Informatics in Education, 2013, Vol. 12, No. 2, 153–179

Java Across Different Curricula, Courses and Countries Using a Common Pool of Teaching Material

Mirjana IVANOVIĆ[1], Zoran BUDIMAC[1], Anastas MISHEV[2], Klaus BOTHE[3], Ioan JURCA[4]

[1] *Department of Mathematics and Informatics, Faculty of Sciences University of Novi Sad, Serbia*
[2] *Institute of Informatics, Faculty of Natural Science and Mathematics University "Ss. Cyril and Methodius" in Skopje, FYR Macedonia*
[3] *Institute of Informatics, Humboldt University Berlin, Germany*
[4] *Department of Computer Science and Engineering, Faculty of Automation and Computers "Politehnica" University of Timişoara, Romania*
e-mail: {mira, zjb}@dmi.uns.ac.rs, anastas.mishev@finki.ukim.mk, bothe@informatik hu-berlin.de, ioan.jurca@cs.upt.ro

Received: July 2013

Abstract. Under the auspices of a DAAD funded educational project, a subproject devoted to different aspects of teaching the Java programming language started several years ago. The initial intention of the subproject was to attract members of the subproject to prepare some teaching materials for teaching essentials of the Java programming language. During the last two years, some advanced Java topics have been selected and appropriate teaching materials have been produced. The available pool of common teaching materials can be used in a wide range of university courses in participating countries. In this paper we share some of the results and experiences collected during the subproject that come from intensive use of the prepared teaching materials for a variety of Java topics in different countries and universities.

Keywords: course development, preparation of teaching material, common course, Java programming language.

1. Introduction

Coinciding with the introduction of the new and innovative trends in European high education, under the auspices of the "Stability Pact of South-Eastern Europe" and "DAAD – Deutscher Akademischer Austausch Dienst" (German foundation – "German Academic Exchange Service"), a project in the domain of education has been established in 2001. The main idea of the project was to create and develop common courses in several fields of computer science and to enable the use of shared teaching materials at a wide range of universities in member countries of the project consortium. In the ten years of its

existence, the project has included participants from fifteen universities, coming from nine countries: Germany, Serbia, FYR Macedonia, Bulgaria, as the core members, and Croatia, Bosnia and Herzegovina, Romania, Albania, and Montenegro as associate members. More about the project, its goals and members can be found in (Bothe *et al.*, 2009; Budimac *et al.*, 2011).

The general goal of the project has been improvement and adjustment of educational processes in South-Eastern Europe, application of the current trends, which have been already introduced and applied in the European Union countries. In the meantime, the project has been divided into several subprojects. The results of activities and efforts conducted under the subproject CTM_JOOP – "Common teaching materials on object-oriented programming using Java", which started in 2004, are presented in this paper.

The rest of the paper is organized as follows. Section 2 brings an overview of use of the Java programming language in different courses/curricula/universities. Section 3 shortly discusses different Java teaching material repositories and compares them with our approach. The structure of the common pool of Java teaching materials prepared and their distribution and use within different courses and universities members of CTM_JOOP are thoroughly discussed in Section 4. Concluding remarks about teachers' and students' satisfaction and about the teaching materials are presented in the last two sections.

2. Use of Java in Different Courses/Curricula/Universities

2.1. *Java for Teaching Programming*

During the 1970s and 1980s, Pascal was used by most educational institutions as the first programming language, but over the following years a lot of different languages have come into use. Along with these changes there has been an ongoing debate about which programming language to adopt for an introductory programming course (Bishop, 1997; Böszörményi, 1998; Duke *et al.*, 2000). From the early days of the Java programming language (King, 1997), it has been seen as the answer to the problem of choosing an appropriate language for an introductory programming course. Although some may not agree, the current consensus and trend in teaching programming is that the object-oriented languages are winners. Java has increasingly become the language of choice for teaching beginners.

In the last several years many universities, including some members of CTM_JOOP, have switched to Java in their introductory programming courses, regardless of whether they chose to start from the imperative or the object-first approach. To prepare good, illustrative, student-friendly teaching materials for Java we realized that it is important to consider the problems and difficulties students encounter, which have been reported in numerous papers (Bishop, 1997; Böszörményi, 1998; Bruce, 2004; Cooper *et al.*, 2003; Gálvez *et al.*, 2009; Jian *et al.*, 2009; Moritz and Blank, 2005). Different approaches can significantly influence ways of teaching and adoption of adequate methodology. Some advantages and disadvantages (Ivanović and Pitner, 2011) will shortly be addressed in the rest of the section.

- **Advantages –** Java possesses important advantages suitable for an introductory programming course, it is: rather simple, object-oriented, distributed, robust, secure, portable, interpreted, multithreaded, dynamic, but it is also suitable for advanced courses (this is crucial for the ideas of CTM_JOOP and intentions to prepare common teaching materials on a wide range of Java features and functionalities).
- **Disadvantages** – Some authors, on the other hand, agree that Java is not an ideal first course language (Böszörményi, 1998; Chen *et al.*, 2004; Collins, 2002; Hosch, 1996; Laakso *et al.*, 2008). They have presented a significant list of Java drawbacks. In the meanwhile, some of the drawbacks have been improved upon in new versions of the language. However, others are still present: no separation of specification from implementation, no preconditions and postconditions, "baroque" visibility rules, the fact that the exceptions not caught within a method must be declared as thrown by that method, lack of templates (Java in its latest versions supports templates but still they are not as powerful as in some other languages).

By removing some of the syntactic absurdities of C++, in the last decade Java has become the dominant language for the majority of undergraduate programming courses. This has been the case at CTM_JOOP universities too and it influenced the main objectives and the efforts in preparing the common pool of Java teaching materials spread over different courses and curricula.

2.2. *Use of Java in Different University Courses*

Java can be employed in a variety of places in a curriculum, starting from the sequence of introductory computer science courses, to an intermediate (or advanced) course in object-oriented programming as well as in many specialized courses, e.g. networking (Yang, 2003), databases (Swain *et al.*, 2002; Thomas, 2003) and software engineering (Benaya and Zur, 2007; Gendreau, 2004).

At almost every member university of CTM_JOOP, there are consistent course sequences through which students further develop their programming knowledge and skills, adopted through the introductory courses. A characteristic programming course sequence involves the following courses: Introduction to Programming, Data Structures and Algorithms, (Intermediate/Advanced) Object-Oriented Programming, Operating Systems.

At the university level, students were taught usually a few representative programming languages (paradigms) and afterwards they would be forced to use one of them as a basic tool for learning other essential concepts and parts of subsequent courses. This way students learned new topics in a particular course but at the same time they built upon their experience and knowledge of a particular programming language (Advanced networking; Benaya and Zur, 2005; Thramboulidis, 2005).

At all member universities of CTM_JOOP, after Java had been introduced during the first two years of study, it has been further used as an essential tool for the subsequent courses. According to that, we decided to concentrate our efforts in developing teaching materials not only on basic Java topics but also on advanced topics, as much as it was possible.

2.3. *Use of Java in Project Members' Courses*

Due to the wide range of advantages and disadvantages of teaching Java as a first programming language, all teachers, members of CTM_JOOP, who teach the introductory programming course, have been constantly in a dilemma whether to stay with the imperative or to switch to the object-first approach. Still some of them apply the imperative and other the object-first approach in their courses. Such diversity of approaches: either teaching the imperative paradigm (with or without Java) or the object paradigm first; either proceeding to the introduction to object-oriented programming (in the first case) or to the advanced object-oriented programming (in the second case), etc., influenced activities, preparation of materials and general results of CTM_JOOP.

All participating universities of CTM_JOOP, in their curricula for different study programs in informatics education, have had courses on introduction to programming. As an essential part of these courses, a particular programming language (Pascal, C, or Java) has been taught. All universities have used Java as a basis for the subsequent courses (see Section 4).

During the first three years of the CTM_JOOP existence, the main goal had been to create and develop common teaching, examination, and assessment materials covering the essentials of the Java programming language within beginners' courses: Introduction to Programming and Object-Oriented Programming I. The main motivation for preparation of the teaching materials was that it had to satisfy a range of requirements and had to be useful for the following approaches in teaching the introductory programming course:

- To serve as a basis for teaching Java as the first programming language in introductory programming courses for first year students.
- To use Java as the first object-oriented language (2nd or 3rd semester), for students that have already used another language (C, Pascal or Modula-2) in the introductory programming course.
- To serve as a good basis for teaching Java as the second programming language and use teaching materials as illustrations for object-oriented concepts and programming for second year students (i.e. 3rd semester students).
- As a consequence of previous requirements it was both necessary and challenging to prepare different versions of teaching materials for a variety of topics. This way, teachers could select the most suitable version of the topic and present it to students: slides, case-studies, assignments, exam questions, literature, etc.

