COURSE OUTLINE

(1) GENERAL

SCHOOL	ENGINEERING		
ACADEMIC UNIT	ELECTRICAL AND COMPUTER ENGINEERING DEPT.		
LEVEL OF STUDIES	Undergraduate		
COURSE CODE	**ECE_K430**	**SEMESTER**	**2**
COURSE TITLE	Object-Oriented Design and Programming		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	**WEEKLY TEACHING HOURS**	**CREDITS**
Lectures	3	
Laboratory	1	
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (4).	4	5

COURSE TYPE *general background, special background, specialised, general knowledge, skills development*	Special Background.
PREREQUISITE COURSES:	No. Students are advised to have already attended the course: Procedural Programming
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	Greek
IS THE COURSE OFFERED TO ERASMUS STUDENTS	No
COURSE WEBSITE (URL)	https://www.ece.uop.gr/

(2) LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A

- *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area*
- *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning*

and Appendix B

- *Guidelines for writing Learning Outcomes*

The main aim of the course is to introduce the basic concepts of object oriented design and its techniques of object oriented programming (OOP) using the Java programming language as a case study.

Keywords: modeling, UML language, object-oriented programming, Java, class, graphical user interface, event-based programming, MVC

Learning Outcomes

Upon successful completion of the course, student will be able to:

At the Knowledge level:

- understand the concept of subtraction and how it is applied to the object-oriented modeling
- be familiar with the UML language diagrams
- know the basic principles of object-oriented programming and differences with the classic structured programming.
- describe the basic concepts of object-oriented programming: object, class, abstract class, subclass, superclass, class variable, object variable, constructor, method, inheritance, exception, encapsulation and polymorphism.
- know the syntactic properties of the Java language and organize the code in different files and packages.

At the Skill level:

- be able to design models of the developing system using the available UML language diagrams
- use efficiently popular integrated development environments (IDEs) such as NetBeans, for compiling Java classes and correcting of syntactic and logical errors.
- implement programmatic solutions that incorporate features of inheritance and polymorphism.
- design a class hierarchy and implement it in Java.
- design and develop a correct, modular, reusable and maintainable Java code.
- be able to know and to use Java key elements appropriately: data types, object classes and subclasses, constructors, methods, interfaces, packages, exceptions, threads and object collection libraries.
- design and implement simple graphical user interfaces and event management.
- handle files via Java program.
- develop multithreaded applications in Java.

At the level of Abilities:

- solve computational problems described in natural language, by using object-oriented design techniques.
- select and combine appropriate object-oriented design patterns for small and medium-sized software systems.
- select and combine the appropriate diagrams to model the software system under development.
- have the ability to design, implement, test, debug and document programs in Java language.
- evaluate the feasibility of expanding a program aimed at expanding the problems it can cover.
- understand the structure and function of Java code that he has not programmed and adapt it to his needs.
- select and combine the appropriate tools / libraries to develop a qualitative and efficient Object-Oriented Program from the description of the problem.

General Competences

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	*……*
Production of new research ideas	*Others…*
	…….

- Search for, analysis and synthesis of data and information, with the use of the necessary technology
- Working independently
- Team Work

- Working in an international environment
- Production of free, creative and inductive thinking
- Production of new research ideas

(3) SYLLABUS

The course is developed in the following 13 lectures:

i. Introduction to Object Oriented Design: Object (encapsulation, concealment of information, interface), class, class hierarchy, inheritance, specialization relationships, aggregation relationships, object communication.

ii. Introduction to UML. Structural diagrams. Behavioral diagrams.

iii. Class diagrams, case diagrams, sequence diagrams, collaboration diagrams, etc.

iv. Introduction to the Java programming language: History, features, advantages / disadvantages, types of programs, compilation / execution, NetBeans development environment.

v. Similarities / differences with language C: Data Types, Operators, Program Control / Repetition Loops, Tables, Strings.

vi. Introduction to Object-Oriented Programing & Basic Program Structure: class, object, variable, method, constructor, object communication, private / public data (public, protected, private), setters / getters methods, method overload, classes variables / constants , class methods.

vii. Inheritance: subclass, superclass, inheritance tree, multiple inheritance, interfaces, polymorphism, overriding of variables and methods, aggregation relationship, abstract classes.

viii. Introduction to awt-swing graphics: Graphic classes, components, events.

ix. GUI & Events management: components, containers, layout managers, event listeners. Internal classes & anonymous classes.

x. Exceptions: need to handle errors, advantages, runtime errors, try / catch / finally structure, capture / handle / throw exception.

xi. Concurrent Programming & Threads: concurrent block, Thread class, runnable interface, thread life cycle, thread priorities, synchronized statement.

xii. Data Input and Output: data streams (open, use, close). Files: text / binary, open, read / write, close.

xiii. Model-View-Controller pattern: Models, Views, Controller and examples of their use. Java coding style: good and widespread programming techniques in Java.

(4) TEACHING and LEARNING METHODS - EVALUATION

DELIVERY *Face-to-face, Distance learning, etc.*	Face to face in class and in the laboratory. Distance learning support via e-Class system.
USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY *Use of ICT in teaching, laboratory education, communication with students*	• Supporting the learning process through the e-Class platform (for notification of the course regulations, for distribution of slides, laboratory exercises, supplementary material, announcements, links, bibliography, etc.).

- During the lectures of the theoretical part, a projector and presentations in electronic form are used, which are also posted on the eclass from the beginning of the semester.
- During the lectures, a computer is used to write and execute code.
- Use of specialized software integrated development environment (such as Netbeans, BlueJ, etc.).
- The lectures of the course are available in eclass and as video lectures.

TEACHING METHODS

The manner and methods of teaching are described in detail.

Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.

The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS

Activity	***Semester workload***
Theory Lectures	39
Laboratory Exercises	13
Preparation of laboratory exercises	13
Project elaboration	25
Independent study of lectures and bibliography	35
Course Total	***125 hours (5 ECTS)***

STUDENT PERFORMANCE EVALUATION

Description of the evaluation procedure

Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other

Specifically-defined evaluation criteria are given, and if and where they are accessible to students.

A. Written final exam that includes:
- Solving exercises
- Multiple choice questions
- Short answer questions

B. Preparation of laboratory exercises and project.

Remarks:
- The final grade results from the weighting of the theory and work grades with coefficients determined at the beginning of the semester and announced to the students via e-class. Indicatively it will be about 70% - 30%
- Laboratory exercises and assignments are submitted electronically and students are asked to take an oral exam on them.
- The exam material and the evaluation process are communicated to the students in the lecture hall and in the e-class.

(5) ATTACHED BIBLIOGRAPHY

- *Suggested bibliography:*

- Farrell Joyce, "Java Programming, 9th edition, ISBN: 978-1337397070 (2019)
- Paul J. Deitel and Harvey M. Deitel, " Java How To Program, Late Objects (11th Edition) ", ISBN: 978-0134791401 (2017)
- Herbert Schildt, "Java: The Complete Reference, 11th Edition", ISBN: 978-1260440232 (2019)

- *Related academic journals:*

- ACM Transactions on Programming Languages and Systems
- ACM Transactions on Software Engineering and Methodology
- IEEE Transactions on Software Engineering