Teaching guide

IDENTIFICATION DETAILS

Degree:	Computer Engineering
Scope	Computer and Systems Engineering
Faculty/School:	Higher Polytechnic School
Course:	OBJECT-ORIENTED PROGRAMMING
Type:	Basic Training
ECTS credits:	6
Year:	1
Code:	5612
Teaching period:	Second semester
Subject:	Computing
Module:	Basic Training
Teaching type:	Classroom-based
Language:	Inglés
Total number of student study hours:	150

Teaching staff	E-mail
Moisés Martínez Muñoz	moises.martinez@ufv.es
Iván Barcia Santos	i.barcia@ufv.es

SUBJECT DESCRIPTION

The course 'Object-Oriented Programming' presents the object-oriented computational paradigm and its contribution to improve the quality of the software, in particular to favor the reusability, extensibility and reliability of the programs in a language-independent way.

This course corresponds to the Basic Training module and, within it, to the subject Computer Science. It is taught in the second semester of the first year of the Degree in Computer Engineering, and it requires a dedication of 150 hours on the part of the student.

It is built upon two foundational conceptual blocks:

Theoretical block: This component aims to introduce students to the fundamentals of the object-oriented programming paradigm, its underlying principles, and motivations. Key concepts such as classes and objects, abstraction, encapsulation, inheritance, and polymorphism will be covered, providing a solid foundation for understanding how this approach contributes to the development of more modular, maintainable, and scalable software.
Practical block: This component focused on the progressive development of technical skills for applying these concepts in real-world settings. Students will work on the design, implementation, and debugging of object-oriented applications, using a modern programming language and an Integrated Development Environment (IDE). They will learn to structure their programs coherently, to create, organize, and relate classes, to improve and refactor code, as well as to implement input/output systems and develop simple user interfaces that allow users to interact with the system.

GOAL

The main objective of the course is to train students to develop software applying the object-oriented programming paradigm. The aim is for the student to apply the key concepts of this paradigm to build modular, reusable and maintainable software systems.

The specific aims of the subject are:

Understand the object-oriented paradigm and its usefulness compared to other paradigms.

Design software solutions by identifying classes, attributes, methods, and relationships between objects.

Implement programs using the principles of encapsulation, abstraction, inheritance, and polymorphism in an object-oriented programming language.

Use an integrated development environment (IDE) to write, debug, and execute code.

Verify the correct functioning of software through functional testing and manual debugging.

PRIOR KNOWLEDGE

For access to the degree program, it is highly recommended to have successfully completed the Introduction to Programming course and to be familiar with the following concepts:

Structure of a program.
Expressions and Operators.
Identifiers and variables.

Control statements: selection and iteration structures.
Functions.
Arrays.
Basic file operations.
Pointers.

COURSE SYLLABUS

Unit 1. The Object-Oriented Programming Paradigm

Background and historical perspective.
What is OOP?
The principle of abstraction. Basic concepts: classes, attributes and methods.
Principles of: abstraction, encapsulation, inheritance and polymorphism.
Object-oriented languages.

Unit 2. Foundations of the Object-Oriented Programming

The .NET Core framework.
The C# programming language.
C# basics: Conditions, and selection instructions.
C# basics: Loop instructions and flow control mechanisms.
C# basics: Classes and objects.
C# basics: Methods.
C# basics: Encapsulation.
C# Basics: Arrays, Collections, and Dictionaries
UML basics.

Unit 3. Relationships in Object-Oriented Programming

Inheritance: constructors, accessibility modifiers, and method overwriting.
Association: aggregation and composition.
Abstraction: abstract classes, abstract methods, and interfaces.
UML relationships.

Unit 4. Extended Object-Oriented Programming

Polymorphism: method overloading, virtual members.
Files (text, csv, json).
Error handling: exceptions.
Helpers: Strings, random numbers, command line arguments.

Unit 5. Graphical User Interfaces (GUIs)

Graphical User Interfaces.
The .NET MAUI framework: Xamarin.Forms, and architectural patterns.
Basic components: pages, layouts, controls, and resources.

Navigation.

EDUCATION ACTIVITIES

For the development of the subject, activities are combined to support both the acquisition of theoretical knowledge and its practical application. Key in-person activities include:

Lecture sessions: Fundamental concepts of the subject are presented with audiovisual support, in a participative environment that encourages interaction both among students and between students and the lecturer.
Problem-solving and practical case sessions: These sessions are aimed at consolidating and applying theoretical content through a variety of methodologies, such as problem-solving, case studies, and project-based work. Collaborative learning is encouraged, promoting the development of key competencies through student cooperation.
Practical sessions with specific technological resources: These sessions focus on developing practical skills related to the knowledge acquired in lectures and problem-solving or case-based activities.

Face-to-face sessions are complemented by a significant amount of independent student work, focused on:

Individual study: Reinforcing the concepts covered in lectures and laboratory sessions.
Individual assignments: Preparing practical works and solving exercises. The lecturer will continuously monitor student progress, both during in-person sessions and through individual or group tutorials.

Additionally, the Canvas will serve as a support platform, providing access to materials, educational resources, activity schedules, and communication channels between students and teaching staff.

DISTRIBUTION OF WORK TIME

TEACHER-LED TRAINING ACTIVITIES	INDIVIDUAL WORK
60 Horas	90 Horas
• (AF1) Lecture sessions. 26h • (AF2) Problem-solving and practical case sessions. 8h • (AF4) Practical sessions with specific technological resources. 22h • (AFE1) Academic monitoring and assessment activities. 4h	• (AFA1) Personal work and independent study. 90h

LEARNING RESULTS

Basic knowledge about the use and programming of computers, operating systems, databases and computer programs with application in engineering.