As several teachers prepared teaching materials for the same topic according to their teaching styles, methodology and the students' pre-knowledge at the given university, we have reached soon a pool of teaching materials covering different essential topics (see Table 2).

After continuation of the project in 2008, we decided to extend our efforts and try to prepare additional teaching materials for different advanced Java topics. We concluded that for the project consortium, the most important and useful would be to cover as many Java topics that could be used in different subsequent courses as possible. Universities, members of CTM_JOOP, have plenty of courses which rely on Java as basic tool for

teaching and illustration of particular course concepts, so efforts to cover more topics would be justified.

Table 1 shows different courses that have been conducted over the last several years at partner universities within which teachers have used intensively the common pool of Java materials. Aims, prerequisites and audience of the courses are also specified.

Table 1

Use of Java in CTM_JOOP partners courses

University – course	Aims of the course and prerequisites (pre-knowledge)	Audience – Students and study programs
Humboldt University Berlin, Germany		
1. Introduction to Programming, 1st semester	To teach students the fundamentals of imperative and object-oriented programming by means of Java. There are no prerequisites.	First year students of "Bachelors curriculum Informatics".
"Politehnica" University of Timişoara, Romania		
1. Object-Oriented Programming, 3rd semester	To teach students the object-oriented paradigm. Students are familiar with programming in C, and attend in parallel a course on Data Structures and Algorithms.	"Computer and Information Technology", an engineering-oriented program.
2. Object-Oriented Programming II, 4th semester	To teach students the essential concepts of network programming using mostly Java technologies. Students are already familiar with the object-oriented programming in Java, operating systems (mainly UNIX), and computer network architecture.	"Computer and Information Technology", an engineering-oriented program.
University of Novi Sad, Serbia		
1. Object-Oriented Programming I, 3rd semester	To teach students the fundamentals of object-oriented programming. Students are familiar with imperative programming and basic data structures and algorithms.	"Business Informatics" and "Computer Science" programs.
2. Object-Oriented Programming II, 4th semester	To teach students the advanced Java features. Students are familiar with object-oriented programming and basic data structures and algorithms.	"Business Informatics" and "Computer Science" programs.
3. Operating Systems I, 5th semester	A classical course on operating systems. Students are familiar with object-oriented programming and basic data structures and algorithms. Some of them maybe are familiar with advanced OOP II course.	"Business Informatics" and "Computer Science" programs.
Ss. Cyril and Methodius University, Skopje		
1. Data Structures, 4th semester	To teach students the fundamentals of data structures and algorithms. Students are familiar with imperative and object-oriented programming in Java.	"Business Informatics" and "Computer Science" programs.
2. Network Operating Systems, 6th semester	Aim of the course is to aggregate the concepts of operating systems and computer networks and to enable students to comprehend the higher-level concepts. Students are familiar with Computer Networks, Operating Systems, basics of Object-Oriented Programming, Data Structures in Java course.	Computer Architecture and Networks program.

3. Related Work

Nowadays there are many electronic versions of educational materials in private and organized repositories (e.g. Open Educational Resources – OER, http://www.oercommons.org/community). There are also many electronic teaching resources including e-versions of different freely available books.

There is a similar situation in the domain of the Java programming language. There are different sources of e-forms of teaching materials including: books, repositories and a wide range of educational environments.

In this section we will shortly present some characteristic cases and compare them with our approach.

One of many interesting e-books on Java is a free, on-line textbook on introduction to programming, which is directed mainly towards beginner programmers, but might also be useful for experienced programmers (Eck, 2011). It is a traditionally written book containing mainly long textual explanations of Java notions and concepts. Contrary to this book, our teaching materials are prepared in forms of .ppt presentations containing essential, key concepts/notions and short effective explanations tailored to the level of knowledge of our students. Our students of course may use this and other similar books as an additional learning material.

Another kind of e-forms of Java teaching materials are repositories of .ppt presentations, more or less tightly connected to the specific courses taught at a particular university. Usually they are created to satisfy local rules and needs and are devoted only to a particular Java programming course. Probably the most organized, generalized and complete set of Java materials is The Java Tutorials by Sun/Oracle (http://docs.oracle.com/javase/tutorial/). It contains free, online practical guides for programmers who want to use the Java programming language in order to create applications. The guides include dozens of lessons and hundreds of complete, working examples. Since they were created for a general purpose, proposed lessons in majority of cases are not adequate for varied levels of teaching Java, varied courses and levels of students' pre-knowledge. All of the mentioned aspects as well as students' abilities and the specific needs and methodology within our project consortium, have been considered during developing our teaching material.

Finally, there are some specific projects/environments devoted to learning the Java programming language. Most of them are highly specialized and devoted to particular needs and aspects of learning.

For example, the Greenfoot approach (Kölling, 2010) is one of the projects created to increase motivation of beginner programmers. Greenfoot is an educational integrated development environment aimed at learning and teaching, which combines an interactive graphical output with programming in Java. It is aimed at pre-university and non-technical students but may be used also as a supplementary source for Java programming for college-level and university-level education.

Having in mind the characteristics and aims of Greenfoot it is obviously not an appropriate instrument and can not satisfy the aims and goals of CTM_JOOP.

4. Distribution and Use of Common Pool of Java Teaching Materials

Most of the teacher members of CTM_JOOP already teach Java, either in introductory programming courses (where some apply the object-first and others the imperative-first approach) or very early in the curriculum, in an object-oriented programming course. Since the students of advanced courses (e.g. Object-Oriented Programming I, Data Structures and Algorithms, Object-Oriented Programming II, Network Operating Systems, Data Bases II, Operating Systems I) have some pre-knowledge of Java, teachers were motivated to adjust their courses in order to use Java as a supporting programming instrument. They were asked, in accordance with the CTM_JOOP goals, to adjust the already available materials or to prepare new ones for their courses based on Java (WS, 2012).

4.1. *Teaching Material for Basic Java Topics*

Teaching methodology at different universities depends on a wide range of factors: students' motivation (most of them are interested only in achieving grades and not in *learning* how to program), good balance between theoretical and practical aspects of teaching, tradition of teaching at a particular university and teacher's specific teaching techniques. Teaching programming to novices is a rather responsible and difficult task. Most of students manage to be reasonably effective in adopting basic syntax and semantics, but preparing students to put the theory into practice is a very sensitive task for teachers. Diversity of teaching approaches within introductory programming courses (especially in Berlin, Novi Sad and Belgrade) among CTM_JOOP members resulted in the production of different teaching materials for the same Java topics (Ivanović *et al.*, 2010; OOJava). Together, we have been developing teaching materials in a goal-oriented manner (at the beginning independently, but in the later years together, through project workshops and meetings, by discussing materials and adjusting them). We distinguished which topics were important for our consortium and developed materials for them in several iterations. The teaching materials produced cover basic Java concepts. They are more or less intended for students who may or may not already be familiar with object-oriented programming concepts. Names and short content of basic topics are enumerated in Table 2.

Regardless of whether we appreciated the diversity of students and teachers or not, we agreed that the basic Java topics have to be introduced in a rather straightforward manner. We agreed that for methodological reasons of teaching programming concepts some logical order of teaching Java topics is necessary to be proposed and suggested. The topic dependency graph in Fig. 1 represents a logical order of teaching basic topics agreed within CTM_JOOP community. We also proposed two topics for teaching essential object-oriented concepts (T05). The main reason was that at several CTM_JOOP universities Java has been taught as the first programming language to students who are not familiar with any other object-oriented language. In this case such concepts have to be taught in a specific manner and with a lot of illustrative examples. On the other hand, at several other universities Java has been taught after a C++ course and students are already familiar with object-oriented concepts. In this case we suggested the more advanced version of the topic T05.

