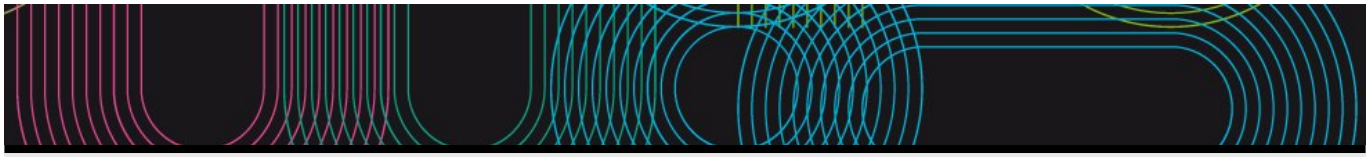

IDENTIFYING DATA				
Programming II				
Subject	Programming II			
Code	V05G300V01302			
Study programme	(*)Grao en Enxeñaría de Tecnoloxías de Telecomunicación			
Descriptors	ECTS Credits	Choose	Year	Quadmester
	6	Mandatory	2nd	1st
Teaching language	Spanish			
Department				
Coordinator	Fernández Masaguer, Francisco			
Lecturers	Blanco Fernández, Yolanda Fernández Masaguer, Francisco Manso Vázquez, Mario Servia Rodríguez, Sandra			
E-mail	francisco.fernandez@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 the practical competitions that allow them analyze, design, develop and debug computer applications following the paradigm of oriented objects programming (OOP). This is an essentially practical subject oriented to the work of the students in the development of software projects. To make this task easier, the subject includes an introduction to the Engineering of the Software. In this sense, the focus is not put on all the well-known phases of the processes of development software (ranging from capture and description of requirements to the deployment of the systems), but just on main stages related to analysis, design, implementation and debugging. Firstly, the engineering of the software is presented as an indispensable discipline for the development of big computer applications, showing the main challenges to face and the basic concepts behind them. Next, the elements of the OOP will be detailed by resorting to UML elements and diagrams, which will be used by the students in their developments. To reach this general aim the contents that will be handled in the subject are the following ones: OOP paradigm: basic concepts, classes and objects. Encapsulation. Concepts of decoupling and cohesion Inheritance, abstraction, polymorphism and reuse Relations between classes: Generalization, 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 Engineering of the Software Concepts of the Engineering of the Software. Historical review or Introduction and concept of Cycle of Life.			

Competencies	
Code	
A6	CG6: The aptitude to manage mandatory specifications, procedures and laws.
A9	CG9: The ability to work in multidisciplinary groups in a Multilanguage environment and to communicate, in writing and orally, knowledge, procedures, results and ideas related with Telecommunications and Electronics.
A59	(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.
A60	(CE51/T19) The ability of basic application of phases of analysis, design, implementation and debugging of OOP programs.
A61	(CE52/T20) The ability of manipulation of CASE tools (editors, debuggers).
A62	(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.
B5	The ability to use software tools to search for information or bibliographical resources

Learning aims		
Expected results from this subject	Training and Learning Results	
To understand the fundamental concepts of the Object Oriented Programming model (OOP) and carry them to practise iusing the most representative object oriented programing languaje (Java).	A9 A59	
To introduce in the use of the UML language, the ISO standard language for software modeling, for the making of structure, behaviour and interaction diagrams, and fundamental for the documentation in the phases of analysis and design of OO programs.	A6 A61 A62	B5
To develop skills in the process of analysis, design, implementation and debugging of OOP applications taking into account the main standards and quality norms.	A60 A62	
To adquiry maturity in development and debugging programming techniques to allow the autonomous learning of new capacities and programming languages.	A62	
To adquiry familiarity with the use of a modern software development tool (Eclipse) to facilitate the design, development and debugging of programs.	A60 A61	

