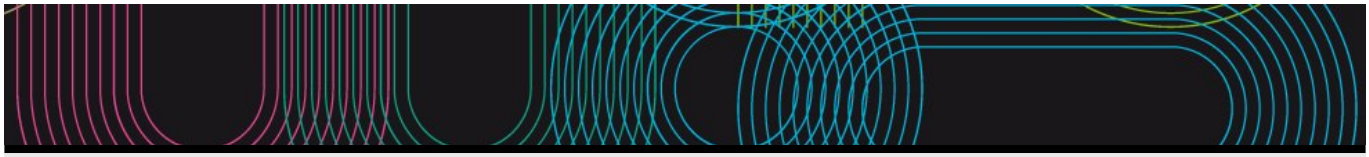

IDENTIFYING DATA				
Programming II				
Subject	Programming II			
Code	V05G300V01302			
Study programme	Degree in Telecommunications Technologies Engineering			
Descriptors	ECTS Credits	Choose	Year	Quadmester
	6	Mandatory	2nd	1st
Teaching language	Spanish			
Department	Telematics Engineering			
Coordinator	Blanco Fernández, Yolanda			
Lecturers	Blanco Fernández, Yolanda Fernández Masaguer, Francisco			
E-mail	yolanda@det.uvigo.es			
Web	http://www.faitic.es			
General description	The general aim of this subject is to provide the students with the theoretical foundations and practical competitions to analyse, design, develop and debug computer applications following the Object-Oriented Programming (OOP) paradigm. Programming II is a mainly practical subject where students have to design and develo pone of several programming projects. With the goal of supporting the students during the development of these software projects, firstly a very brief introduction to the discipline of Software Engineering and its relationship with the OOP paradigm will be given, putting the focus on the stages of analysis, design, implementation and debugging. Next, we will analyse in detail the foundations of OOP, highlight the advantages of UML diagrams for the design tasks that the students will have to carry out. The main contents that will be explained in the subject are the following ones: - Basic concepts of Software Engineering. - Basic concepts of Object-Oriented Programming: classes and objects - Encapsulation. Hiding principle. Concepts of decoupling and cohesion - Inheritance, abstraction, polymorphism and reuse - Relationships between classes: generalisation, association and dependency. - Communication between objects: methods, events, messages. - Persistence. Storage in files and in databases. - Generation, capture and processing of exceptions. - Introduction to the UML modeling language.			

Competencies	
Code	
B6	CG6: The aptitude to manage mandatory specifications, procedures and laws.
B14	CG14 The ability to use software tools to search for information or bibliographical resources.
C50	(CE50/T18)The ability to develop, interpret and debug programs using basic concepts of Object Oriented Programming (OOP): classes and objects, encapsulation, relations among classes and objects, and inheritance.
C51	(CE51/T19) The ability of basic application of phases of analysis, design, implementation and debugging of OOP programs.
C52	(CE52/T20) The ability of manipulation of CASE tools (editors, debuggers).
C53	(CE53/T21) The ability of developing programs considering to the basic principles of software engineering quality taking into account the main existing sources of norms, standards and specifications.

Learning outcomes		
Expected results from this subject	Training and Learning Results	
To understand the basic concepts of Object Oriented Programming (OOP).	B14	C50
To know the main UML diagrams for the documentation in the phases of analysis and design of programs according to the OOP.	B6 B14	C52 C53

To develop skills in the process of analysis, design, implementation and debugging of applications according to the OOP, taking into account the main standards and norms of quality.	B6 B14	C51 C53
To acquire maturity in techniques of development and debugging of programs to allow the autonomous learning of new skills and programming languages.	B6	C51 C52 C53

Contents

Topic	
1. Introduction to the object oriented paradigm	a. Brief introduction to the subject and its organization. b. Birth of the paradigm c. Foundations: classes and objects d. Concepts of encapsulation, inheritance (generalization), and polymorphism e. Brief introduction to UML
2. Encapsulation	a. Classes, interfaces and packages b. Methods and member variables. Visibility. Scope of resolution c. Constructor method d. Parameter passing: pointers and references e. Pointers to objects
3. Inheritance	a. Derived classes and types of inheritance b. Abstract Classes c. Multiple Inheritance d. Object class
4. Object oriented design	a. Design foundations b. Use of UML diagrams
5. Polymorphism	a. Overloading and overwriting b. Abstract classes and interfaces c. Generic classes
6. Exception handling	a. Exceptions foundations b. Handling of Java exceptions