Table 2
Basic Java topics

Basic Topics	Contents
T01. Getting Started.	Introduction to the Java technology, Java programming environment.
T02. The Language Overview (Elements of Java).	Basic Java elements, Program structure.
T03. Primitive Data Types.	Declaring and initializing variables, Simple I/O, Operators.
T04. Statements – Control Structures.	Expressions, Java Statements (Declaration, Control flow statements).
T05. Introduction to OO Programming.	**Version1**. **Basic OO programming concepts for novices:** Objects, Classes, Built-in Java classes, Type casting, java.lang.*, java.util.*. **Version2**. **OO programming concepts in Java for students familiar with C++:** Objects, Classes, Inheritance in Java and other programming languages.
T06. Reference Data Types.	Creating new classes, Constructors, Overloading, Arrays Composition, Inheritance, Polymorphism, Interfaces, Abstract classes, Inner classes. Linked data structures.
T07. Packages.	Creating and using packages, Naming packages, Managing source and class files.
T08. Exception Handling.	Try-catch and Throw statements, User defined exception handling, User-defined generation of exceptions.
T09. JavaBeans Basics. Basic Elements of Windows and Applets.	JavaBean as a component model, Core concepts of JavaBeans, Properties, Event model, Event handling, Introspection, Bean persistence, Bean persistence in XML, JFrame, JApplet, Running applets.
T10. Quick Introduction to UML.	What is UML and why is it useful? Why should UML be combined with Java? UML diagrams: Class diagram, Collaboration diagram, Sequence diagram, Activity diagram, State diagram, and Other diagrams: use case, deployment, component, package. Several illustrative examples.
T11. Introducing SE Principles in Java Programming.	Application of SE principles on "Mouse in Maze" example: Requirements analysis, Design, Implementation and Test.

Topics are hierarchically ordered starting from the easiest one at the top level and going to more and more complicated ones at the bottom level. At each level of hierarchy there are one to three possible topics. There is no particular dependency and strict order in teaching the topics on the same level as they are not strictly interdependent. An arrow between two topics Tp (previous) and Tn (next) on different levels of hierarchy means that the topic Tn depends on the topic Tp, i.e. that some of its notions require notions from Tp. Arrows and dependences in our graph are in fact simplified forms of dependences presented in the Truc framework (Pedroni and Meyer, 2010) for object-oriented modeling of object-oriented concepts.

Teachers who do not absolutely agree with the proposed order of teaching or decide to change it for some reason can apply their own logic and style, choosing an appropriate topic from the common pool of teaching materials. So, a teacher can combine his/her own teaching materials with some topics from the common pool.

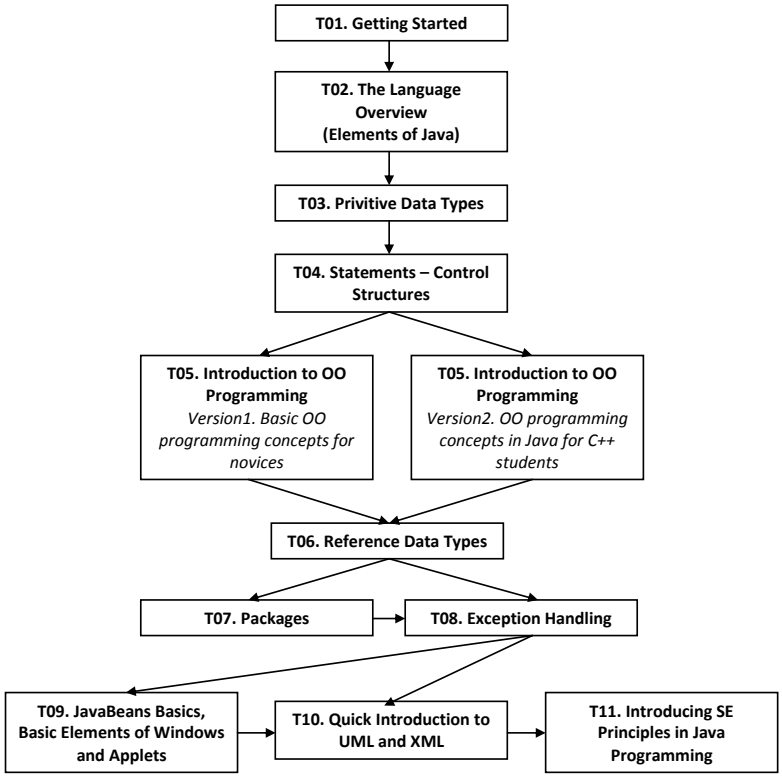

Fig. 1. Dependency graph for teaching basic Java topics.

4.2. *Teaching Materials for Advanced Java Topics*

Teachers from CTM_JOOP universities, teaching other more or less advanced courses, have had a rather easier task as they were involved only in preparation of particular teaching materials for specific courses. As the final result of our efforts, we have been supplied with a common pool of teaching materials for different advanced Java topics as well. The produced teaching materials cover many advanced Java topics suitable for use mainly in elective or higher-year courses at universities, members of CTM_JOOP. They are primarily intended for students who enroll in advanced courses to broaden their knowledge having in mind that they already must be familiar with basic Java programming concepts. Similarly to the basic topics, for the advanced topics T21-T24 different presentations are prepared, emphasizing particular aspects of the same notions and concepts.

Some possible advanced Java topics like Design Patterns; Model-View-Controller; Domain-Specific Modeling (e.g. model transformations in Java) are not considered to be included in the current course materials. The main reason is that some of them are too specific and not interesting to the broader project community; some of them are part of advanced master courses like "Design, Patterns and Architecture". Names and short content of advanced topics are presented in Table 3.

Table 3
Advanced Java topics

Advanced topic	Contents
T12. Strings.	String, StringBuilder, StringBuffer, StringTokenizer, Regular Expressions, Formatting Input, Scanning Output.
T13. Windows & Applets.	The JFrame Class, The JApplet Class, Running Applets, Drawing, Introduction to Interactive Interfaces, Layout Management, The *Swing* Event Model, Overview of *Swing* Components, Animations.
T14. Collections.	What are and Why Collections, Core Collection Interfaces, Implementations, Algorithms, Custom Implementations, Interoperability, Arrays, Containers, Generics.
T15. The Java I/O System.	What is an I/O Stream, Types of Streams, Stream Class Hierarchy, Control Flow of an I/O Operation using Streams, Byte Streams, Character Streams, Buffered Streams, Standard I/O Streams, Data Streams, Object Streams, File Class.
T16. Serialization.	What is Serialization, What is preserved when an object is serialized, *transient* keyword, Process of Serialization, Process of Deserialization, Changing the default Protocol, Creating own Protocol.
T17. Java 5 New Features.	Generics, Boxing, Varargs, Enhanced for loop, Enumerations, Static imports, Annotations, Formatting, Threading, Overriding return types, Unicode.
T18. Generics.	What are and why use Generics, Use of Generics, Generics and sub-typing, Wildcard, Type erasure, Interoperability, Creating your own generic class.
T19. Enumerated Types.	Definitions, Constant Variable Implementation, Class Implementation.
T20. Annotations.	What are and Why use annotations, How to define and use Annotations, 3 different kinds of Annotations (Marker Annotation, Single Value Annotation, Normal Annotation).
T21. Threads.	**Threads in Java:** Spawning, Joining, Priority, User interface threads. **Classical Synchronization Problems in Java:** Producer-Consumer Problem, Readers-Writers Problem Generalizations, Dining Philosophers, Semaphores, Event Counters, Bounded Semaphores, Blocking Barriers.
T22. Network Programming (TCP, UDP, URL, Socket).	**Network Programming:** Basic Networking Concepts, Client and Server Programming, IP and Java Sockets. **URL and URL Connection:** HTTP Protocol, Associated Classes, CGI, HTTP Commands.
T23. Distributed Applications.	**Client/Server Programming:** TCP Client and Server Programming, Application Level Protocols, Multithreading. **Serialization:** Object Transport, Serialization Process, Externalization. **RMI:** Architecture, Stub, Skeleton, Communication, Parameter Passing.
T24. Java Security.	**Security:** Security Models, Concepts, Policies. **Java Cryptography Architecture:** JCA/JCE, PKI, Secure Communication, Tools in the JDK.

Advanced topic	Contents
T25. Data Bases (JDBC).	What is JDBC, Step-by-Step Use of JDBC API, DataSource & Connection Pooling, Transaction, Prepared and Callable Statements.
T26. Java Internet Programming, Servlets, JSP.	Server side Programming Model, JSP, JSTL, Servlets and Containers, JSF, AJAX.
T27. Enterprise JavaBeans.	Why is Enterprise Computing so Complex, Component Models and Containers, Session Beans, Entity Beans.
T28. Mobile Agent Technology Using Java.	What is an Agent, Why Mobile Agents, Timeline, Agent Standards and Agent Systems, JADE Intro, JADE Programming Model, JADE Agent Communication.

