Object-Oriented Programming

Prof. Daniele Toti

COURSE AIMS AND INTENDED LEARNING OUTCOMES

The course aims to provide students with an overview of object-oriented programming, with specific reference to the Java language.

In this context, the course covers, in depth, most aspects related to object-oriented programming of standalone or desktop applications using the Standard Edition (SE) of the Java language, with particular emphasis on the latest versions of the language that enjoy long-term support.

The expected learning outcome is the ability to develop medium-complexity applications with a graphical interface, such as simple video games or windowed applications. A student who successfully completes the course will also have the necessary tools to take certification exams related to Java SE and will have acquired the basic skills to work as a software developer in a professional setting.

COURSE CONTENT

* Introduction to object-oriented programming and the Java language
* Basic syntax, packages, imports, main method, console I/O
* Variables, assignments and primitive types
* Programming environment and IDE: Eclipse
* Fundamental mechanisms: comments, conditional statements, loops, operators
* Constructors, attributes, access modifiers, methods, casting
* Heap and stack
* Strings, arrays, wrapper classes
* Inheritance, polymorphism, abstract classes and interfaces
* Collections and generics
* Exceptions
* File and file system management: java.io, java.nio
* Using external libraries
* Dependency management with Maven
* Enums
* Nested classes
* Concurrency and multithreading
* Dates, Locale, ResourceBundle
* Elements of functional programming, Lambda expressions and Streams
* Graphical user interfaces with Java Swing: fundamental aspects, 2D graphics elements, threads and workers
* Event-driven programming
* Introduction to object-oriented design
* Design patterns: GRASP, GoF
* Architectural patterns: MVC
* Software version control
* Aspects related to the profession of software developer, with references to real projects and companies in the industry

READING LIST

Lecture slides, exercise notes, solutions to completed exercises and online resources communicated to the students. Recommended textbooks:

* *Head First Java, 3rd Edition*
* *Think Java, 2nd Edition*
* *Thinking in Java, 4th Edition*
* *Applying UML and Patterns, 3rd Edition*
* *Design Patterns*

Some of these textbooks will be made available to students on the Blackboard platform along with the rest of the materials. The course will not strictly follow any of these texts but will cover, integrate, and delve into various topics. Therefore, attendance is strongly recommended.

TEACHING METHOD

Theory lectures, exercises, individual and group assignments, development and discussion of sample video games such as Battleships and Match-3.

ASSESSMENT METHOD AND CRITERIA

The exam consists of an oral test that is made up of two parts:

1. Discussion of the project activity
2. Verification of knowledge of the topics covered during the course

The project activity involves the development of a medium-complexity Java application with a graphical interface, to be agreed upon with the instructor, typically a (more or less simple) video game. The application should be “mavenized”, meaning that dependency management is handled by Maven. The project can be carried out in a group (strongly recommended, maximum two people) or individually. The project activity also includes the drafting of a brief software documentation report, which, among other things:

* Describes the functioning of the application
* Presents the model of the graphical interface
* Lists the user input commands
* Describes the adopted software architecture, including the UML class diagram related to the main classes
* Mentions any used libraries
* Clearly describes the individual contributions of any group members.

The project material should be submitted at least one week before the date on which one of the group members intends to take the exam. The submission method will be agreed upon with the instructor and should include, at minimum:

* Source files of the program, included in the corresponding Maven-based Eclipse project
* Executable JAR file of the program generated with Maven
* Accompanying report in Word/PDF format

The project discussion during the exam will focus on the design choices and their implementation and will be conducted by both members of the working group simultaneously. Therefore, it is strongly recommended for group members to take the exam together in the same session.

Knowledge of the course topics will be assessed through theory questions, reasoning questions, and the potential proposal of code to write and/or analyze.

NOTES AND PREREQUISITES

This course is not suitable as a first programming course. Students should already possess basic programming knowledge. Specifically, it is recommended that students have completed the Laboratory of Computer Programming course and have familiarity with fundamental programming concepts and object-oriented programming topics covered in that course.

*Office hours and location*

Information regarding student office hours can be found on the instructor’s personal page.

Further information can be found on the lecturer's webpage at http://docenti.unicatt.it/web/searchByName.do?language=ENG or on the Faculty notice board.