Planning

	Class hours	Hours outside the classroom	Total hours
Lecturing	28	42	70
Problem solving	5	8	13
Autonomous problem solving	6	17	23
Case studies	3	9	12
Project based learning	7	18	25
Case studies	1	0	1
Problem solving	3	0	3
Laboratory practice	2	0	2
Project	1	0	1

*The information in the planning table is for guidance only and does not take into account the heterogeneity of the students.

Methodologies

	Description
Lecturing	Classes that will combine the explanation of the concepts involved in the subject and the performance of small exercises. These may be solved by the teacher or by the students, individually and/or in groups. The aim is to encourage debates in class and strengthen the adquisition of skills. Through this methodology the competencies CE50, CE51 and CE53 are developed.
Problem solving	In the laboratory the professor will show pieces of Java programs with the goal of improving the students' understanding about the main OOP-related concepts. Through this methodology the competences CE50, CE51 and CE53 are developed.
Autonomous problem solving	The students will resolve the practices proposed by the professor. The best solutions and possible doubts that arise will be guided by the professor in order to identify common errors. Through this methodology the competences CE50, CE51, CE53, CG6 and CG14 are developed.
Case studies	The professor will supervise and guide the students during the design of the UML diagramas, with the goal of identifying common errors. Through this methodology the competences CE51 and CE52 are developed.
Project based learning	The students will implement the software system proposed by the teacher during the second part of the course, combining work in the laboratory supervised by the teacher with work out of the laboratory. Through this methodology the competencies CE50, CE53, CG6 and CG14 are developed.

Personalized attention

Methodologies	Description
Problem solving	Revision and comments of solved exercises. Glossary of frequent errors to avoid. Recommendations of style and organization. Tips about best coding practices.
Project based learning	The professor keeps track the level of understanding of the students, supporting them in particular doubts, errors of design and possible improvements in the code.
Autonomous problem solving	Reviewing and comments for each group throughout the development phase, helping them in compilation and understanding of execution-related problems, besides the detection and solution of conceptual errors.
Case studies	Analysis, detection of errors and discussion about possible improvements for students-proposed UML designs.

Assessment

	Description	Qualification	Training and Learning Results
Project based learning	The project consists of the final design (UML diagrams), the Java code and the corresponding documentation. The code must necessarily be compiled and run on the computers of the laboratory. The project can be carried out individually or in groups of 2 people, as per the assessment mechanism chosen by each student. The professor will interview the students in order to check the authorship of their projects and to carry out different functionality tests. In case of groups, both members must attend the interview. The questions posed by the professor must be answered individually by each student in order to corroborate his/her level of understanding and involvement during the project development. Each student must identify the part of the software project he/she has implemented. Students who fail when demostrating their authorship will not pass the subject in the first call. Otherwise, the mark of each student will depend on (i) his/her particular answers during the interview, (ii) the amount of correct functionality tests, and (iii) the quality of the Java coding regarding the adoption of OOP techniques.	35	B6 B14 C50 C53
Case studies	The students will design the software project by the modeling language UML, by including different types of diagrams along with the corresponding documentation to guide the decisions taken. The UML design can be developed individually or in groups of 2 people as per the assessment mechanism chosen by each student. In case of groups, the individual grade of each student will depend on the quality of the UML diagrams delivered.	5	C51 C52
Problem solving	Each student will take □individually and without material of support-- an exam on the official date approved by the Board of School. This exam will combine problems, short-answer questions, multiple choice and true/false tests, which are aimed at assessing the level of understanding of the students on the theoretical concepts explained in the subject.	50	C50 C51 C53
Laboratory practice	This test consists of a set of Java practices for beginners that will help the students to get in touch with a new programming language based on the OOP paradigm. These Java practices will be only delivered by the students who choose the Continuous Assessment mechanism. These practices will be developed in groups of 2 students and the final remark of each of them will depend on (i) his/her individual answers to the questions posed by the professor during a personal interview, and (ii) the quality and correct funcionality of the Java code delivered.	10	C50 C51 C52 C53