As the pool of materials encompasses a lot of advanced Java topics, some relationship and interdependences between them have to be determined. Teachers from CTM_JOOP universities, who teach advanced programming courses and who have significant practical experience in realization of a wide range of commercial software products in Java, suggested the advanced Java topics dependency graph (see Fig. 2). An arrow between

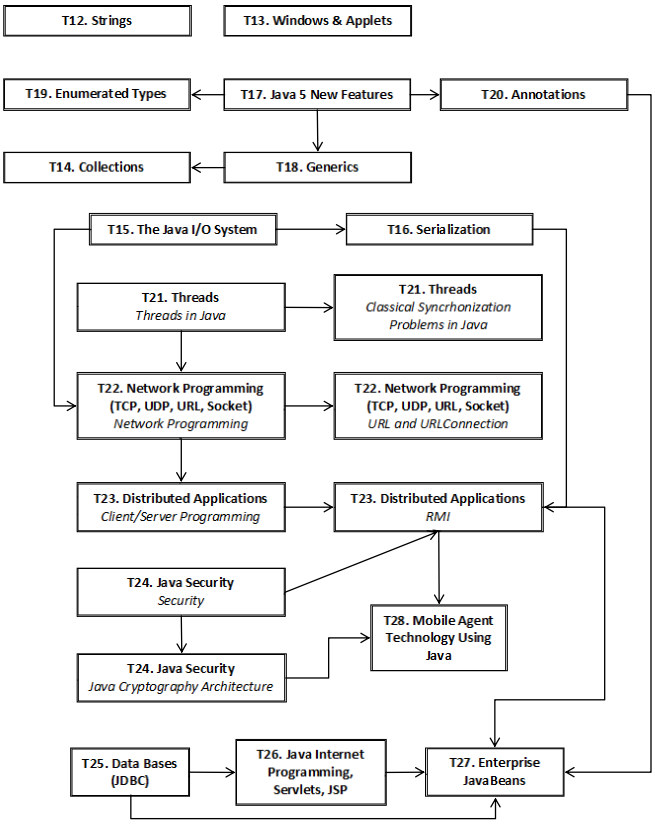

Fig. 2. Dependency graph for teaching advanced Java topics.

two topics Tp (previous) and Tn (next) means that the topic Tn depends on the topic Tp when any of its notions requires a notion from Tp. Arrows and dependences in our graph are in fact simplified forms of dependences presented in the Truc framework (Pedroni and Meyer, 2010).

Java 5 brought a number of improvements at the language level, including Enumerated Types, Generics, and Annotations. In order to put those three topics into perspective, the Java 5 New Features topic should be presented first. Furthermore, Generics should be presented before Collections, because the modern Java collection framework makes heavy use of generics. Although the given figure does not show direct dependencies of almost all topics on collections and generics, these two Java features are present in everyday use, especially in more complex applications.

Understanding of the Java I/O System and its extensive set of classes is crucial for Network Programming – sending and receiving data through sockets and the URL connection system. Similarly, because servers often accept connections in a blocking fashion, students should understand Threads before venturing into the network programming.

The topic on Distributed Applications is divided into two subtopics – Client/Server Programming and RMI. The first subtopic presents the client-server model, and it does so using the concept of TCP sockets. Although low-level, this approach is important, because all distributed communication in Java is ultimately performed through sockets. The second subtopic then introduces the remote method invocation, which is at the higher level of abstraction, and significantly simplifies the development of distributed applications.

The most efficient implementation of Mobile Agents in Java is based on RMI. Mobile agents also need to be protected from malicious attacks – e.g. during the migration process, or while communicating with other agents. Therefore, the understanding of Java Security features is beneficial.

Starting with the version 3.0, the development of Enterprise JavaBeans has been simplified by replacing external descriptors with in-code Annotations. To exploit the full potential of EJBs, students should therefore have a thorough understanding of the improved approach. Finally, remote communication with EJBs is based on RMI (and/or CORBA), so there is the interdependency between these topics.

4.3. *Possible Use of Common Teaching Materials in Different Courses and Universities*

Members of CTM_JOOP (from FYR Macedonia, Germany, Romania and Serbia) were understandably enthusiastic, eager and motivated to prepare and use the common pool of Java teaching materials. They were involved in educational processes at their universities as teachers of core Java programming courses or other advanced courses within which Java is an essential practical/implementation tool. So naturally they were highly motivated a) to participate in preparing high-quality teaching materials and b) to use materials, prepared by other colleagues, for their courses and particular subjects.

As there had been a significant level of consistency among introductory programming courses (Introduction to Programming, Object-Oriented Programming) we have been developing teaching materials together in a goal-oriented manner. We distinguished important topics for our community and developed materials for them in several iterations.

A teacher from Germany together with colleagues from Serbia translated and rearranged German teaching materials already existing for the introductory programming course in the object-first manner. The professors from Serbia translated and rearranged Serbian teaching materials already existing for the Object-Oriented Programming I course, in English. As a result, for most of the introductory topics there are at least two different presentations. This gives an opportunity to all other teachers of Java programming courses to combine their own teaching materials with some topics from the common materials according to their teaching style and students' preferences, expectations and abilities.

Distribution of efforts and credit hours for students for each course (which intensively use topics from the common pool of teaching materials) and information on how much of this budgeted effort will be used for Java-related issues where the common materials are used are summarized in Table 4.

The CTM_JOOP project has its own Web site: http://perun.pmf.uns.ac.rs/java/. The site is primarily used by the project members. The section of the site titled *Archive*, which is updated whenever a new piece of material (slides, assignments, etc.) is developed, is the most important one. Administrator for the site is a teaching assistant from Serbia.

This section is password protected so that only project members are able to download the available resources. The available materials are divided in four parts: Basic Java, Advanced Topics, Data Structures and Algorithms in Java, and Environments for Java Programming. Every part consists of a number of topics covered by slides prepared for lectures and possibly some supporting Java examples, short or longer exercises, or some other additional materials. Apart from these core resources, all in English, some additional, unsorted bits of material (for instance exam assignments) are provided in languages in which they were originally written (Serbian, Romanian, and others).

Except that crucial body of joint teaching materials, in the section *Workshop presentations*, project members can download slides and accompanying resources that were used during presentations held at the annual project workshops. This section is thus

Table 4

Courses, credit hours, use of common material

University – course	Credit hours – ECTS	Java-related issues where the common material is used.
Humboldt University Berlin, Germany		
1. Introduction to Programming, 1st semester	12 ECTS	100% use of common material
"Politehnica" University of Timişoara, Romania		
1. Object-Oriented Programming, 3rd semester	4 ECTS	75% use of common material
2. Computer Network Programming, 6th semester	4 ECTS	62.5 % use of common material
University of Novi Sad, Serbia		
1. Object-Oriented Programming I, 3rd semester	7 ECTS	100% use of common material
2. Object-Oriented Programming II, 4th semester	7 ECTS	80% use of common material
3. Operating Systems I, 5th semester	7 ECTS	20% use of common material
Ss. Cyril and Methodius University, Skopje		
1. Data Structures, 4th semester	6 ECTS	60% use of common material
2. Network Operating Systems, 6th semester	6 ECTS	35% use of common material

updated annually (soon after the end of each workshop). Apart from providing an overview of activities conducted during the course of this subproject, this part of the site serves as the source of valuable additional materials that can be used at some of the institutions participating in the subproject either for teaching purposes or as guidelines for conducting courses on object-oriented programming, tutorials on using various tools and methodologies, etc.

In the rest of this section, a brief description of use of the available common teaching materials in different courses and at different universities from several countries is given. Each subsection is devoted to a particular course and describes specific aspects of the course.

4.3.1. *Introduction to Programming – Humboldt University, Berlin*

The course Introduction to Programming is a comprehensive large module in the 1st semester at the Institute of Informatics at Humboldt University Berlin, Germany. It covers lectures, exercises and hands-on learning for the beginners in the Bachelors curriculum Informatics. The idea is to teach them the fundamentals in imperative and object-oriented programming by means of Java.

The diversity of students' pre-knowledge is a hard challenge for the staff. At secondary school, students receive rather different qualifications concerning programming fundamentals and abilities. There is the scale from 'not any' to 'very much' (Fig. 3) novices enrolled to Institute of Informatics claim to have in terms of programming knowledge gained in secondary school. Apart from general prerequisites in programming, there is also a wide diversity of programming languages that beginners gained experience with before the studies: Basic, Pascal, Delphi, C, C++ and Java. Thus, some of them already know what object-oriented programming is, some of them possess abilities in imperative programming, and finally, a large part of them has to start from scratch. The same stands for other universities, members of CTM_JOOP.

