UAB
Universitat Autònoma de Barcelona

Object-oriented Programming

Code: 104389
ECTS Credits: 6

2025/2026

Degree	Type	Year
Computational Mathematics and Data Analytics	FB	1

Contact

Name: Guillermo Eduardo Torres

Email: guillermo.torres@uab.cat

Teachers

Enric Sala Esteva

Teaching groups languages

You can view this information at the end of this document.

Prerequisites

There are no official prerequisites, but it is necessary to have programming skills in Python.

Objectives and Contextualisation

The main objective is for students to acquire knowledge of Object-Oriented Design principles and Design Patterns in order to create software designs represented using UML diagrams, which will then be implemented in an object-oriented programming language.

This includes fundamental concepts such as classes, objects, methods, attributes, inheritance, composition, and encapsulation. Based on these principles, students will be introduced to Software Design Patterns, which provide reusable solutions to common problems in software development.

UML diagrams allow for a clear representation of the software's structures and relationships, facilitating the visualisation and comparison of different designs, as well as the planning of their implementation.

Learning Outcomes

1. CM09 (Competence) Develop effective algorithmic solutions to computational problems in accordance with the established requirements for object-oriented programming.

2. CM10 (Competence) Analyse the computational complexity of the algorithmic solutions to develop and implement the one that guarantees the best performance for object-oriented programming.
3. CM11 (Competence) Ensure the correct functioning of an algorithmic solution in accordance with the requirements of the problem to be solved for object-oriented programming.
4. KM09 (Knowledge) Recognise the methods, systems and technologies specific to object-oriented programming.
5. SM09 (Skill) Use operating systems and software commonly used in object-oriented programming.
6. SM10 (Skill) Use algorithmic and data representation structures suitable for problem-solving in object-oriented programming.

Content

1. Object-Oriented Concepts.
   Introduction to the fundamental principles of object orientation, such as encapsulation, inheritance, composition, and the use of interfaces, as well as graphical representation through UML class diagrams.

2. Object-Oriented Programming in Python.
   Use of a programming language to apply object-oriented concepts, including advanced language features.

3. Object-Oriented Design Principles.
   Analysis of principles for structuring maintainable and scalable software, and for assigning responsibilities consistently across components.

4. Design Patterns.
   Study of reusable solutions to common software design problems.

5. Coding Style and Best Practices.
   Application of style conventions, code documentation, use of comments, and techniques to improve software traceability and comprehension.

6. Incremental Practical Development.
   Progressive application of theoretical and practical content through exercises and a main project developed in phases.

Activities and Methodology

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Practicum sessions	13	0.52	CM09, CM10, SM09, CM09
Programming	50.5	2.02	
Study	50.5	2.02	
Theory lectures	36	1.44	CM11, KM09, SM10, CM11

The course follows a project-based learning methodology (PBL), in which students progressively develop a programming project applying the principles of Object-Oriented Programming (OOP). The project is structured

in milestones, which allow consolidating the content covered throughout the course, promoting meaningful, active, and practice-oriented learning.

Lecturer classes. Key concepts of the object-oriented paradigm are introduced, providing the necessary foundations to tackle the project development. These classes also include solving small design and object-oriented programming problems. Additionally, bibliographic references (books, articles, online resources) are provided to complement the face-to-face sessions. The lecture slides should not be considered the sole study resource; students are expected to actively consult the recommended materials available on the course's virtual campus.

Practical classes. At the beginning of the course, students are organized into small working groups. During these sessions, 1) the weekly progress on the project is reviewed and technical and/or conceptual doubts are addressed, and 2) continuous evaluation, both group and individual, is carried out according to the assessment criteria defined in the practice guidelines. These sessions are not designed to develop the entire project from scratch but to monitor and evaluate the work the groups have previously advanced autonomously.

Autonomous work. Learning programming requires dedication and continuous practice. Therefore, students are expected to work independently outside the classroom by studying theoretical content, solving exercises, reviewing examples, and steadily advancing in the project development.

The virtual campus contains all the course materials as well as the detailed course planning. All written content is provided 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.

Assessment

Continous Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Exams	50%	0	0	CM11, KM09, SM10
Practicum	50%	0	0	CM09, CM10, SM09

The assessment of the course is based on a continuous evaluation system that combines individual and group activities, with the aim of assessing both conceptual understanding and the practical application of theoretical-practical content.

