



## IDENTIFYING DATA

### Programming II

|                   |  |           |      |
|-------------------|--|-----------|------|
| Subject           | Programming II   |           |      |
| Code              | V05G300V01302  |           |      |
| Study programme   | Degree in Telecommunications Technologies Engineering                                  |           |      |
| Descriptors       | ECTS Credits   | Choose    | Year |
|                   | 6  | Mandatory | 2nd  |
| Teaching language | Spanish  |           |      |
| Department        |  |           |      |
| Coordinator       | Fernández Masaguer, Francisco  |           |      |
| Lecturers         | Blanco Fernández, Yolanda<br>Caeiro Rodríguez, Manuel<br>Fernández Masaguer, Francisco |           |      |
| E-mail            | francisco.fernandez@det.uvigo.es   |           |      |
| Web               | <a href="http://www.faitic.es">http://www.faitic.es</a>                                |           |      |

**General description** The general aim of this subject is to provide to the student the theoretical foundations and the practical competitions that allow him analyse, design, develop and debug computer applications following the paradigm oriented to objects. This is an essentially practical subject and in this sense is oriented to the work of the students in the realisation of one or several projects.

To facilitate the development of the projects, in the subject will realise firstly a very brief introduction to the discipline of Software Engineering , linking it with the paradigm of the object oriented programming (OOP) and restricting it only to the stages of analysis, design, implementation and debugging. Then we will analyse in detail the elements of OOP, using elements and UML diagrams that they will be used by the students in his developments.

To reach this general aim the contents that will be handled in the subject can be summarized in the following items:

- 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
- Relations 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 |                   |
|---|-------------------------------|-------------------|
|   | B14                           | C50               |
| 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<br>B14                     | C52<br>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<br>B14                     | C51<br>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<br>C52<br>C53 |

### Contents

| Topic   |   |
|---|---|
| 1. Introduction to the object oriented paradigm | <ul style="list-style-type: none"> <li>a. Brief introduction to the subject and its organization.</li> <li>b. Birth of the paradigm</li> <li>c. Foundations: classes and objects</li> <li>d. Concepts of encapsulation, inheritance (generalization), and polymorphism</li> <li>e. Brief Introduction to UML</li> </ul> |
| 2. Encapsulation                                | <ul style="list-style-type: none"> <li>a. Classes, interfaces and packages</li> <li>b. Methods and member variables. Visibility. Scope of resolution</li> <li>c. Constructor method</li> <li>d. Passing parameters: pointers and references</li> <li>e. Pointers to objects</li> </ul>                                  |
| 3. Inheritance                                  | <ul style="list-style-type: none"> <li>a. Derived classes and types of inheritance</li> <li>b. Abstract Classes</li> <li>c. Multiple Inheritance</li> <li>d. Object class</li> </ul>  |
| 4. Object oriented design                       | <ul style="list-style-type: none"> <li>a. Design foundations</li> <li>b. Use of UML diagrams</li> </ul>   |
| 5. Polymorphism                                 | <ul style="list-style-type: none"> <li>a. Overloading and overwriting</li> <li>b. Abstract classes and interfaces</li> <li>c. Generic classes</li> </ul>  |
| 6. Exception handling                           | <ul style="list-style-type: none"> <li>a. Exceptions foundations</li> <li>b. Handling of Java exceptions</li> </ul>   |

### 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 | <p>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 acquisition of skills.</p> <p>Through this methodology the competencies CE50, CE51 and CE53 are developed.</p> |

|   |   |
|---|---|
| Troubleshooting and / or exercises            | In the laboratory, the teacher will pose small challenges to be solved collectively so that the underlying concepts and the different options of resolution can be discussed, and to provide students with the skills object of the subject.<br>Through this methodology the competencies CE50, CE51 and CE53 are developed.                                |
| Autonomous troubleshooting and / or exercises | Students will solve independently the problems posed by the teacher in the laboratory. The solutions and the doubts that arise in addressing these problems will be put together to agree the best way of resolution.<br>Through this methodology the competencies CE50, CE51, CE53, CG6 and CG14 are developed.  |
| Case studies / analysis of situations         | Putting in common of the designs proposed by the students to solve the project to be carried out during the second part of the course. The comparison of the different proposals will serve to select the best options and as a feedback, if appropriate, to improve the designs.<br>Through this methodology the competencies CE51 and CE52 are developed. |
| Projects                                      | 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.<br>Through this methodology the competencies CE50, CE53, CG6 and CG14 are developed.  |