The intention in this course is to give the students with less pre-knowledge the chance to be able to follow the lectures, and on the other hand, to present the subject matter in an expeditious way. Besides pure programming, the course also introduces the main ideas of software development, e.g. basic software life cycle models, some UML diagrams and software quality issues. Topics from the common teaching materials delivered in the course are: T01–T06, T08, T11, T13, T18, and T21.

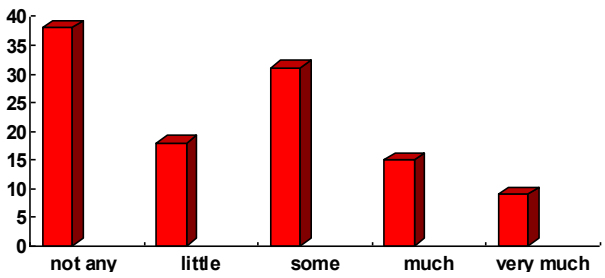

Fig. 3. 2011 year, Questionnaire: Did you get programming skills before your study?

The whole content of the course is structured into 3 parts: I – Fundamentals (general topics not related to Java); II – Concepts of imperative programming languages (T1–T4); III – Concepts of object-oriented programming languages (T5 and onwards). About two thirds of the whole course is devoted to object-orientation in part III.

As a support, tutorial groups specialized for students with little pre-knowledge are introduced. The students with some pre-knowledge in programming view the course contents as a valuable systematization and extension of their existing knowledge. They are active participants in the lectures. The rest of the students in that group, however, would like to have a faster delivery of course contents which are not possible for the average student. Since the teaching materials are placed on the course website, some of the better students decide not to take part in the lectures.

4.3.2. *Object-Oriented Programming, "Politehnica" University of Timişoara*

Object-Oriented Programming (OOP) is a core course for second year students, 3rd semester, at the Department of Computer Science and Engineering, Faculty of Automation and Computers of the "Politehnica" University of Timişoara, Romania. Students enrolled this course are already familiar with programming in C, and attend in parallel a course on Data Structures and Algorithms in C. The delivery of the course is based on a weekly schedule of 2.5 hours of lectures and 2 hours of lab exercises.

OOP is the first course in which students encounter the object-oriented paradigm and thus the lectures start with a motivation of the new paradigm and a presentation of its main characteristics in a rather theoretical way. Elements of topics T10 and T11 are used in these lectures. Afterwards, the lectures cover the main OOP concepts as found in Java: classes and objects, inheritance and polymorphism, interfaces, nested classes and interfaces, exceptions and assertions, generics, enumerated types, packages and the collections framework (topics T05–T08, T12, T14, T15, T17–T19). The last part of the semester is dedicated to the more advanced topic of concurrent programming (using topic T21), and to an introduction to graphical user interfaces. Almost all lectures contain small Java programs which are executed in front of the students and discussed with them. Lab exercises are generally synchronized with the lecture topics for each week. It might appear that the contents of this course are rather ambitious, but most of the students are doing well in lab exercises.

Polymorphism is a concept not easily understood by students, so it is required to increase the number of exercises dealing with it. Students especially like the lectures on concurrent programming, feeling that threads are useful especially in the context of multicore processors, but have some difficulty in grasping the new concepts introduced in Java 5. The GUI part of the course is perceived as rather complicated, in spite of the obvious interest for event-driven programming.

4.3.3. *Object-Oriented Programming I, University of Novi Sad*

Object-Oriented Programming I is a core course, for second year students, 3rd semester, at the Faculty of Sciences, University of Novi Sad, Serbia. Students who enroll this course are already familiar with imperative programming and basic data structures and algorithms. Nevertheless, the experience of over 8 years of delivering this course tells us that our students are not still ready for more advanced Java topics. It might seem somewhat

strange having in mind the ambitious introductory programming course for first year students in Berlin. The main reasons can be found in the following facts:

- Educational and social policies and students' needs in the two countries are different: after finishing secondary school in Serbia, young people have no real opportunity to get an appropriate job. Most of them see studying as a better option than looking for a job (and as a way of postponing decisions). They continue their education lacking real motivation for hard/serious studying. On the other hand, young people in western countries mostly decide to continue education since that is what they really wish and they are motivated to gain high-quality knowledge.
- Year after year, quality of the pre-knowledge that students in Serbia gain in secondary school is rapidly decreasing, which is influenced by a wide spectrum of reasons, and as a consequence the introductory programming course also must be taught at a lower level.

We believe that it is more useful and convenient that most of the second year students master the essentials of Java and OO programming which they can further improve and upgrade in subsequent courses or by self-studying. Most of them in fact have had average or modest grades and success in secondary school and are neither used to hard work nor motivated to deal with advanced Java programming in the second year of study. Thus, the course covers basic topics T01–T11. Starting from the next school year, we plan to gradually introduce the essential parts covered by topic T17 to the students.

At the end of the course we give students a quick overview of the basic elements of Windows and Applets. Optional topics for the course are: T10 – Quick Introduction to UML and XML and T11 – Introducing SE Principles in Java Programming. Whether they are taught depends on students' quality and motivation (they are not too excited about them) and usually they have been skipped.

4.3.4. *Data Structures, Ss. Cyril and Methodius University, Skopje*

Data Structures is one of the fundamental programming courses in the 3rd semester, at the Institute of Informatics, Faculty of Natural Science and Mathematics, Ss. Cyril and

Table 5
Topics in the Data Structures Course

Topic	Contents
Algorithms and complexity.	Detailed and simplified model of a computer, Counting operations, Time and space complexity.
Asymptotic notation.	Asymptotic upper and lower bound, Asymptotic analysis.
Foundational data structures.	Arrays, Linked lists, Performance issues.
Abstract data types.	Abstraction, Specification, Implementation.
Stacks and queues, Lists, Hash tables, Trees, Priority queues and heaps.	Various abstract data types.
Sorting.	Abstract sorter, Sorting with comparison, Other sorting algorithms.
Search trees.	Binary search trees, AVL trees, B-trees.

Methodius University, Skopje, FYR Macedonia. The course is delivered through 2 lecture hours, 3 hours of exercises and 3 hours of lab work per week and it is the third in the sequence of programming courses. The first course in the sequence introduces basic programming concepts to the students, using C as the supporting language. The second course is in fact introduction to the Data Structures course and presents to the students the basic elements of object-oriented programming in Java. Topics T01–T08 from the common pool of materials are used in this course. The course Data Structures continues the use of Java as the language for practical exercises. The additional topics covered by the course are listed in the Table 5.

From the Java point of view, the course introduces several advanced language features, with generics being the most notable (T18 from the common pool). Design patterns, covered in the T14 and T17, are also presented.

4.3.5. *Object-Oriented Programming II, University of Novi Sad*

Object-Oriented Programming II, is an elective course, for second year students, 4th semester, at the Faculty of Sciences, University of Novi Sad, Serbia. Students who enroll this course have already completed the Object-Oriented Programming I course and two courses on Data Structures and Algorithms (I and II). This course has been designed recently and we have not delivered it yet, so we do not have any experience on whether the topics are properly and consistently selected. However, looking through different advanced Java programming courses available on the Internet, we can see that most of them have a more or less similar structure. Large scale program design and implementation issues will be covered, using the Java API, the Java Abstract Windowing Toolkit and the Java Collections Framework. Generally, topics include data and procedural abstraction, generics, collection interfaces and implementations, the event-driven model of computation, GUI building using Swing, streams and files. The aims of the course are: to provide deeper insights into object-oriented programming techniques; to let students practice a component-based approach to large program design; to introduce the key aspects of the Java API and Swing; to establish a consistent programming style in Java.

Apart from topics T12–T21, introductory parts of topics T22 and T23 from the common pool of Java teaching materials could be considered to be included in the course. Topics: T22 – Network Programming (TCP, UDP, URL, Socket) and T23 – Distributed Applications, are rather demanding and require that students have additional knowledge and understanding of operating systems and networking concepts. According to that, these topics are planned to be an optional part of the course. Basic networking concepts, and basic client and server programming concepts could be presented only if students have appropriate skills and pre-knowledge and are able to cope with such advanced concepts. On the other hand students will have an opportunity to study the topics they missed later, in some other advanced courses (i.e. Operating Systems, Elective Seminars).

