Course name	Semester	Hours	Final exam.	Credits
Object Oriented Programming	I/2	2Course-2Lab.	E	5

Course description	This course resumes the introduction to object-oriented programming with new object-oriented concepts: inheritance, class reusing, polymorphism, object-oriented containers, object factories, and exception handling. It also covers some specific charecteristics of the Java language (applets, graphics, event handling, threading), and applies object-orientation concepts to the design, coding, and testing of Java programs. Students in the course should expect to spend a fair amount of time on their own developing programs. Programming, like most other skills, is best learned by doing.
Prerequisite(s)& Corequisite(s)	Programming concepts and design recipes taught in " Programming languages " are prerequisites.
Textbook(s) and web materials	1. C. S. Horstmann and G. Cornell, *Core Java 2: Vol.1 - Fundamentals* , 7/e, Prentice Hall, 2005. 2. Deitel & Deitel, *Java: How to Program* , 4/e, Prentice Hall, 2003. 3. D. Geary, *Graphic Java 2, v.1 - Mastering the JFC,* 3/e, *v.2 - Swing,* Sun Microsystems Press, 1999-2000. 4. B. Eckel, *Thinking in Java* , 3/e, Prentice Hall. 5. www.java.sun.com In Romanian: 6. A. Athanasiu et al., *Limbajul Java. O perspectiva pragmatica* , Ed. Agora, 1996. 7. S.Tanasa, C.Olaru, S.Andrei, *Java de la 0 la expert* , Polirom, 2003.
Course objectives	The course objectives are: • To present advanced programming concepts in Java language . • To develop skills in the software design and programming using Java and its standard class libraries. • To develop understanding of problems and build skills in the use of abstraction in order to manage the problem complexity. A student completing this course should: 1. Be able to design and write Java programs to solve complex problems that meet requirements expressed in natural language. 2. Have a clear understanding of how a complex program is built in Java. 3. Have a deeperunderstanding of the semantics of object-oriented programs in terms of responsibility and collaboration. 4. Have a first approach to UML and various stages of the software development process.
Topics covered	**Review of some basic object-oriented concepts**. Concept instances vs. objects. Data and algorithm encapsulation. Instance and local variables. Instance methods. Parameter passing. Method overloading. *static* modifier. Basic input/output (2h) **Inheritance**. *extends* relationship. Method overriding. Static vs. dynamic binding. Polymorphism. Constructors revisited. Class hierarchies. The *protected* modifier. The *Object* class. Command line arguments. Abstract

	classes and methods. Wrappers. ArrayList, an extendible array. (4h) **Interfaces**. Interface definitions. Type conversions. Object Cloning. Inner Classes. (4h) **Applets**. Communication in web. HTTP protocol. HTML language. HTML tags and attributes. The applet context. Applet structure. Drawing on applets. Applet parameters. (3h) **Graphic programming**. The *java.awt* package. Object container concept. Graphic components. The *Graphics* class. Colors. Fonts. Dealing with images. Populating a container with components. Frames vs. applets. Container layout management. (4h) **Event handling**. Event-driven paradigm. The AWT event hierarchy. Event listeners and adapters. Event multicasting. The *Observer* pattern. (3h) **Exception handling**. Dealing with errors. Catching exceptions. Logging. Assertions. Debugging techniques. (2h) **Advanced input/output**. More about the *java.io package*. Dealing with web resources. (2h) **Collections framework.** *Collection* and *Map* hierarchies. Putting collections and maps to use. List and tree processing. Sorting and search algoritms. (4h)
Laboratory	This is a list of the main laboratory topics: 1. Review of the Java language. Class definition. I/O. Arrays. 2. Inheritance and polymorphism. 3. Inheritance. Abstract classes. 4. Interfaces. Inner classes. 5. Applets. 6. Programming graphical interfaces. 7. Event handling. 8. Observer. 9. Exception handling. 10. I/O with files. Introduction to collections. 11. Sorting and searching algorithms with Java collections. 12. Recapitulative exercises. Final exam simulation. 13-14. Laboratory and test redoing. There are 10 short assignments (exercises published in the course site in the weekend before the laboratory class) and few small projects. • The short assignments will cover all chapters of the course and put into practice concepts of object-oriented programming, and their relationship to the Java language. Assignments should be prepared for the next class period. Some may be collected for grading; others will be reviewed in class using Java 1.4 SDK that is available on the lab PCs. The student is strongly encouraged to install a copy on your own PC from the Sun web site. • The projects give the students an opportunity to integrate their knowledge from *all* the topics covered during laboratory classes and apply them to a problem in an area that interests them. Attendance at each laboratory is required. Only two laboratory sessions can be re-done!
Grading	**40%** Final exam (one or two programs to be developed) **60% Semester activity** - Labs and lectures attendance (10%) - 2 pre-announced tests (30%). Attendance at tests is required. Only one test can be re-done!

	- Lab activity/homework assignements (20 %). The points per assignment will vary depending importance and effort.
Professional significance	This course is about computer *programming* using advanced topics of the *object-oriented* paradigm. It emphasizes principles of sound design and good programming practice, aimed at developing programs of high quality and maintainability. Perseverence and discipline are mandatory attitudes in object-oriented software development. Programmers derive great satisfaction from seeing their object-oriented designs come to life, but they have to invest a lot of time and thought. Upon completion of the course, participants will have both the theoretical knowledge and practical experience to use Java to design quality programs of any complexity. This course actually belongs to the background of any engineering field.