Contents	
Topic	
1. Introduction to OO paradigm	a. Brief introduction to the subject and organization. b. Birth of the paradigm c. Bases: 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 variable member. Visibility. Resolution of field. c. Method constructor d. Step of parameters: 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 Basics b. Use of UML diagrams
5. Polymorphism	a. Overloading and overwriting b. Abstract classes and interfaces c. Generic classes
6. Exception Handling	a. Exception Basics b. Handling Java exceptions

Planning			
	Class hours	Hours outside the classroom	Total hours
Master Session	28	42	70
Troubleshooting and / or exercises	9	9	18
Autonomous troubleshooting and / or exercises	4	10	14
Case studies / analysis of situations	1	1	2
Projects	9	31	40
Case studies / analysis of situations	0	1	1
Troubleshooting and / or exercises	3	0	3
Practical tests, real task execution and / or simulated.	2	0	2
*The information in the planning table is for guidance only and does not take into account the heterogeneity of the students.			

Methodologies	
	Description
Master Session	Classes that will combine explanation of theoretical concepts and realisation of small exercises. These will be able to be resolved by the professor or by the own students individually and/or in group. The goal is to boost the debate in the class and reinforce the acquisition of skills. This methodology is oriented to the acquisition by the student of competences CE50, CE51 and CE53.
Troubleshooting and / or exercises	In the computer rooms, the professor will pose challenges to be resolved by the students, thus discussing collectively the possible options to face a solution. This methodology is oriented to competences CE50, CE51 and CE53.

Autonomous troubleshooting and / or exercises	Students individually will resolve the problems posed by the professor in the computer room. Solutions and doubts that arise in addressing these problems will be put together to agree the best way to fix each concern. This methodology is oriented to competences CE50, CE51, CE53 and A9.
Case studies / analysis of situations	Put in common of the designs posed to solve the project that have to carry out during the second part of the course. The comparison of the different proposals will serve to select the best options and be used, if it is timely, to improve the designs realised. This methodology is oriented to competences CE51 and CE52.
Projects	The students will develop a software project defined by the professor. The development of this project will require face-to-face work in the computer room (supported by the professor) and individual work. This methodology is oriented to competences CE50, CE53, A6 and A9.

Personalized attention

Methodologies	Description
Troubleshooting and / or exercises	The personalized attention will consist of following-up the work of each student, tracking the solutions proposed for each problem posed in the sessions in the computer room and the exhibition of their UML designs for the proposed software project.
Projects	The personalized attention will consist of following-up the work of each student, tracking the solutions proposed for each problem posed in the sessions in the computer room and the exhibition of their UML designs for the proposed software project.
Autonomous troubleshooting and / or exercises	The personalized attention will consist of following-up the work of each student, tracking the solutions proposed for each problem posed in the sessions in the computer room and the exhibition of their UML designs for the proposed software project.
Case studies / analysis of situations	The personalized attention will consist of following-up the work of each student, tracking the solutions proposed for each problem posed in the sessions in the computer room and the exhibition of their UML designs for the proposed software project.

Assessment

	Description	Qualification
Projects	The students, organized into groups of 2 people, will submit a software project during the week from 2 to 6 of December of the course. This submission must include the UML diagrams of the final design, the code and the documentation about the implementation details. The software must run correctly on the computers of the educational laboratories. In the assessment, the professor will consider both the correct execution of the program and the designed adopted in the development. With this test CE53, CE50, A6 (CG6), A9(CG9) and B5 competences will be assessed.	30
Case studies / analysis of situations	The students, organized into groups of 2 people, will submit and present in the computer room the design defined for the project software, including UML class diagrams. This design will be submitted during the week from 4 to 7 of November of the course. With this test CE51, CE52 and A9 competences will be assessed.	10
Troubleshooting and / or exercises	Each student will take a final exam in the official date published in www.teleco.uvigo.es, which will consist of the following types of questions: resolution of problems, short-answer questions about the theoretical concepts explained in master sessions, true/false assessments, multiple-choice tests. Note that support materials are not allowed. The number and the combination of the aforementioned questions will be defined for each particular exam. With this test, CE50, CE51 and CE53 competences will be assessed.	50
Practical tests, real task execution and / or simulated.	The students, organized into groups of 2 people, will submit the Java initiation practices proposed in the computer room. This submission will take place during the week from 21 to 25 of October of the course. With this test, CE50, CE52 and CE53 competences will be assessed.	10