4.3.6. *Operating Systems I, University of Novi Sad*

Another course for which some selected topics from the common pool of teaching materials could be used is Operating Systems I, a core course for third year students, 5th semester, at the Faculty of Sciences, University of Novi Sad, Serbia. Students who enroll

this course have already completed the core course Object-oriented Programming I and some of them, probably a minority, have completed the elective course Object-Oriented Programming II. Generally speaking, as our students are not highly motivated to select demanding elective courses, such as Object-Oriented Programming II, the majority of students who enroll the Operating Systems I course will not be familiar with advanced Java features. But this knowledge could prove very useful for understanding some of necessary features in Java like the concurrency library. To understand these concepts better and to quickly adopt and use them, students should be presented with some parts of topics T14 and T18.

As a consequence of the growing importance of application servers and the excellent support in Java for thread handling, threads and locking have become topics that every computer science graduate ought to know. So, a course on operating systems can take advantage of Java's support for threads and the teacher can select appropriate topics from the common pool of teaching materials. A highly suggested topic is T21, which can serve as a core for the course. The topic T17 can be useful in presenting the modern concurrency library added in Java 1.5. The library provides an example of a well structured and powerful concurrency library that is simple to use and shows elegant solutions for common concurrency problems. As an introduction to the practical use of concurrency, the topic T23 can be used. The parts dealing with details of proper implementation of multi-threaded servers, introduction of thread pools, and use of background threads in client applications when communicating over network are some of the more important highlights of the topic.

4.3.7. *Computer Network Programming, "Politehnica" University of Timişoara, Romania*

Computer Network Programming is an elective course for third year students, 6th semester, at the Department of Computer Science and Engineering, Faculty of Automation and Computers of the "Politehnica" University of Timişoara, Romania. Students are already familiar with OOP (Java), operating systems (mainly UNIX), and computer network architecture. The course consists of 2 hours of lectures per week, and 2 hours of lab exercises per week, and is selected by approximately one third of the students.

The lectures start with a short review of the TCP/IP family of protocols, followed by the presentation of sockets API, and examples of client/server programming with sockets and UNIX system calls in C. Next, the concept of Remote Procedure Calls (RPC) is introduced and illustrated with the SUN's implementation. Almost two thirds of the course consists, however, of the techniques of Java platform for object-oriented network programming: sockets, Remote Method Invocation (RMI), servlets and JSPs, and Enterprise Java Beans (EJB). In recent years the topic of Web Services has also been introduced. The lectures use the topics T22–24, T26 and T27 of the common teaching materials, with some adaptations and additions.

For the lab exercises students are required to develop 3 small projects: the first uses sockets in C, the second is based on sockets in Java, and for the third the students can opt between developing an RMI-based application or a Web application with servlets and JSP, possibly including EJBs.

The most obvious missing part in the list of topics is JDBC, which is partially covered in a different course, and students might use it in the third project. However, the intention for the future is to cover it explicitly during lectures.

4.3.8. *Network Operating Systems, Sts. Cyril and Methodius University*
One of the advanced courses which can highly benefit from using the wide range of topics available in the common pool of teaching materials is Network Operating Systems, elective course, for third year students, 6th semester, at the Institute of Informatics, Faculty of Natural Science and Mathematics, Ss. Cyril and Methodius University, Skopje. The prerequisites for this course include courses Computer Networks, Operating Systems and Data Structures and students who take this course are familiar with basic OOP and data structures in Java.

The goal of the course is to aggregate the concepts of operating systems and computer networks and to enable the students to comprehend the higher-level concepts. Course lectures cover 4 major areas: multi-processor scheduling, network and distributed processing, distributed process management and security. The foundational topics include introduction to Java threads, synchronization and serialization. On top of them, topics like network programming, URL handling, general notion of client/server programming and remote method invocation are built up. Finally, selected topics in security are presented, using Java security and cryptographic architecture. The use of Java in this course has proven to be a good choice.

Within the course the following topics from the common pool of teaching materials are used: T16, T21–T24. The latest version of these topics includes detailed presentations, lecture notes and additional examples. Along with the presentations and examples, the materials include homework assignments and practical exams.

4.3.9. *Possibilities for Some Other Courses*
In the last several years, a lot of Web-based applications have employed databases as data repositories, and many existing database systems are adopting the Web as the interface to provide easy access to the data from the Internet. Similar to the situation in other educational institutions and countries, most of our graduate students have been starting their careers in the domain of developing Web-based applications, especially using Java and Java-based technologies. Thus, it would be very beneficial for students to gain experience in developing Web applications with database support.

During the preparation of the common pool of Java teaching materials and having in mind several advanced courses on information systems, databases and software engineering at our institutions, we thought that it could be useful to cover the necessary additional Java concepts connected to development and implementation of Web-based information systems like JDBC, servlets and JSP, Enterprise JavaBeans.

Elective courses Databases (DB) II, Information Systems (IS) II and Advanced Topics in Software Engineering (ATSE) are part of the curriculum at the University of Novi Sad, Serbia so teachers who are responsible for delivering these courses can use previously prepared presentations as basic or additional teaching materials. Within the DB II course, the teacher can use material from the topic T25, especially for the practical part of the

Table 6
The current and possible future use of common teaching materials for different courses

University – course	Topics which used within the course	Topics which could be used in future
Humboldt University Berlin, Germany		
1. Introduction to Programming, 1st semester	T01–T06, T08, T11, T13, T18, T21	
"Politehnica" University of Timişoara, Romania		
1. Object-Oriented Programming, 3rd semester	T05–T12, T14, T15, T17–T19, T21	
2. Computer Network Programming, 6th semester	T22–T24, T26, T27	
University of Novi Sad, Serbia		
1. Object-Oriented Programming I, 3rd semester	T01–T11, T17	
2. Object-Oriented Programming II, 4th semester	T12–T23	
3. Operating Systems I, 5th semester	T14, T17, T18, T21, T23	
4. Databases II		T25
5. Information Systems II		T26, T27
6. Advanced Topics in Software Engineering		T28
Ss. Cyril and Methodius University, Skopje		
1. Data Structures, 4th semester	T01–T08, T14, T17, T18	
2. Network Operating Systems, 6th semester	T16, T21–T24	

course; the topics T26 and T27 could be useful within the IS II course; while the topic T28 can be used within the ATSE course.

Other project partners will also have an opportunity in the future to re-consider the possibility of incorporation of these advanced topics in their other advanced courses.

Finally, a summary showing which topics have been used at which university so far and which topics could be used by some other courses in the near future, is given in Table 6. It is obvious that the common teaching materials developed within CTM_JOOP are highly useful and broadly exploited in a wide range of project member courses. Such large-scale use of the common topics confirms students' and teachers' satisfaction and proves quality and appropriateness of the prepared teaching materials.

5. Discussion and Experiences about the Common Teaching Materials

Members of CTM_JOOP, i.e. teachers of different courses based on the Java programming language, especially for Introductory and Object-Oriented Programming I courses, have been intensively using a number of topics from the common pool of teaching materials, for the last several years. In the meanwhile, presentations of some topics have been slightly changed and improved and some ideas for adding new topics (e.g. 3D programming) appeared. Teachers of introductory and object oriented courses based on Java are very satisfied with the common pool of teaching materials for several reasons:

- They have access to plenty of high-quality teaching Java materials prepared to support different teaching/learning styles (a kind of multi-view, multi-functional approach).

- There are several different presentations for the same Java topics, which gives them the opportunity to select the most appropriate one depending on, first of all, students' pre-knowledge and abilities (which vary from year to year). They can easily swap one presentation with another or even use different parts of different presentations for the same topic. Sometimes, another teacher's approach might allow students to understand better certain concepts of a programming language.
- Based on above mentioned facts, an exchange of experiences in teaching and using the prepared materials in different socio-cultural environments, among teachers is much easier. It brings new flavor and new ideas for further improvements and innovations in teaching materials and methodology. Thus, year after year we produce more high-quality Java teaching materials together. The joint work shortens time necessary to prepare and update the materials, compared to how it would be if there was only one teacher doing it.
- Each year, teaching assistants exchange ideas and experiences about practical tasks and problems for lab exercises. They have been enriching the common pool of tasks and problems for the practical part of the programming courses. This gives an opportunity to the teaching assistants to choose the most appropriate tasks to illustrate a particular topic, depending on the students' pre-knowledge and motivation. They can also offer additional tasks for students' homework depending on their skills:
    a. More advanced tasks for more skilled and ambitious students.
    b. Additional, simpler tasks for students who struggle to understand the presented topic.
- Generally speaking, for students who learnt basic concepts of imperative programming, it is usually difficult to understand the underlying concepts of OOP (Jian *et al.*, 2009; Madden and Chambers, 2002).

