DEGREE CURRICULUM

OBJECT ORIENTED PROGRAMMING

Coordination: MARTÍNEZ RODRÍGUEZ, SANTIAGO

Academic year 2019-20

2019-20

Subject's general information

<table>
<tr><td>Subject name</td><td>OBJECT ORIENTED PROGRAMMING</td></tr>
<tr><td>Code</td><td>102368</td></tr>
<tr><td>Semester</td><td>2nd Q(SEMESTER) CONTINUED EVALUATION</td></tr>
<tr><td>Typology</td><td>
<table>
<tr><th>Degree</th><th>Course</th><th>Character</th><th>Modality</th></tr>
<tr><td>Bachelor's degree in Digital Interaction and Computing Techniques</td><td>1</td><td>COMMON</td><td>Attendance-based</td></tr>
</table>
</td></tr>
<tr><td>Course number of credits (ECTS)</td><td>6</td></tr>
<tr><td>Type of activity, credits, and groups</td><td>
<table>
<tr><th>Activity type</th><th colspan="2">PRALAB</th><th>TEORIA</th></tr>
<tr><td>Number of credits</td><td>3</td><td>3</td><td>3</td></tr>
<tr><td>Number of groups</td><td>1</td><td>2</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Coordination</td><td>MARTÍNEZ RODRÍGUEZ, SANTIAGO</td></tr>
<tr><td>Department</td><td>COMPUTER SCIENCE AND INDUSTRIAL ENGINEERING</td></tr>
<tr><td>Teaching load distribution between lectures and independent student work</td><td>6 ECTS = 25x6 = 150 working hours:
40% -> 60 in-class hours,
60% -> 90 autonomous work hours.</td></tr>
<tr><td>Important information on data processing</td><td>Consult this link for more information.</td></tr>
<tr><td>Language</td><td>Catalan.</td></tr>
<tr><td>Distribution of credits</td><td>Santi Martínez: 3
Marc Tomàs: 3</td></tr>
</table>

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
MARTÍNEZ RODRÍGUEZ, SANTIAGO	santi.martinez@udl.cat	3	
TOMÀS GLEYAL, MARC	marc.tomas@udl.cat	3	

Subject's extra information

We assume the students have all the concepts of Algorithms and Programming as we build upon them into two directions: object-oriented programming and recursive design.

Learning objectives

The main learning objectives are:

- To apply the Object Oriented Programming paradigmn to simple problems.
- To use the basic Java file types.
- To design simple recursive algorithms.
- To use the Java standard documentation.
- To use an Intengrated Development Environment.

Competences

Basic Competences

- **B01**. That students have demonstrated to possess and understand knowledge in an area of study that starts from the base of general secondary education, and is usually found at a level that, although supported by advanced textbooks, also includes some aspects that imply knowledge coming from the vanguard of his/her field of study.

Transversal Competences

- **CT3**. Acquire training in the use of new technologies and information and communication technologies.
- **CT5**. Acquire essential notions of scientific thought.

General Competences

- **CG2**. Design, develop, evaluate and guarantee the accessibility, ergonomics, usability and security of computer systems.
- **CG3**. Use adequate hardware and software platforms to develop and execute interactive digital applications.
- **CG5**. Know the basic subject areas and technologies needed to learn and develop new methods and technologies, and those that help to adapt to new situations.
- **CG7**. Solve problems through initiative, determination, independence and creativity.
- **CG8**. Capacity for abstraction and critical, logical and mathematical reasoning.

Specific Competences

- **CE2**. Capacity to understand and master the basic concepts of discrete mathematics, logics, algorithmic and computational complexity, and its application to solve computational problems.
- **CE3**. Basic knowledge of the use and programming of computers, operating systems and databases, and their use in the development of interactive applications.
- **CE4**. Capacity to know, understand and evaluate the structure and architecture of computers, as well as the basic components that conform them.
- **CE16**. Capacity to design and evaluate person-computer interfaces that guarantee the usability of systems, services and computer applications.
- **CE17**. Capacity to apply knowledge on design to propose and defend a design concept for an interactive system and use proper creative technologies to develop each project.
- **CE24**. Capacity to understand the human factors involved in any interactive process between humans and technology, as well as

being able to adequately apply them in the design of interactive products and services, and their interfaces.

Subject contents

Unit 1. Introduction to Java

1.1 From C to Java
1.2 The ACM Task Force Library
1.3 The main program
1.4 Using auxiliary functions
1.5 Arrays in Java
1.6 Strings in Java

Unit 2. Object Oriented Programming

2.1 Objects and references
2.2 Graphic classes in the ACM library
2.3 The String class
2.4 Class definition in Java

Unit 3. File processing

3.1 Types of files
3.2 Sequential text files
3.3 Random access binary files
3.4 MergeSort

Unit 4. Recursive design

4.1 Function calls
4.2 Thinking recursively
4.3 Recursivity using cursors
4.4 Binary search
4.5 Multiple recursion

Methodology

Each week students attend 2 hours with a Large group and 2 hours with a Medium Group. Medium Group sessions are practices.

