Universitat Autònoma de Barcelona

2019/2020

Object-Oriented Programming

Code: 104389
ECTS Credits: 6

Degree	Type	Year	Semester
2503740 Computational Mathematics and Data Analytics	FB	1	2

Contact

Name: Joan Serrat Gual
Email: Joan.Serrat@uab.cat

Use of Languages

Principal working language: catalan (cat)
Some groups entirely in English: No
Some groups entirely in Catalan: Yes
Some groups entirely in Spanish: No

Teachers

Lluis Gomez Bigorda

Prerequisites

There are no official prerequisites, but you need to know Python.

Objectives and Contextualisation

One of the most employed methodologies of design and programming is object-orientation. Software is organized in classes that contain methods (procedures) and attributes (data). The instances or objects of these classes send messages to each other (they make "calls" to methods of other objects) in order to achieve the desired functionality. Apart from the concept of class, three other fundamental elements are inheritance, composition and separation of public and private parts of classes. These elements are therefore necessary for OO programming, but not sufficient: you need to know how to use them to reach solutions that are easily extendable in the future. There are a number of design principles or heuristics that tell us how to use these elements for this goal. And over them, design patterns are being built, which are solutions at the OO design level to recurring problems, which often appear when solving seemingly independent problems. In this subject, the OO concepts are introduced and practiced as well as design principles and patterns.

Learning revolves around a project --- the practicum --- in which it is necessary to apply all the topics explained in order to design and implement a software application based on a statement that plays the role of a document of requirements. The programming languages used in the course (and the practicum) will be Python, but we will use also Java for some exercises because it better supports the concepts of object orientation.

Competences

- Apply basic knowledge on the structure, use and programming of computers, operating systems and computer programs to solve problems in different areas.
- Design, develop and evaluate efficient algorithmic solutions to computational problems in accordance with the established requirements.
- Make effective use of bibliographical resources and electronic resources to obtain information.

- Students must be capable of applying their knowledge to their work or vocation in a professional way and they should have building arguments and problem resolution skills within their area of study.
- Students must be capable of communicating information, ideas, problems and solutions to both specialised and non-specialised audiences.
- Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.
- Using criteria of quality, critically evaluate the work carried out.
- Work cooperatively in a multidisciplinary context assuming and respecting the role of the different members of the team.

Learning Outcomes

1. Describe the basic operation of computation systems.
2. Evaluate and analyse the complexity of computational algorithmic solutions in order to develop and implement that which guarantees best performance.
3. Make effective use of bibliographical resources and electronic resources to obtain information.
4. Recognise and identify the methods, systems and technologies pertaining to computing.
5. Select and use algorithmic structures and the representation of appropriate data to solve a problem.
6. Students must be capable of applying their knowledge to their work or vocation in a professional way and they should have building arguments and problem resolution skills within their area of study.
7. Students must be capable of communicating information, ideas, problems and solutions to both specialised and non-specialised audiences.
8. Students must have and understand knowledge of an area of study built on the basis of general secondary education, and while it relies on some advanced textbooks it also includes some aspects coming from the forefront of its field of study.
9. Understand the basic concepts in the structure and programming of computers.
10. Understand the basic principles of computer logic.
11. Use operating systems and programmes commonly used in various fields.
12. Using criteria of quality, critically evaluate the work carried out.
13. Verify and ensure the correct operation of an algorithmic solution in accordance with the requirements of the problem to be resolved.
14. Work cooperatively in a multidisciplinary context, taking on and respecting the role of the distinct members in the team.

Content

1. Object orientation concepts: encapsulation, inheritance, composition, interface. UML class diagrams
2. Object-oriented programming in Python3 and Java
3. Assignment of responsibilities: GRASP patterns
4. General principles of object-oriented design
5. Design patterns: creational, structural, behavioral
6. Coding style, assertions and logging messages

Methodology

As stated in the objectives, the subject follows a learning methodology based on the development of a project. Its concrete implementation with regard to the use of class hours is as follows:

- Theory classes. In them the teacher introduces the theoretical contents and gives useful references (books, articles, web pages) so that the student can then continue their learning, as well as develop the proposed project. The transparencies used in these classes should not be taken as the sole source of study. It is necessary to study the recommended books, and the articles, chapters of open access books and others, available on the web page of the subject. In these classes we will also solve small problems, design and / or OO programming in Java and Python.

- Practical sessions. The students will have formed at the beginning of the course groups of 2-3 students. In these sessions students come to 1) 'pass on accounts' with the teacher of the work done each week of the project and resolve doubts, and 2) to be evaluated continuously, in accordance with the scale in the statement of practice Therefore, the practice * no * is done only during these sessions but requires a previous job.

All the materials of the subject along with their detailed planning will be found on the virtual campus or be on the web page that will be announced. Most of these materials are in English.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Practicum sessions	75	3	7, 14, 11
Theory lectures	75	3	2, 9, 1, 4, 5

Assessment

There is a group assessment part and one individual assessment. The note of the first (PR) corresponds to the practicum. The second (EX) of the average of two exams during the term (EX1, EX2), or a retake exam (EXR1, EXR2).

The algorithm for calculating the final mark (FM) is the following (all marks over 10):

EX = 0.5 * (max (EX1 + EXR1), max (EX2, EXR2))
if (EX> = 4.0) and (PR> = 4.0):
FM = (EX + PR) /2.0
else:
FM = min (EX, PR)

Optionally, we will propose some exercises and small problems that will add points to EX1, EX2 and the corresponding retake exams.

If you do not deliver an exam or practice it, the mark in this part is zero. There is no retake for the practicum. The final grade will be "non-evaluable" if no exam or practicum has been delivered.

One thing that we want to ensure through the different evaluations is that each member of the group has really contributed to the practice in the same way as the others. Therefore, if, as a result of the questions the teacher makes during the project sessions or written test, we have evidences/hints that this does not happen, then the practicum mark will be individual: we will evaluate separately the part of the project that each member has done.

Notwithstanding other disciplinary measures that are deemed appropriate, and in accordance with the current academic regulations, the irregularities committed by the student who can lead to a variation of the qualification of an act of self- evaluation. Therefore, plagiarizing, copying or letting copying the practice or any other evaluation activity will imply fail with a zero and can not be recovered in the same academic year. If this activity has a minimum associated mark, then the student won't pass the course.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes

Exams	50%	0	0	2, 9, 1, 8, 7, 6, 4, 5
Practicum	50%	0	0	12, 10, 9, 1, 7, 4, 14, 3, 11, 13

Bibliography

Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.

Mastering object-oriented Python. Steven F. Lott. Packt publishing, 2014.

Head first object oriented analysis and design. B.D. McLaughlin, G. Pollice, D. West. O'Reilly, 2007.

Design patterns explained simply. Alexander Shvets. https://sourcemaking.com/design-patterns-ebook.

Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.

Clean code: a handbook of agile software craftmanship. R.C. Martin. Prentice Hall, 2008.