  From time to time, at the University of Novi Sad, Serbia, we use questionnaires to try to find out the reasons and difficulties in adopting Java and OO concepts. However, it is still not clear whether imperative-first approach in the introductory programming course is the main reason. Some other authors, teachers within CTM_JOOP, have tried to define a path of teaching Java specially designed for the students who have already learned an imperative programming language (Jian *et al.*, 2009). Along this path, the concepts of the Java language and the principles of OOP have to be taught step by step, gradually introducing the students into the world of Java object-oriented programming. The presented path of topics (Fig. 1) seems to completely and successfully fit teachers and students expectations over the last several years.
- Since several years ago, we decided to involve undergraduate students in project's regular annual workshops. They are usually required to prepare presentations which highlight difficulties they are facing and the expectations they have when learning particular Java subjects. Such presentations always provoke intensive discussions between teachers and students and give adequate and essential feedback for improving some parts of the existing teaching materials.

Having in mind all these advantages, teachers are highly motivated to continue to use, update, upgrade and constantly innovate the available presentations and obtain, year after year, a higher quality of the common pool of teaching materials.

Correspondingly, students' satisfaction is manifold:

- They consider the pool of e-forms of teaching materials valuable and useful. Most of them are satisfied that the teaching materials are written in English as this way they get accustomed to appropriate English terminology which is necessary for their future job and career.
- In the questionnaire used for collecting students' feedback (for Data Structures course, Ss. Cyril and Methodius University, Skopje), the course material was highly rated. Just as an illustration, almost 70% of the students were highly satisfied with the lecture presentations. Furthermore, nearly 80% of the students stated that the English language used in the presentations was not at all an obstacle. Finally, nearly 70% of the students were strongly confident about the acquired knowledge and were ready to apply it.
- As for most of other courses, for programming courses, 80% of the project universities use Moodle LMS and make teaching materials easily available for students, in the same way as at other universities (Kulji and Lines, 2005). So the students are satisfied that they can access the teaching materials from anywhere.
- Some of them are enthusiastic and proud about using internationally prepared teaching materials. They feel that this makes them a part of European Higher Education Area (EHEA) and that they can use high-quality teaching materials that give them the opportunity to continue education at some other European University.

Unfortunately, we noticed one possibly negative consequence. The existence of high-quality teaching materials available through Moodle decrease students' motivation. As a consequence they more frequently decide not to attend regular face-to-face classes. They believe that they can easily learn topics from such high-quality sources at a time that it is more convenient to them.

Teachers of other advanced courses in the curriculum gladly select and use some of the advanced Java topics. As the pool of materials encompasses a lot of Java topics, the proposed relationships and interdependences between them (Fig. 1 and Fig. 2) are useful and helpful in selecting appropriate ones. Depending on a particular course, teachers can suggest to extra motivated and skilled students to use some topics for self-studying and gaining more knowledge and skills.

6. Conclusion

During several years of existence of the subproject CTM_JOOP - "Common teaching materials on object-oriented programming using Java" under the DAAD project, the teachers of programming courses using (or which are based on) Java from Balkan countries have had a great opportunity and faced a challenge to produce a high-quality common pool of teaching materials. The existence of such materials offers numerous advantages for teach-

ers as well as for students. The prepared pool of teaching materials as well as its proposed organization is very important for all universities members of the project because:

- It covers a wide variety of Java topics including basic concepts but also a lot of advanced features.
- It can be used for different courses starting from introductory up to advanced (see Table 6).
- Different particular topics can be used in different institutions smoothly,
- It offers great opportunities for teachers to select the most appropriate among several presentations (and their parts) of the same topic, according to their style of teaching and students' affinities.
- Teachers have an opportunity, within annual workshops, to make fruitful discussions; to exchange teaching experiences and assess their teaching approaches; and as a consequence to raise their level of teaching and improve their teaching quality and competences.

The initial motivation for CTM_JOOP was to try to discuss and exchange experiences and later to try to resolve the issue of diversity of teaching approaches within introductory programming courses among consortium universities: choosing between the imperative or the object-first approach and selection of an appropriate programming language. Since at all the universities the introductory programming courses and object-oriented programming style were taught during the first two years of study, we assumed that students of both years at each university have had more or less similar abilities to cope with the object orientation. Apart from that, aims of all courses were almost the same: to teach students object-oriented programming using Java. So there were a lot of similarities between the Introduction to Programming course in Berlin (very demanding, object-first approach, first year) and the Object-Oriented Programming courses in Serbia and Romania (medium level, third semester). Accordingly, it was not too demanding to adjust teaching materials and make straightforward changes and adjustments. In order to improve the existing teaching materials available for Java programming courses we concentrated first of all on applying the adequate methodological approach. It resulted in the production of different teaching materials for the same Java topics (Ivanović *et al.*, 2010; OOJava).

- All presentations prepared for different Java topics have been used for several years in educational processes at CTM_JOOP institutions (Table 6). The following experiences gathered during this period concurred with our expectations that the materials will be well prepared and that they would be improved constantly over the following years. Different teachers worked on their particular topics according to their affinities or areas of their educational-scientific expertise.
- Through periodical meetings and workshops teachers have discussed: different teaching techniques (Vesin *et al.*, 2013), views, and opinions; students' mentality, social and cultural background in different countries; students' pre-knowledge, motivation and expectations. According to the results of these discussions, creators of the materials gained a more complex insight into the topics, and therefore a better quality of the materials has been achieved.
- To have a pool of course materials poses a challenge to teach the course in different variants over the years. As an example, it turned out that the concept *applet* is

not so fundamental in the Introduction to Programming course. At the same time, applets are rather challenging for beginners. As a consequence, we left the concept out and included other more important topics, e.g. the concept of generics.

- There are topics with a history of mutual improvements influenced by several partners. For example, Topic T11 – Mouse in Maze, has been developed in a sequence of iterations and ended up in a rather mature and stable version. Partners from Germany and Serbia contributed by ideas and new slides.

Thus, this long-lasting project and the mutual collaboration have led to tighter connections between teachers from different universities/countries. Exchange of experiences and use of common teaching materials has strengthened relationships and motivated teachers to continue their efforts and activities in order to constantly innovate, improve, and expand the existing Java pool of teaching materials.

At the moment, the pool of Java teaching materials prepared is available only for members of CTM_JOOP. The fruitful effects and results of the project and successful long-lasting use of the developed teaching materials motivate us to think about a way to make these materials visible and accessible for non-project members' colleagues/universities as well.

Acknowledgent.

Authors are partially supported by DAAD, through project "Software Engineering: Computer Science Education and Research Cooperation".

The work is partially supported by Ministry of Education, Science and Technological Development of the Republic of Serbia, through project no. III47003: "Infrastructure for Technology Enhanced Learning in Serbia".

References

Advanced networking – introduction to seminar. *Advanced Topics on Computer Networking*. (2011). http://user.informatik.uni-goettingen.de/~teleprak/SS2005/Introduction_PraktikumSS05.ppt

Benaya, T., Zur, E. (2005). Advanced Programming in Java. In: *Workshop – Teaching Methodology, Proceedings of ITICSE*. Monte de Caparica, Portugal, 348.

Benaya, T., Zur, L. (2007). Understanding threads in an advanced Java course. In: *Proceedings of ITiCSE*. Dundee, Scotland, United Kingdom, 323.

Bishop, J.M. (1997). A philosophy of teaching Java as a first teaching language. In: *Proceedings of SACLA*, 9. http://www.cs.up.ac.za/cs/jbishop/Homepage/Pubs/Tech-reports/SACLA97.pdf

Böszörményi, L. (1998). Why Java is not my favorite first-course language. *Software – Concepts & Tools*, 19(3), 141–145.

Bothe, K., Schützler, K., Budimac, Z., Ivanović, M., Putnik, Z., Stoyanov, S., Stoyanova-Doyceva, A., Zdravkova, K., Jakimovski, B., Bojić, D., Jurca, I., Kalpić, D., Çiço, B. (2009). Experience with shared teaching materials for software engineering across countries. In: *Proceedings of Informatics Education Europe IV*. Freiburg, Germany, 57–62.