Other comments on the Evaluation

Thee are two modalities in the evaluation of this subject: continuous evaluation (EC) and traditional evaluation (ET). The students will have to choose one of the two modalities taking into account the following restrictions:

- The EC includes the 4 proofs described in the separated evaluation.
- So much by EC as by ET, the students will have to realise a project of laboratory. To facilitate the election of EC or ET the students will have in Faitic platform of the project to realise from the day 20 September.
- In ET the project will realise of individual form.
- The students that opt by the EC will have to deliver in the first week of November, the UML design of the project posed in the laboratory (corresponding to the 3ª proof of evaluation). By means of said delivers the students engage

to be followed the EC and renounce to the ET. From this moment, these students will not be able to appear as "No presented".

- The students that do not deliver the UML design of the project in the week of the 4 to 7 November, renounce to the EC, so that they will be evaluated by means of the mechanism of ET. It does not exist the possibility to add to EC in the following intermediate proofs.
- The proofs of EC will not be in no case recoverable, not being able to repeat out of the dates stipulated by the educational.
- They will not save qualifications (of proofs of EC neither of practical projects or final examinations) of a course to another.
- The EC only will apply in the first announcement, in the rest of announcements governs only the ET.

First announcement. Students that opt by EC. They will be evaluated as follows:

- Theoretical part:
    - Examination written (50%). Individual examination. It corresponds with the proof 3 described in the separated "Evaluation". It will not allow material of support.
- Practical part:
    - Practices of initiation in Java (10%). In groups of 2 students. It corresponds with the proof 4 described in the separated "Evaluation".
    - Project (40%). In groups of 2 students. Divided in two parts:
        - Design (10%). It corresponds with the proof 2 described in the separated "Evaluation".
        - Implementation (30%). It corresponds with the proof 3 described in the separated "Evaluation". This project will have to be delivered individually the first week of December of the educational period. For his evaluation will realise , like previous requirement, a proof or interview of authorship:
            - If the student does not surpass it, the evaluation of the implementation realised by an examination practise.
            - If the student surpasses the proof of authorship, his note of evaluation (that it will be the same for both members of the group) will take into account: correct design, correct functionality, quality of the code and use of technicians of OOP.
- The requirements to approve will be:
    - A minimum of 1/3 on the total in the theoretical part.
    - A minimum of 1/3 on the total in the part of implementation of the project (or 1/3 on the total of the practical examination in his case).
    - A total note (sum of the 4 proofs) equal or upper to 5.

For the proof of authorship of the practical part (that it can suppose individual questions of diverse nature) will be compulsory that the code delivered can be compiled and executed in the teams of the educational laboratories.

First announcement. Students that opt by ET. They will be evaluated as follows:

- Theoretical part:
    - An examination written (whose description coincides with the proof 3 of EC). The result of this examination will suppose 50% of the final qualification. It will not allow material of support.
- Practical part:
    - The realisation of a software project that will suppose the other 50% of the final qualification. Of individual realisation. This project will consist of design (UML diagrams), the Java code and the documentation generated explanatory of the implementation. The note of evaluation will take into account: correct design, correct functionality, quality of the code and use of technicians of OOP model. This project will have to be delivered individually the first week of December.
    - The realisation of an interview with the tutor oriented to determine the authorship of the project. Said interview will take place in the laboratory thelast academic week of the course. If the student does not surpass the proof of authorship will have to go to an examination practise.