Large Group: Theory Classes (3 ECTS)

- Theory: Classes supported by class notes.
- Practical application: always working on concrete examples.

Medium Group: Practical Classes (3 ECTS)

- Aimed to the resolution of practical cases by the students (there is a problems collection which includes exams from previous years)
- Personal tutoring of projects and difficulties.
- Use of an Integrated Development Environment.

Autonomous work:

- Software projects are done non-presentially.
- We recommend students to solve the problems in the collection to practice and get feedback from the teaching staff.

Development plan

Week	Large Group Activity	Medium Group Activity	Autonomous Work
1	Presentation + From C to Java (1, 2, 3)	javac and java. Problem 1	Study and problem solving
2	From C to Java (rest)	Problems 2, 3, 4	Study and problem solving

Week	Large Group Activity	Medium Group Activity	Autonomous Work
3	Introduction to OOP (1, 2)	Problems 5, 6, 7	Study and problem solving Project 1
4	Introduction to OOP (3, 4)	Problems 1, 2	Study and problem solving Project 1
5	Introduction to OOP (5, 6, 7)	Problems 3, 4, 5	Study and problem solving Project 1
6	Further OOP (8, 9)	Problems 6, 7, 8	Study and problem solving Project 2
7	Further OOP (10, 11)	Problems 9, 10, 11	Study and problem solving Project 2
8	Further OOP (12, 13, 14)	Previous exams	Study and problem solving
9	Evaluation		
10	File management (1, 2, 3)	Sol. exam. Problem 1	Project 2
11	File management (4, 5, 6)	Problems 2, 3	Study and problem solving Project 2
12	File management (7, 8)	Problems 4, 5	Study and problem solving Project 3
13	Recursive design (1, 2, 3)	Problems 9, 10	Study and problem solving Project 3
14	Recursive design (4, 5, 6)	Problems 1, 2	Study and problem solving Project 3
15	Recursive design (9, 10)	Previous exams	Study and problem solving
16	Evaluation		
17	Evaluation		
18			Study and problem solving Project 3
19	Recovery		

- Numbers in the second column correspond to the section in the class notes of the subject.
- Those in the third column correspond to the numbers in the problems collection.

Evaluation

Evaluation activities

Acronym	Evaluation Activity	Weight	Minimum Score	Group	Compulsory	Recoverable
EP1	1st Midterm Exam	25%	4	No	Yes	Yes (with 2nd midterm exam)
EP2	2nd Midterm Exam	25%	4	No	Yes	Yes (recovery exam, with a max. score of 8)
PR1	Project 1	15%	No	Yes (≤ 2)	No	Yes (at the end of the course, with a max. score of 8)
PR2	Project 2	20%	No	Yes (≤ 2)	No	Yes (at the end of the course, with a max. score of 8)
PR3	Project 3	15%	No	Yes (≤ 2)	No	No
Final Score = 0.25 · EP1 + 0.25 · EP2 + 0.15 · PR1 + 0.2 · PR2 + 0.15 · PR3						

Remarks:

- Subject is passed if Final Score is greater or equal to 5.

- A passed 2nd midterm exam recovers a failed 1st midterm exam.
- 1st midterm exam score is only taken into account if its greater than 2nd midterm exam (if not, the 2nd midterm exam score is used).
- If the student has to take the recovery exam, the score of the first midterm exam won't be taken into account and the maximum score of the theory part will be 8.
- If the student has to re-send the first or second project at the end of the course, the project maximum score will be 8.
- A project detected as a copy (or a non-original work) will be qualified with 0 and it won't be recoverable.

Bibliography

Basic

- Class notes (in spanish).
- Eric S. Roberts, The Art & Science of Java: An Introduction to Computer Science, Pearson Education, 2008 (hi ha una versió preliminar disponible en pdf).
- Eric S. Roberts, Thinking Recuersively with Java, John Wiley & Sons, 2006.

Complementary

- Documentació de la biblioteca ACM Java Task Force http://jtf.acm.org/
- Kathy Sierra y Bert Bates, Head First Java,O'Reilly, 2003.
- Jorge A. Villalobos y Rubby Casallas, Fundamentos de Programación. Aprendizaje Activo Basado en Casos. Pearson Pentice-Hall, 2006.

Adaptations to the methodology due to COVID-19

The course notes documents will be provided together with explanatory videos of the theory part.
Video conferences will be held to help with the resolution of problems.
Besides, the students can ask to be attended by videoconference.

Adaptations to the evaluation due to COVID-19

The two midterm exams will be substituted for four additional programming exercises that, summed, will correspond to the 50% of the grade of the subject (12.5% each exercise).
Any failed exercise can be recovered returning it before the end of the course with a maximum score of 8.