Bruce, K. (2004). Controversy on how to teach CS1: a discussion on the SIGCSE-members mailing list. *In inroads – The SIGCSE Bulletin*, December.

Budimac, Z., Putnik, Z., Ivanović, M., Bothe, K., Schützler, K. (2011). On the assessment and self-assessment in a students teamwork based course on software engineering. *Computer Applications in Engineering Edu-*

cation, 19(1), 1–9.
Chen, X., Kurtonina, N., Taylor, S. (2004). First programming languages revisited. *In: Proceedings of College Teaching & Learning Conference*, Florida, 1–8.
Collins, D. (2002). Java Second – the suitability of Java as a first programming language. In: *Proceedings of The Sixth Annual Java & the Internet in the Computing Curriculum Conference*, 7.
Cooper, M., Dann, W., Pausch, R. (2003). Teaching objects-first in introductory computer science. In: *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, Nevada, USA, 191–195.
Duke, R., Salzman, E., Burmeister, J., Poon, J., Murray, L. (2000). Teaching programming to beginners choosing the language is just the first step. In: *Proceedings of ACE*, Melbourne, Australia, 79–86.
Eck, D.J. (2011). *Introduction to Programming Using Java*. Sixth Edition, Version 6.0. `http://math.hws.edu/javanotes/`
Gálvez, J., Guzmán, E., Conejo, R. (2009). A blended e-learning experience in a course of object oriented programming fundamentals. *Knowledge Based Systems,* 22(4), 279–286.
Gendreau, T.B. (2004). Teaching network programming with Java. In: *Proceedings of Midwest Instruction and Computing Symposium*, University of Minnesota. `http://www.micsymposium.org/mics_2004/Gendreau.pdf`
Hosch, F. (1996). Java as a first language: an evaluation. *ACM SIGCSE Bulletin*, 28(3), 45–50.
Ivanović, M., Pitner, T. (2011). Technology-enhanced learning for java programming – duo cum faciunt idem, non est idem. *ACM Inroads*, 2(1), 55–63.
Ivanović, M., Budimac, Z., Mišev, A., Bothe, K., Jurca, I. (2010). Teaching Java through different courses – multi-country experiences. In: *Proceedings of Conference on Computer Systems and Technologies – CompSysTech, ACM International Conference Proceeding Series 471.* Sofia, Bulgaria, 413–418.
Jian, S., Wenyong, W., Zebing, W. (2009). A teaching path for Java object oriented programming. *International Forum on Information Technology and Applications*, 3, 465–468.
King, K.N. (1997). The case for Java as a first language. In: *Proceedings of the 35th Annual ACM Southeast Conference*, 124–131.
Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education*, 10(4), article 14.
Kuljiš, J., Lines, L. (2005). Supporting the development of effective e-learning resources: a student-centered approach. In: *Proceedings of 27th International Conference on Information Technology Interfaces*, 283–288.
Laakso, M.J., Kaila, A., Rajala, T., Salakoski, T. (2008). Define and visualize your first programming language. In: *Proceedings of Eighth IEEE International Conference on Advanced Learning Technologies*, 324– 26.
Madden, M.G., Chambers, D. (2002). Evaluation of student attitudes to learning the Java language. In: *Proceedings of Principles and Practice of Programming in Java*, 125–130.
Moritz, S.H., Blank, G.D. (2005). A design-first curriculum for teaching Java in a CS1 course. *ACM SIGCSE Bulletin*, 37(2), 89–93.
OOJava – Subproject CTM_JOOP – "OOP with Java" within DAAD project "Software Engineering: Computer Science Education and Research Cooperation". `http://perun.pmf.uns.ac.rs/java/archive.html`
Pedroni, M., Meyer, B. (2010). Object-oriented modeling of object-oriented concepts a case study in structuring an educational domain. In: Hromkovič, J., Královič, R., Vahrenhold, J. (Eds.), *LNCS 5941*. Springer-Verlag, 155–169.
Swain, M., Anderson, J.A., Korrapati, R., Swain, N.K. (2002). Database programming using Java. In: *Proceedings of SoutheastCon Conference*, 220–225.
Thomas, K. (2003). Teaching databases at southampton university. In: *Proceedings of Teaching, Learning and Assessment in Databases*. Coventry, 123–126.
Thramboulidis, C. (2005). Teaching advanced computing concepts in Java: a constructivism-based approach. *Journal of Informatics Education and Research*, 7(3), 1–12.
Vesin, B., Ivanović, M., Klašnja-Milićević, A., Budimac, Z. (2013). Ontology-based architecture with recommendation strategy in Java tutoring system. *Computer Science and Information Systems Journal*, 10(1), 237–261.
WS (2012). *Workshops of DAAD project "Software Engineering: Computer Science Education and Research Cooperation"*, `http://www2.informatik.hu-berlin.de/swt/intkoop/daad/`
Yang, C.D. *(*2003). Teaching wireless networking and security with Java 2 micro edition (J2ME™). In: *Proceedings of 33rd ASEE/IEEE Frontiers in Education Conference*. Boulder, CO, T2C–7.

M. Ivanović holds the position of full professor at the Faculty of Sciences, University of Novi Sad, Serbia since 2002. She is the Head of Chair of Computer Science and a member of University Council for Informatics. She is author or co-author of 13 textbooks and of more than 240 research papers on multi-agent systems, e-learning and web-based learning, software engineering education, intelligent techniques (CBR, data and web mining), most of which are published in international journals and conferences. She is/was a member of Program Committees of more than 100 international conferences and is the Editor-in-Chief of Computer Science and Information Systems Journal. She has been principal investigator and participant of more than 20 international projects.

Z. Budimac holds the position of full professor at the Faculty of Sciences, University of Novi Sad, Serbia since 2004. Currently, he is the Head of the Computing Laboratory. His fields of research interests include educational technologies, agents and distributed systems, case-based reasoning, and programming languages. He was principal investigator of more than 20 projects. He is the author of 13 textbooks and more than 220 research papers, most of which are published in international journals and international conferences. He is/was a Program Committee member of more than 80 international conferences and is an Editorial Board Member of Computer Science and Information Systems Journal.

A. Mishev holds the position of assistant professor at the Faculty of Computer Science and Engineering, UKIM, Skopje, FYR Macedonia. In the focus of his research are infrastructures for collaborative computing and research, primarily Grid and High Performance Computing systems. He researched in the areas of computer architectures and networks, software engineering, Internet technologies and e-learning. He participated in the implementation of over 25 international projects targeting the development of IT infrastructure and IT education. He currently holds a position of Vice-dean for Educational Affairs at the FCSE, UKIM. He is author of over 35 scientific papers published in international journals and proceedings of conferences. He is a member of several associations in the field of information technology (IEEE, ICT-ACT).

K. Bothe holds the position of full professor of software engineering at the Institute of Informatics at Humboldt University Berlin. His research interests cover: compiler constructions, software engineering, software testing methodology and tools, programming languages, e-learning, and logic programming. Since 2000 he is the project leader of a DAAD project "Software Engineering: Education and Research Cooperation" with 15 universities of 8 countries as part of the special DAAD program "Academic reconstruction of South Eastern Europe". From 2004–2007 he was grantholder of an EU Tempus project "Joint M.Sc. Curriculum in Software Engineering".

I. Jurca is a professor (retired) of Software Engineering at the Department of Computers, Faculty of Automation and Computers, "Politehnica" University of Timişoara, Romania. His primary interests are in development of object-oriented distributed applications and software performance evaluation. He is the author of 8 books (in Romanian) and more than 40 papers published in journals and conference proceedings.

Bendro Java kalbos mokymo medžiagos banko naudojimas įvairiose mokymo programose, kursuose ir šalyse

Mirjana IVANOVIĆ, Zoran BUDIMAC, Anastas MISHEV,
Klaus BOTHE, Ioan JURCA

Remiant DAAD prieš keletą metų pradėtas projektas, skirtas ištirti Java programavimo kalbos mokymo aspektus. Pradinė projekto intencija – paskatinti projekto dalyvius paruošti mokomąją medžiagą Java programavimo kalbai mokyti. Per pastaruosius dvejus metus buvo pasirinkta keletas Java temų ir parengta atitinkama medžiaga. Visa medžiaga laisvai prieinama projekto veiklose dalyvaujančioms šalims. Straipsnyje supažindinama su sukurtos medžiagos naudojimo įvairiose šalyse ir universitetuose rezultatais bei įgyta patirtimi.