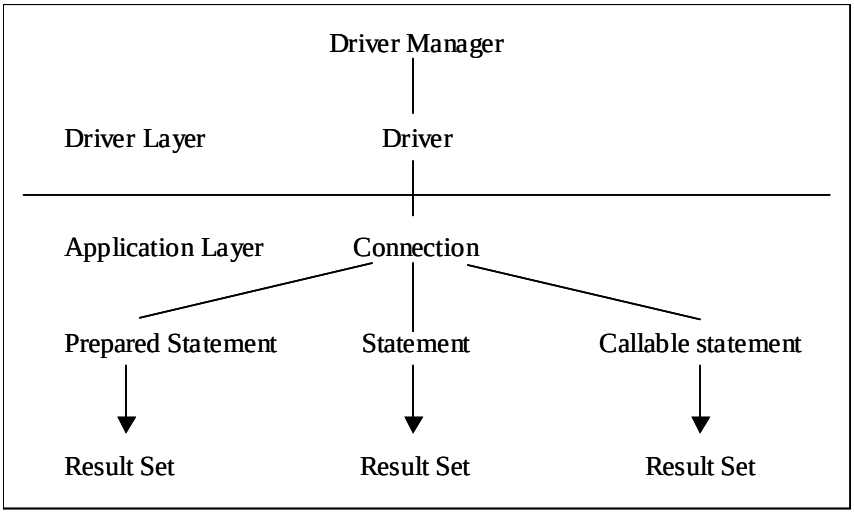
java.sql ResultSet Allows access to the rows of a previously executed statement.

java.sql ResultSetMetaData Gives information like type and properties of the columns in a resultSet.

java.sql.DatabaseMetaData Provides information about the database as a whole.



Relationshp between the main JDBC Interfaces



The **DriverManager** Class is really a utility class used to manage JDBC drivers. The class provides methods to obtain a connection through a driver, register and deregister drivers, setup logging and set login timeouts for database access.

The **DriverLayer.** There is a one to one correspondence between the database and the JDBC Driver. This approach is common in multi-tier designs. The DriverManager class, sits above the Driver and Application Layers.

The **Driver Interface.** Every JDBC program must have at least one JDBC driver implementation. The Driver Interface allows the DriverManager and JDBC Application Layers to exist independently of the particular database used. A JDBC driver is an implementation of the Driver Interface Class.

jdbc : (subprotocol> : <subname>

jdbc: Oracle : Products

The four main interfaces that every DriverLayer must implement and one that bridges the Application and Driver Layers are the Driver, Connection, Statement and ResultSet.

The Application Layer

The Application Layer encompasses three interfaces that are implemented at the Driver Layer but are used by other application developers. The three main interfaces are Connection, Statement, and ResultSet.

A connection object is obtained from the driver implementation through the DriverManager.getConnection( ) method call. Once a connection object is returned, the application developer may create a Statement object to issue against the database. The result of a statement is a ResultSet object which contains the result of the particular statement. (if any).

Statement basics is the vehicle for sending SQL queries to the database and retrieving a set of results. Statements can be SQL updates, inserts, deletes or queries. The statement interface provides a number methods designed to make the job of writing queries to the database easier.

ResultSet basics interface defines methods for accessing tables of data generated as the result of executing a statement. ResultSet column values may be accessed in any order, they are indexed and may be selected by either the name or the number (numbered from 1 to n) of the column. ResultSet maintains the position of the current row, starting with the first row of data returned.

Student Activity 1

1. Define ODBC. Name its various components.
2. What is JDBC ?
3. List varous JDBC API Components.
4. What can the following JDBC API Components do :

   (a) Application (b) Driver Manager (c) Driver (d) Data Source
5. What can the following JDBC Components do :

   (a) Application (b) Driver Manager (c) Driver
6. Describe various JDBC interfaces.
7. Define the Application layer.

Methods

Driver Interface

abstract Connection connect (String url, properties info) throws SQLException.

public abstract boolean acceptsURL (String URL) throws SQLException.