Final Mark Calculation (FM)

All grades are out of 10 points. The final mark is calculated using the following algorithm:

$EX = \max(0.5 * (EX1 + EX2), EXR) + EX_{extra}$

$PR = 0.5 * (M1 + M2)$
if $(EX >= 4.0)$and $(PR >= 4.0)$:
$FM = 0,5 * (EX + PR)$
else:
$FM = \min(EX, PR)$

Donde:

- EX1 and EX2 are the first and second exams, which are individual written tests taken during the course.
- EXR is the resit exam, which may replace the average of EX1 and EX2 if it improves the grade.
- M1 and M2 are the evaluation milestones of the group and individual practical work carried out throughout the course.
- EX_{extra} refers to additional points earned through optional exercises.

Single Assessment: This course does not offer the single assessment system.

Practical Assessment

- The practical component is divided into several parts or *milestones*, assessed in situ during practical sessions.
- Each milestone has a specific score, as indicated in its corresponding assignment description.
- These evaluations cannot be retaken.
- Submissions must be made through the virtual campus. Late submissions can receive a maximum score of 5 points.
- In case of doubt about authorship or individual contribution within a group, the evaluation will become individual, with grades assigned according to each member's contribution.
- Attending and completing the practical sessions progressively is essential for assimilating the theoretical-practical concepts.

Exam Assessment

- Two written exams (EX1 and EX2) will be held during the term.
- A resit exam (EXR) is available and can improve the exam grade if its result is higher than the average of EX1 and EX2.
- Optional exercises (EX_{extra}) may add extra points to the theory grade (EX) and cannot be retaken.

Other Relevant Information

- Practicalwork, exercises, or exams from previous academic years will not be validated.
- Failure to submit an exam or part of the practical work will result in a grade of zero for that component.
- The final grade will be marked as "not evaluable" if thestudent does not submit any assessable activity.
- The Honours Distinction (MH) may be awarded at the instructor's discretion to students with a final grade above 9.0, taking all activities into account.
- Submission deadlines and evaluation dates will be published on the virtual campus and may be updated in case of unforeseen events; any changes will be communicated via that platform.
- A date and location will be provided for exam reviews; no reviews will be held afterwards if the student does not attend.
- This course/module does not include a single final assessment option.
- Plagiarism, copying, or any irregularity in assessable activities will result in a grade of zero for that activity, with no resit option in the same academic year and possible course failure.

Use of AI

- AI tools may be used as support tools for learning (for example, to improve writing, style, clarity of exposition, language accuracy, or to obtain assistance with technical aspects). Under no circumstances may they replace and/or supplant the student's learning activity or their acquisition of the specific knowledge of the course.
- It is not acceptable to use artificial intelligence tools to generate content for assignments that are subject to assessment. Any assessed tasks/activities suspected of having been generated by AI instead of the student will be considered as plagiarism and will be graded with a 0.

Bibliography

Basic Reading

Theoretical classes are based on a set of slides that summarise the key concepts of the course. These materials serve as a guide, and it is recommended to consult the following works for a deeper understanding and to broaden your knowledge:

Design Patterns, E. Gamma, R. Helm, R. Johnson, J. Vlissides. Pearson Education, 2003.
A classic book on object-oriented design patterns, written by the authors who introduced the concept. The original English version was published in 1995, so some examples and diagrams may feel outdated, but the fundamentals are explained very clearly.

Design Patterns Explained Simply, Alexander Shvets.
Online book: sourcemaking.com. Another related work by the same author is available at refactoring.guru. Provides clear explanations illustrated with examples in multiple languages, including Python, C++, and Java. Ideal as a practical introduction.

Recommended Reading

Head First Design Patterns, E. Freeman & E. Robson, 2021.
A highly didactic book on design patterns that uses simple, humorous examples to explain concepts.

Python: Master the Art of Design Patterns, D. Phillips, C. Giridhar, S. Kasampalis. Packt Publishing, 2016.
A comprehensive book (over 700 pages) focused on the detailed implementation of design patterns in Python. Very useful for understanding how to apply them effectively in this language.

Mastering Object-Oriented Python, Steven F. Lott. Packt Publishing, 2014.
Offers an advanced view of the Python language, delving into key aspects of object-oriented programming. Chapters such as the one on the logging system and other language-specific topics are particularly noteworthy.

Software

- Python using the package manager Pip or Anaconda to install libraries such as NumPy and Matplotlib.
- Spyder IDE for writing and debugging Python code.

Groups and Languages

Please note that this information is provisional until 30 November 2025. You can check it through this link. To consult the language you will need to enter the CODE of the subject.

Name	Group	Language	Semester	Turn
(PLAB) Practical laboratories	1	Catalan/Spanish	second semester	morning-mixed
(PLAB) Practical laboratories	2	Catalan/Spanish	second semester	afternoon
(TE) Theory	1	Spanish	second semester	morning-mixed