Universitat Autònoma de Barcelona

2021/2022

Object-Oriented Programming

Code: 104389
ECTS Credits: 6

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

The proposed teaching and assessment methodology that appear in the guide may be subject to changes as a result of the restrictions to face-to-face class attendance imposed by the health authorities.

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

Other comments on languages

Study materials (slides, handouts etc) are in English.

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 3 students. In these sessions students come to 1) review with the teacher 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. All these materials are in English.

Annotation: Within the schedule set by the centre or degree programme, 15 minutes of one class will be reserved for students to evaluate their lecturers and their courses or modules through questionnaires.

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 (EXR).

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

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

The practicum

The realization of the practicum is essential to assimilate the concepts of theory. The first thing we want to make sure is that the practicum has really been done by the group that delivers it, and that each member of the group has contributed to the same extent as the others. The second is that the practicum is done progressively during the course, in parallel with the theory. That's why we will do the following:

- The mark for the practice is obtained by combining the grade of the parts into which it is divided. Each part will be evaluated in situ in one of the practical sessions that we will announce at the beginning of the course (the milestones). The points that can be obtained from each part will be specified in the handout. These assessments have no retake, afterwards.

- On the evaluation days of the practicum (milestones) we will do a quick written validation test. If, as a result of the answers to this test or to the questions asked by the teacher during the assessment, we have doubts about the authorship or contribution of the group members, then the mark will change from group to individual and depending of how much and what part of the practice each member has done.

The exercices

We will propose some exercises and small problems that are optional. These exercises can be done individually or in small groups, the same or different than the practice. The exercises may add points to the theory note EX, with a top of 10. We say 'may' because obtaining these extra points is conditioned to a validation test in order to verify that the student has effectively completed the exercise he has delivered. We will do these validation tests only once, in the EX1 and EX2 exams, and they do not have retake afterwards.

Other

We do not preserve the mark of the practicum, exercises or exams from previous courses. Students taking again the course do not receive any differentiated treatment.

If an exam or part of the practicum is not delivered, the mark for that part is zero. The final grade will be 'no-avaluable' if nothing to be evaluated has been delivered, either exam or practicum.

Milestones are assessments equivalent to written exams. If a student does not show up, the mark is zero.

MH marks will be awarded at the discretion of students with a final final grade of 9.0 and taking into account all kinds of work done (exams, practicum and exercises).

The dates of continuous evaluation and delivery of works will be published on the virtual campus of the subject and may be subject to change in response to possible incidents. It will always be reported through the virtual campus, which will be the communication mechanism between teachers and students.

For written exam-type assessment activities, a place, date and time of revision will be indicated in which the student will be able to review the activity with the teacher. If the student does not show up for this review, this activity will not be reviewed later.

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 orletting 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

Basic

Theory classes are based on a set of slides, which are a summary, a synthesis of concepts that are best explained in these books, which are well worth reading or consulting to better understand them and to expand knowledge.

Design patterns. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Pearson Education 2003.
It is a classic of object-oriented design patterns, by the authors who proposed this concept. The original English version is from 1995 and so its examples and diagrams are old. But it explains the concepts very well.

Design patterns explained simply. Alexander Shvets. Book online,
https://sourcemaking.com/design-patterns-ebook.
Well explained and illustrated with examples, which are also programmed in Java, Python, C ++ and more.

Head first design patterns. E. Freeman, E. Freeman, K. Sierra, B. Bates. O'Reilly, 2004.
A very visual book on object-oriented design patterns. Very didactic, he explains them step by step and with examples in Java. A second edition (2020) has recently appeared.

Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems. Aurélien Géron. O'Reilly, 2019.
The practicum consists of the object-oriented design and programming of a classification and regression method known as random forest. Three chapters in this book explain very well the concepts around this method and the very algorithm that needs to be implemented.

Complementary

Introduction to Programming Using Java, David J. Eck. 2020.
Book online, http://math.hws.edu/javanotes/. Some exercises are in Java language and this is an online and updated book that explains the language from scratch. It does not go into the most advanced details butit is more than enough for what we need.

Python: master the art of design patterns. D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
A very extensive book (700pp) on design patterns in Python. A very positive aspect is that it shows how to program them in Python in detail, one by one.

Object-oriented mastering Python. Steven F.Lott. Packt publishing, 2014.
Advanced Python, or rather, going into the details of the language. "Grasp the intricacies of object-oriented programming in Python in order to efficiently build powerful real-world applications." Of interest is the chapter on logging and some others.

Software

- Python Anaconda, with libraries numpy, matplotlib.
- Pycharm community edition, plus plugin PlantUML.
- IntelliJ IDEA community edition