DriverManager class >> Methods

- public static synchronized Connection getConnection (String url, Properties info) throws SQLException.
- public static synchronized Connection getConnection (String url) throws SQL Exception.
- public static synchronized void registerDriver (java.sql.Driver driver) throws SQLException.
- public static synchronized void registerDriver (java.sql.Driver driver) throws SQLException.

Connection Methods

- Statement createStatement( ) throws SQLException.
- PreparedStatement prepareStatement (String sql).
- CallableStatement prepareCall (String SQL) throws SQLException.
- Void setAutoCommit (boolean autocommit) throws SQLException.
- void commit() throws SQLException.

Statement Basics Methods

ResultSet executeQuery

ResultSet executeQuery (String sql) throws SQLException.

int executeUpdate (String sql) throws SQLException.

boolean executes (String sql) throw SQLException

ResultSet getResultSet( ) throws SQLException

int getUpdateCount( ) throws SQLException

boolean getMoreResults( ) throws SQLException.

ResultSet Basics Methods

boolean next( ) throws SQLException ResultSetMetaData getMetaData( ) throws SQLException.

void Close( ) throws SQLException.

Interface DriverManager (some examples)

i.
```
jdbc.drivers = imaginary.sql.Driver: oracle.sql.

Driver : webLogic.sql.Driver

class. forName (driver);

DriverManager. registerDriver (this);

Connection con;

Con = DriverManager. getConnection ("jdbc : sybase : // dbserver :
8080 / billing", dbuser, dbpasswd);
```

where dbuser and dbpasswd is a username and password.

24.3 REGISTER DRIVER

```
jdbc.drivers = imaginary.sql.Driver:oracle.sql.Driver: webLogic.sql
Driver

Class.forName (Driver);

DriverManager.registerDriver (this);
```

24.4 ESTABLISH A SESSION

```
Connection conn;

Driver sybdriver = new SybaseDriver( );

Conn = sybDriver.connect ("jdbc : sybase:// dbserver : 8080/billing",
props);
```

24.5 EXECUTE A QUERY

```
int rowcount;

rowcount = stmt. executeUpdate (

"Delete from customer where customer id = 'MCG1023");
```

24.6 RESULTSET

```
boolean result = stmt.execute (SQLString);

int count = stmt.getUpdateCount( );

while (result || (count! = -1)) {
```

```
if (result)
{
        results = stmt.getResultSet( );
}
else if (count! = -1)
{ }
result = stmt.getMoreResults( );
count = stmt.getUpdateCount( );
}
```

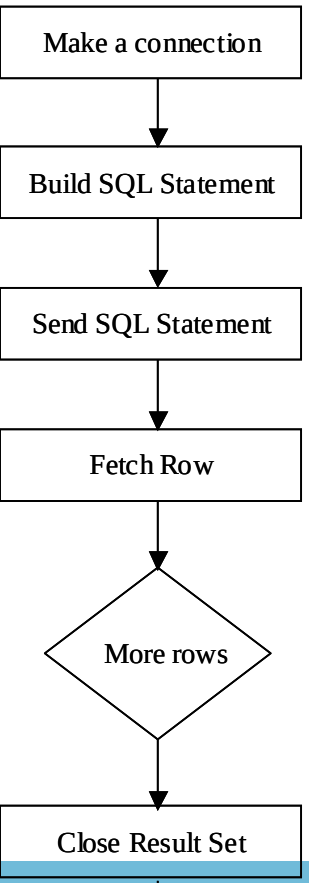


Figure 24.1 Overview of getting the result set

24.7 CLOSING THE SESSION

void close( ) throws SQLException

Normally a ResultSet is closed when another statement is executed, but it may be desirable to release the resources earlier.

```
boolean isClosed( ); // returns true if closed.

void setAutoClose (boolean autoclose);

boolean getAutoClose( );

// closing a connection

import java.sql.*;

class simple

{

        public static void main (String s[])

        {

                try

                {

                        // . . .

                        String url = "jdbc: odbc: mysource";

                        Class forName ("Sun.jdbc.odbc.jdbcOdbc Driver");

                        Connection myconn = DriverManager.
                        getConnection (url, "javauser", "hotjava");

                        // . . .

                        if (! myconn.isClosed( ))

                        myconn.close( );

                        // . . .

                        }

                        catch (java.lang.Exception ex) { }

                }

        }

}
```