Other comments on the Evaluation

There exist two assessment mechanisms in this subject: continuous assessment (CA) and eventual assessment (EA). The students must choose one of them considering the following conditions:

- CA consists of the 4 tests described in the Assessment section of this document (exam, Java practices for beginners, UML design and Java implementatation of the project).
- Students who sit EA must develop the software project (whose specifications will be published at faiTIC platform) individually.
- By the submission of the UML design of the project, students make a commitment to be assessed via CA, thus renouncing the EA mechanism. In virtue of this commitment, the final mark of these students cannot be □Not taken□.
- Students who do not deliver the UML design in time renounce EC mechanism, thus being assessed as per the

requirements of EA. Note that it will be not possible to join the CA in the next tests.

- The schedule of the midterm/intermediate exams will be approved in the Comisión Académica de Grado (CAG) and will be available at the beginning of each academic semester.
- CA tests will be carried out only on the dates defined by the professors. These CA tests cannot be repeated later.
- The grades obtained in the CA and other exams and practical projects are only valid for the current academic year.
- Plagiarism is regarded as serious dishonest behavior. If any form of plagiarism is detected in any of the tests or exams, the final grade will be FAIL (0), and the incident will be reported to the corresponding academic authorities for prosecution.

Students who sit CA in the first call will be assessed as follows:

Theoretical part: Exam (50%). Individual exam without any type of supporting material. It is the third test described in the Assessment section. The grade of this exam can be retained for the second call in case the student gets 4.5 or more points (out of 5 points).

Practical part. It consists of the following tests:

- **Java practices for beginners** (10%). To be developed in groups of 2 people. It is the fourth test described in the Assessment section.
- **Project** (40%). To be developed in groups of 2 people. The project consists of two parts:
    - **UML design** (5%). It is the second test described in the Assessment section.
    - **Java implementation** (35%). It is the first test described in the Assessment section. This part consists of the Java code and the corresponding Javadoc documentation, besides the authorship interview with the professor.

The students must fulfill the following requirements to pass the subject via the CA mechanism:

- To get at least 1/3 of the maximum grade of the theoretical part.
- To get at least 1/3 of the maximum grade of the Java implementation of the project in the practical part.
- To get a final grade (theoretical part + practical part) equal or greater than 5.
- If the final grade is equal or greater than 5 but some of the part does not fulfill the aforementioned minimums, then the final grade will be 4.5 (out of 10 points).

Students who sit EA in the first call will be assessed as follows:

Theoretical part: **Exam** (50%). Individual exam without any type of supporting material. It is the third test described in the Assessment section. The grade of this exam can be saved for the second call in case the student gets 4.5 or more points (out of 5 points) in this test.

Practical part: **Project** (50%). To be developed individually. It is the first test described in the assessment section, including the UML design, the Java code and the corresponding Javadoc documentation, along with the authorship interview driven by the professor.

The students must fulfill the following requirements to pass the subject via the EA mechanism:

- To get at least 1/3 of the maximum grade of the theoretical part.
- To get at least 1/3 of the maximum grade of the practical part.
- To get a final grade (theoretical part + practical part) equal or greater than 5.
- If the final grade is equal or greater than 5 but some of the part does not fulfill the aforementioned minimums, then the final grade will be 4.5 (out of 10 points).

Students will be assessed as follows in the second call:

Theoretical part: **Exam** (50%). Individual exam without any type of supporting material. It is the third test described in the Assessment section. The grade of this exam will never be retained.

 Practical part: Project (50%). In the assessment of this part, three scenarios can be considered as per (i) the

assessment mechanism chosen by the students (CA or EA) , and (ii) the grades obtained in the implementation of the project during the first call. Regardless the particular scenario, the students who sat EA in the first call must deliver the project individually in the second call.

