Course name	Semester	Hours	Final exam.	Credits
Object Oriented Programming	**I/1**	**2Course-2Lab.**	**E**	**5**

Course description	This course provides an introduction to object-oriented programming with the Java language. It introduces the main object-oriented concepts: classes, objects, messages, encapsulation, static properties, arrays, I/O and covers the fundamentals of the Java language. 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)	None.
Textbook(s) and web materials	1. C. S. Horstmann and G. Cornell, *Core Java 2: Vol.1 - Fundamentals* , 6/e, Prentice Hall, 2002. 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 introduce the Java language as a true object-oriented 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. Have a clear understanding of the OO terminology generaly and that used to describe features of Java. 2. Be able to design and write Java programs to solve moderately complex problems that meet requirements expressed in natural language. 3. Have a clear understanding of what comprises a correct program in Java. 4. Be able to understand the Java API documentation. 5. Have an informal understanding of the semantics of object-oriented programs in terms of responsibility and collaboration.
Topics covered	**Introduction to programming languages.** Computer organization. Hardware vs. software. Computer programming. Programming language taxonomy. Syntax vs. semantics of a programming language. Syntactic component hierarchy. Compilers vs. interpreters. Developing computer programs. Run-time environment. Characteristics of the Java language. Writing a Java program (4h). **Fundamentals of Java programming.** 1. Data Side. Data types and their representation in memory. Variables. Variable declarations. Right- and left- value. Constants. Data encapsulation:

	data structures.(3h). 2. Algorithm Side. Operators and expressions. Operator precedence. Declarations vs. instructions. Assignment instruction. Control flow instructions. Compound statement. Algorithm encapsulation: methods. Method parameters. Parameter passing mechanisms. Method overloading. (4h). **Introduction to object-oriented programming**. Conceptual side: concepts and their relationships. Computer side: encapsulating data and algorithms. Concept instances vs. objects. Class concept. Class vs. object: *static* modifier. Class members. Class declaration. Global vs. local variables. Name visibility: *public* and *private*. (4h) **Using classes**. The *String* class. Dealing own classes. Static variables and methods. Object construction. Constructors. Class packaging. Java packages. (3h) **Arrays**. Array declarations. Dealing with arrays. Multidimensional Arrays. Passing arrays as method parameters. (3h) **Exception handling**. Dealing with errors. Catching and handling exceptions. (2h) **Streams and Files.** Data streams e their hierarchies. Putting streams to use. Reading with *BufferedReader*. Writing with *PrintWriter*. File Management. *StreamTokenizer* and *StringTokenizer*. (4h)
Laboratory	Here is a list of the main laboratory topics: 1. Program compilation, debugging and execution, basic elements in Java. 2. Variables, expressions, instructions. Writing static methods. 3. Strings. 4. Decisions, selections and loops. 5. Loops vs. recursion. Recursive algorithms. 6. Defining own classes. Dealing with instance members. 7. Constructors. Method overloading. 8. Arrays . 9. Dealing with more classes. Packages 10. Input/Output operations on files. 11. Input/Output operations on files. 12. Final exam simulation. There are 12 short assignments (exercises given in class), due one week after the students get them, and one or two 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. The projects should be delivered at last two weeks before the final examination. Students must pass the project portion of the class to pass the class. Attendance at each laboratory is required!
Grading	**40%** Final exam (individual quiz + one program to be developed) **60% Semester activity**

	- Labs and lectures attendance (10%) - 2 pre-announced tests (30%) - Lab activity/homework assignements (20 %). The points per assignment will vary depending importance and effort.
Professional significance	This course is about computer *programming* using the *object-oriented* paradigm. It emphasizes principles of sound design and good programming practice, aimed at developing programs of high quality and maintainability. Object-oriented programming is not so easy and involves creative design and a significant theoretical background. 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 small programs. Such a course actually belongs to the background of any engineering field.