Example 1 : How to execute a query

```
import java.sql.*;

class sempexample

{

        public static void main (String args[ ])

        {

                String url = "jdbc : odbc: mysource";

                try

                {

                        class.forName
                ("sun.jdbc.odbc.jdbcOedbDriver");
```

```
                DriverManager.setLogStream java.land.System.out);

                Connection myconn = DriverManager.getConnection (url,
                                        "javauser", hotjava");

                Statement mystat = myconn.createStatement( );

                ResultSet rs = mystat.executeQuery ("select name,
                id, salary from employees order by"+"salary desc");

                myconn.close( );

        }

        catch (java.lang.Exception ex)

        {

                ex.printStackTrace( );

        }

    }

}
```

Example 2: How to perform an update

```
    import java.sql.*;

    class sumpexample 1

    {

        public static void main (String args[ ])

        {

            String lurl = "jdbc: odbc: mysource";

            try

            {

                class.forName
                ("sun.jdbc.odbc.jdbcOdbcDriver");

            DriverManager.setLogStream (java.lang.System.out);

            Connection myconn=DriverManager.getConnection(url,
                                        "user", "owd");

            Statement mystat = myconn.CreateStatement( );

            int res=mystat.executeUpdate("Update"+employees set
                        Salary = salary * |.| where id = 1");

            myconn.close( );

        }

        catch (java.lang.Exception ex)

        {

            ex.printStackTrace( );

        }

    }

}
```

Student Activity 2

1. Describe the methods of the following :
    a. Driver Interface.
    b. Driver Manager Class.

c. Connection Methods.

d. statement Basics Methods.

e. ResultSet Basics Methods.

2. Give an example of Interface DriverManager.

3. How will you register a driver ?

4. How will you establish a session ?

5. How will you execute a query ? Give an example to explain :

6. How will you get the result set ?

7. How will you close the JDBC session ? Give an example to explain it.

24.8 SUMMARY

- JDBC is written entirely in Java and ODBC is a C interface.
- Some important classes for JDBC are DriverManager, Driver, PreparedStatement, Statement, Callable Statement, ResultSet.
- DriverManager Class is class to manage JDBC drivers.
- Driver Class is an interface implemented by the driver manager class.
- Statement class is the class for sending SQL queries to the database and retrieving a set of results.
- ResultSet interface defines methods for accessing tables of data generated as the result of executing a statement.
- Methods of connection, DriverManager class, Driver class statement, ResultSet are used for further processing.

24.9 KEYWORDS

Java Database Connectivity (JDBC) Interface : Allows Java applets, Servlets and applications to access data in popular database management systems.

ODBC : A microsoft's implementation of Call Level Interface (CLI) which allows the programmer to develop, compile and deploy an application without targeting a specific DBMS.

24.10 REVIEW QUESTIONS

1. State whether the following are true or false :

   a. Application layer has DriverManager, Driver Class.

   b. Statements are anything related to SQL updates, deletes, queries etc.

   c. Establishing a session is done through connection object.

   d. ResultSet class is used for storing results.

   e. Through JDBC we can cancel different databases.

2. What do you mean by database ? How it is different in client server and single tier ?

3. Describe the terms DataSource, DriverManager, Driver, Application classes in JDBC.

4. What does ODBC stand for ? What is the different between the ODBC and JDBC processing (if any) ?

5. Create a database with the customer table and item table, then through JDBC create a program where we are storing the details of customer and trying out SQL manipulations like inserting, deleting, and simply extracting.

6. Table item in the above part needs to be updated by one more item. Do it through Java.

Answers to Review Questions :

1. (a) True (b) True (c) True (d) True (e) False

24.11 FURTHER READINGS

Jim Farley ; *O'Reilly, Java Distributed Computing.*

Robert W. Bill ; *Jython for Java Programmers* ; 2001, Sam Publishing.