CPSC 211
Introduction to Software Development

Winter 2010-2011
Term 1

Instructor

- Name:
- Office:
- Email:
- Office Hours:

Course Objectives

- When you complete this course, you will be able to:
  - move from personal software development methodologies to professional standards and practices
    - design software following standard principles and formalisms
    - create programs that interact with their environment (files etc.) and human users according to standard professional norms
    - develop effective software testing skills
  - given an API, write code that conforms to the API to perform a given task
  - identify and evaluate trade-offs in design and implementation decisions for systems of an intermediate size
  - read and write programs in Java using advanced features
    - collections, exceptions, etc.
  - extend your mental model of computation from that developed in CPSC111
    - recursion, concurrency, etc.
  - work with an existing codebase, including reading and understanding given code, and augment its functionality [in assignments]

Components & Evaluation

- Your grade in this course will be based on the following activities:
  - lab participation (5%)
  - in-class exercises/participation (5%)
  - project/assignments (25%)
  - a midterm examination (20%)
  - a final examination (45%)
- To pass this course, you must obtain a 50% overall mark and, in addition, you must:
  - **pass the assignments AND**
  - **pass the final examination.**
- **Students who fail the assignments or the final exam will be assigned, as final grade in the course, the minimum of 45% and the grade computed using the above formula.** The instructors reserve the right to modify the course grading scheme at any time.

Administation

- Main web sites for the course:
  http://www.ugrad.cs.ubc.ca/~cs211/
  - contains most course material (notes, labs, assignments, etc.)
- WebCT site for the course
  - contains bulletin board and grades
  - will be available soon
- Carefully read the course information at
  http://www.ugrad.cs.ubc.ca/~cs211/courseInfo/courseInformation.html
- Labs start on Monday
- The midterm will be on
  Tuesday, October 19 at 5:00-7:00pm.
  - Let me know by the end of this week if you have a conflict with this time.

Review: Classes, Objects, References

- A typical object oriented program consists of
  - a set of class definitions
  - a set of objects that interact with each other
- Class methods define the object's behavior (i.e. what an object can do)
- References provide a way to distinguish and access the objects
  - a reference holds the address of an object
- Computation is performed by applying methods to objects

Review: Example

```
public class Account
{
   private Customer owner;
   private double balance;
   public Account( )  { banance = 0; }
   public void setOwner(Customer c)
         { ... }
   ....
}
public class Customer
{
   private String name;
   public Customer() { ... }
   public setName(String n) { ... }
    .....
}
```

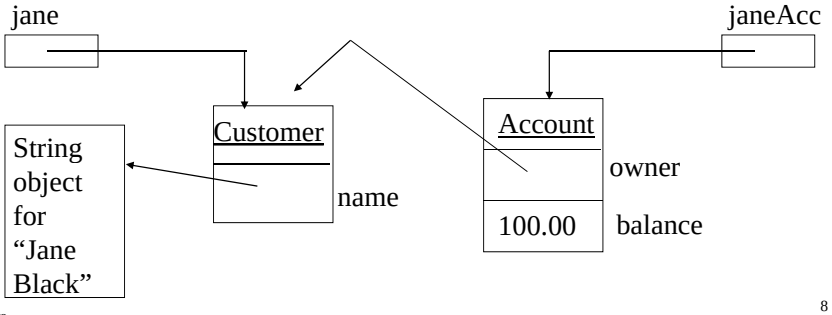
```
public class AccountTest
{
   Account janeAcc =
            new Account();
   Customer jane =
            new Customer();

   jane.setName("Jane Black")
   janeAcc.setOwner(jane);
   janeAcc.deposit( 100.00);
   .....
}
```

Find the classes, objects and references shown on this page

Review: Memory Diagrams

- Show how objects and references are stored in the computer
- Show the relationship between them.
- Are informal and used for pedagogy
- Example: A memory diagram for Jane, and her account:



Review Question 1

How many b's will this code print to the screen?

```
for (int i = 1; i <= 5; i++)
  for (int j = 0; j < 4; j=j+2)
     System.out.println("b");
```


Review Question 2

What does the following code print to the screen?

```
int a = 4;
if (a < 4)
   if (a < 1)
      System.out.println("good");
 else
   System.out.println("bad");
```


Review Question 3

Assume that Cat and Dog are subclasses of Mammal. Which of the following statements are valid?

```
a) Cat montana = new Cat();
b) Cat tuxedo = new Mammal();
c) Cat silas = new Dog();
d) Mammal animal = new Cat();
e) Mammal fluffyAnimal = new Dog();
f) animal = montana;
g) montana = fluffyAnimal;
```


Review Question 4

Consider the Counter class on the right. What is printed out by the following code?

```
Counter c1 = new Counter();
Counter c2 = new Counter();
c1.addOne();
c2.subtractOne();
c2.addOne();
c1.addOne();
System.out.println(
                c1.getCount());
System.out.println(
                c2.getCount());
```

What if we remove "static" from the declaration of count?

```
public class Counter {

   private static int count = 0;

   public void addOne() {
      count++;
   }
   public void subtractOne() {
      count--;
   }
   public int getCount() {
      return count;
   }
}
```

Review Question 5

- Write a static method `sumArray` that takes an array of ints as its only parameter and returns the sum of the values in the array.
- For example, if `sampleArray` was defines as

```
int[] sampleArray = {2, 3, 2};
```

and passed as a parameter, the method would return 7.