- The requirements to approve will be:
    - A minimum of 1/3 on the total in the theoretical part.
    - A minimum of 1/3 on the total in the project or practical examination according to the case.
    - A total note (sum of the 2 proofs) equal or upper to 5.

For the proof of authorship of the practical part (that it can suppose individual questions of diverse nature) will be compulsory that the code delivered can be compiled and executed in the teams of the educational laboratories.

Second announcement / Announcement of end of course / extraordinary Announcement. In this announcement that does not govern the EC. The evaluation will be as follows:

- Theoretical part:
    - An examination written (whose description coincides with the proof 3 of EC). The result of this examination will suppose 50% of the final qualification. It will not allow material of support.

- Practical part:
    - The students that do not deliver the project in the first announcement, will evaluate with an examination of individual programming in the laboratory that will take place in the date fixed by the Board of School for this. The evaluation of this proof will suppose 50% of the final qualification.
    - The part practises to realise for the students that deliver the project in the first announcement, will depend on the note of the project obtained in the first announcement, as the following:
        - *It notices >= 1,5.* It will keep them the note, not having to present to the practical examination of the second announcement. They will be able to, however, improve the punctuation of the project delivering a new version of the one of the first announcement together with the new functions to realise that they published in his moment in the Faitic platform. Likewise, they will have to deliver a document that collect the changes and updates realised to the project on the version that deliver in the first announcement.
        - *Note between 1.5 and 1.* They will be able to opt between going to the examination practise or realise the project of the second announcement. No it keeps them the note of the project of the first announcement, but if the one of initiation and the one of UML design.
        - *It notices < 1.* They will be able to opt between going directly to the examination practise or realise the project expanded. In any case loses the note of practices of the part of initiation and UML design. That is to say, it evaluated them on 5, independently of if they opt by the project or the practical examination.

- The requirements to approve will be:
    - A minimum of 1/3 on the total in the theoretical part.
    - A minimum of 1/3 on the total in the project or examination according to the case.
    - A total note (sum of the 2 proofs) equal or upper to 5.

Sources of information

Basic references:

[2] *Introduction to Java programming*. Y. Daniel Liang, 8ª edición. 2010, Pearson.

Other references:

[1] *Programación orientada a objetos con Java: una introducción práctica usando BlueJ*. D. J. Barnes, M. Kölling. 3ª edición. 2007, Pearson.

[3] *Data Scructures & Algorithms in Java*. Michale T. Goodrich, Roberto Tamassia, 5ª edición. 2010, Willey.

[4] *Java Tools*. Andreas Eberhart, Stefan Fischer. 2002, Wiley

[5] *Java In A Nutshell*. David Flanagan, 5ª edición. 2005, O'Reilly.

[6] *Thinking in Java*. Bruce Eckel, 4ª edición. 2006, Prentice Hall

[7] *Learning Java*. Patrick Niemeyer, 3ª edición. O'Reilly Media

[8] *How to Think Like a Computer Scientist. JavaTM Version*. 4ª version. Online: http://www.greenteapress.com/thinkapjava/

[9] *Java notes*. Fred Swartz. Online: http://www.leepoint.net/notes-java/index.html

[10] *Java SE. Oracle*. Online: http://www.oracle.com/technetwork/java/javase/downloads/index.html

[11] *Java 2 Platform Standard Edition 5.0. API Specification*. Online: http://download.oracle.com/javase/1.5.0/docs/api/

[12] *The Java Tutorials*. Oracle. Online: http://download.oracle.com/javase/tutorial/

[14] *Open-oriented Analysis and Design with Applications*. Grady Booch, Robert Maksimchuk, Michael Engel, Bobbi Young, Jim Conallen, Kelli Houston, 3ª edición. 2007, Addison Wesley.

[17] *Fundamentals of Object-oriented design in UML*. Meilir Page-Jones. 2002, Addison Wesley.

Recommendations

Subjects that it is recommended to have taken before

Programming I/V05G300V01205