Knowledge of the structure, organization, operation and interconnection of computer systems, the fundamentals of their programming, and their application to solving engineering problems.

SPECIFIC LEARNING RESULTS

Identifies the object-oriented programming paradigm and its main advantages.

Develops programs that facilitate code reuse.

Creates programs based on the object-oriented programming paradigm, from designing and defining classes and objects to implementation and subsequent debugging, using a specific programming language and development environment.

Designs application interfaces in both console/command-line mode and through graphical interfaces or windowed environments.

Applies methodologies and best practices in the development of object-oriented programs.

Correctly use exceptions in object-oriented programs.

LEARNING APPRAISAL SYSTEM

ORDINARY CALL

The evaluation system includes the following types of assessments:

[1] SE1 - Written or oral examinations (70%) - The main assessment of the course is based on in-person examinations designed to measure both the student's conceptual understanding and their ability to apply acquired knowledge in practical situations. These assessments will be conducted without the aid of external resources and under controlled conditions, ensuring individual authorship and fairness in evaluation.

Minimum pass mark: 5/10.

[2] SE2 - Daily activities, individual and group tasks and exercises (10%) - These assessed activities are aimed at reinforcing active student participation throughout the course. Short exercises will be set, mainly to be completed during class time, allowing for continuous monitoring of content and encouraging ongoing learning. These activities are formative in nature, promoting reflection and the immediate application of covered concepts. Evaluation will be based on participation, submission, and how well the responses align with the stated objectives.

Minimum pass mark: not applicable.
This item will not be marked unless the student has a minimum attendance of 80%.

[3] SE3 - Individual and group projects (20%) - Students will be asked to develop projects that allow for the practical and integrated application of the knowledge gained during the course. These projects will help consolidate competencies related to the design, coding, and organization of applications developed following the object-oriented programming paradigm. Progress will be monitored primarily in class, through partial submissions, to ensure steady progress and the student's direct involvement in the software development process.

Minimum pass mark: 5/10.

EXTRAORDINARY CALL

Students who have not achieved the minimum required mark in any of the assessment items, and have therefore failed the ordinary session, may be eligible for reassessment during the resit extraordinary call.

Assessment items that have been passed will be carried forward to the resit session.
The item [2] SE2 — Daily activities, individual tasks and exercises is non-recoverable.

ACADEMIC EXEMPTION

Students who are exempt from the requirement to attend classes must complete the same assessments ([1] and [3]) as the rest of the class.

The item [2] SE2 — Daily Activities, Assignments and Individual Exercises will not apply to students granted academic exemption. Instead, this percentage will be added to item [3].
The item [3] SE3 — Individual and Group Projects, which will then account for 30% of the final mark.

ACADEMIC INTEGRITY

Any form of fraud or plagiarism committed by a student during an assessment activity will be penalized in accordance with the UFV's Code of Conduct.

Plagiarism includes any attempt to deceive the assessment system, such as copying exercises, exams, practical work, assignments, or any other type of submission, whether from another student or from unauthorised sources or devices, with the intention of leading the lecturer to believe the work is their own.

ETHICAL AND RESPONSIBLE USE OF ARTIFICIAL INTELLIGENCE

1.- The use of any Artificial Intelligence (AI) system or service shall be determined by the lecturer, and may only be used in the manner and under the conditions indicated by them. In all cases, its use must comply with the following principles:
a) The use of AI systems or services must be accompanied by critical reflection on the part of the student regarding their impact and/or limitations in the development of the assigned task or project.
b) The selection of AI systems or services must be justified, explaining their advantages over other tools or methods of obtaining information. The chosen model and the version of AI used must be described in as much detail as possible.
c) The student must appropriately cite the use of AI systems or services, specifying the parts of the work where they were used and describing the creative process followed. The use of citation formats and usage examples may be consulted on the Library website(https://www.ufv.es/gestion-de-la-informacion_biblioteca/).
d) The results obtained through AI systems or services must always be verified. As the author, the student is

responsible for their work and for the legitimacy of the sources used.
2.- In all cases, the use of AI systems or services must always respect the principles of responsible and ethical use upheld by the university, as outlined in the Guide for the Responsible Use of Artificial Intelligence in Studies at UFV. Additionally, the lecturer may request other types of individual commitments from the student when deemed necessary.
3.- Without prejudice to the above, in cases of doubt regarding the ethical and responsible use of any AI system or service, the lecturer may require an oral presentation of any assignment or partial submission. This oral evaluation shall take precedence over any other form of assessment outlined in the Teaching Guide. In this oral defense, the student must demonstrate knowledge of the subject, justify their decisions, and explain the development of their work.

BIBLIOGRAPHY AND OTHER RESOURCES

Basic

Radek Vystaveél C#: Programming Basics for Absolute Beginners 1st edition
(Radek Vystaveél C#: Programming Basics for Absolute Beginners 1st edition , ||Martin Fowler UML Distilled: A Brief Guide to the Standard Object Modeling Language 1st edition)

Additional

Bjarne Stroustrup The C++ Programming Language 4th edition

Ian Griffiths Programming C# 8.0 1st edition
(Ian Griffiths Programming C# 8.0 1st edition , ||Donald E. Knuth The arf of Computer Programming - Volume 1 3rd edition)