[Scenario #1] The grade of the project is retained for the second call. Although the students can retrieve the grades obtained in the first call, they can also improve their remarks by delivering a new version of the project with additional functionalities (which will be available through faiTIC). In this case, the students must provide a document describing the changes they made in the design of first version of the project for accomplishing the new functionalities.

o This scenario is applicable to students who sat CE, whose grade for the project was equal or greater than 1.5 (out of 4) and whose grade for the UML design was equal or greater than 0.3 (out of 0.5). In case of a new submission, the implementation of the project will be assessed with up to 3.5 points (out of 10) and the UML design with up to 0.5 points (out of 10), since the remark of the Java practices for beginners (up to 1 out of 10 points) will be recovered from the first call.

o This scenario is also valid for students who sat EA, whose grade for the project was equal or equal to 2.5 (out of 5). In case of a new submission, the project will be assessed with up to 5 points (out of 10).

[Scenario #2] The grades obtained in the UML design and Java practices for beginners are retained, and a new project must be delivered in the second call.

o This scenario is valid for students who sat CE whose grade for the project was equal or greater than 1 and lower than 1.5 (out of 4), being their remark for the UML design equal or greater than 0.3 (out of 0.5). The project will be assessed with up to 3.5 points (out of 4).

[Scenario #3] All the grades obtained in the first call are discarded. A new project must be delivered, which will be assessed with up to 5 points.

o This scenario is applicable to students who sat CA and got a grade in the UML design lower than 0.3 (out of 0.5), or a remark in the project lower than 1 (out of 4).

o This scenario is also valid for students who did not deliver project during the first call, and for pupils who did not manage to prove their authorship during the personal interview with the professor.

The students must fulfill the following requirements to pass the subject:

- To get at least 1/3 of the maximum grade of the theoretical part.
- To get at least 1/3 of the maximum grade of the practical part (3.5 in CA and 5 in EA).
- To get a final grade (theoretical part + practical part) equal or greater than 5.
- If the final grade is equal or greater than 5 but some of the part does not fulfill the aforementioned minimums, then the final grade will be 4.5 (out of 10 points).

Students will be assessed as follows in the extraordinary call:

Theoretical part: **Exam** (50%). Individual exam without any type of supporting material. It is the third test described in the Assessment section.

Practical part: **Project** (50%). To be developed individually. It is the first test described in the assessment section. The project consists of the UML design, the Java code and the corresponding Javadoc documentation, besides the authorship interview with the professor.

The students must fulfill the following requirements to pass the subject:

- To get at least 1/3 of the maximum grade of the theoretical part.
- To get at least 1/3 of the maximum grade of the practical part.
- To get a final grade (theoretical part + practical part) equal or greater than 5.
- If the final grade is equal or greater than 5 but some of the part does not fulfill the aforementioned minimums, then the final grade will be 4.5 (out of 10 points).

Sources of information
Basic Bibliography
W. Savitch, **Absolute Java**, 4ª edición, Pearson, 2010
Y. D. Liang, **Introduction to Java programming**, 8ª, Pearson, 2010

P. Deitel, H. Deitel, **Java: How to program**, 9ª, Pearson, 2011

Complementary Bibliography

B. Eckel, **Thinking in Java**, 4ª edición, Prentice-Hall, 2006

P. Niemeyer, D. Leuck, **Learning Java**, 4ª edición, O'Reilly., 2013

Oracle, **Java SE. Oracle**,

Oracle, **Java API Specifications**, 2016

G. Booch, J. Rumbaugh, I. Jacobson, **The Unified Modeling Language User Guide**, 2, Addison-Wesley., 2005

S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, **The Java Tutorial. A short course on the basics**, 4ª edición, Prentice-Hall, 2006

A. Eberhart, S. Fischer, **Java Tools**, Wiley, 2002

M. Page-Jones, **Fundamentals of object-oriented design in UML**, Addison-Wesley, 2002

M. Fowler, **UML Distilled: A Brief Guide to the Standard Object Modeling Language**, 3ª edicion, Addison-Wesley., 2003

Jean-Michel DOUDOUX, **Développons en Java 2.10**, 2016

Recommendations

Subjects that it is recommended to have taken before

Programming I/V05G300V01205