### Personalized attention

| Methodologies                                 | Description  |
|---|--|
| Troubleshooting and / or exercises            | Revision and comments of solved exercises. Glossary of frequent errors to avoid. Recommendations of style and organization.  |
| Projects                                      | Beside commenting with the group, different alternatives, recommendations and strategies for the good realization of the project, we examine with group members the level of understanding of the project, particular doubts that can arise, errors of design and code and options of improvement. |
| Autonomous troubleshooting and / or exercises | Review and comments with each group of the diverse practise proposed during his development phase. Help for compilation and execution errors. Detection and solution of conceptual errors.   |
| Case studies / analysis of situations         | Review and general critic of the UML design UML of each group during his development.  |

### Assessment

|          | Description  | Qualification | Training and Learning Results |
|----------|--|---------------|-------------------------------|
| Projects | <p>Students, organised in groups of two, will deliver the proposed software project at most the day 5 January. This will consist of his final design (UML diagrams ), the code and the documentation generated explanatory of the implementation. That the code delivered can be compiled and executed in the laboratory computers will be necessary condition to surpass this proof of evaluation.</p> <p>During the last week of the course, the students will have an interview with the professor in the schedule of the laboratory, devoted to show the authorship of the project and realise diverse proofs of functionality. Both members of the group must attend the interview. The issues raised therein will have to be answered individually to be able to ascertain the degree of understanding and implication of the student in the project developed, owing each student identify the parts of the project that has implemented. The adequate answers will be used to establish, together with a group of tests of functionality and the analysis of the quality of the code, the individual cualification of each student.</p> <p>In case that a student do not accredit properly the authorship, no will give him by cost the project, and considered suspense in the corresponding announcement.</p> <p>For the students that accredit properly the authorship, the evaluation of the project will take into account so much the proportionate answers in the authorship interview, like the correct functionality, like the quality of the code and the use of the technicians of the programming oriented to objects. The determination of the correct functionality will realise by means of a group of around 50 tests on the software delivered.</p> | 30            | B6 C50<br>B14 C53             |

|  |   |    |                          |
|--|---|----|--------------------------|
| Case studies / analysis of situations                    | At the end of the 9 <sup>a</sup> week of the academic course the students, organised in groups of two, will deliver the UML design of a project software.<br><br>In lective hours the members of each group will do with the professor a brief authorship interview of this design, which together with the delivered design, will be used to establish the individual note of each student.  | 10 | C51<br>C52               |
| Troubleshooting and / or exercises                       | Written exam to be done individually in the date published in <a href="http://www.teleco.uvigo.es">www.teleco.uvigo.es</a> for this purpose, which will consist of the combination of the following types of questions: resolution of problems, short questions about the theoretical concepts explained in the master sessions, to justify reasonably if one or more statements are true or false, small tests about theoretical and application aspects. The number and combination of these questions will be defined for each particular exam. Support materials (notes, books, collections of problems) are not allowed. | 50 | C50<br>C51<br>C53        |
| Practical tests, real task execution and / or simulated. | At the end of the 7th week of the academic course the students, organized in pairs, will submit the Java initiation practices proposed in the laboratory.<br><br>In lective hours the members of each group will realise with the professor a brief authorship interview of the initiation practices, which together with a group of tests of correct operation on the software delivered, will be used to establish the individual note of each student.   | 10 | C50<br>C51<br>C52<br>C53 |

### Other comments on the Evaluation

There are two modalities of evaluation of this subject: continuous evaluation (CE) and traditional evaluation (TE). The students will have to choose one of the two modalities taking into account the following conditions:

- The CE includes the 4 proofs described in the evaluation section.
- Whether they opt for the CE or for the TE, students must develop a project. To facilitate the choice between CE and TE, the specifications of the project will be available in Fatic the 4th week of the academic course.
- In the TE the project will be carried out individually.
- The students that choose the CE will submit at the end of the 9th week of the academic course the UML design of the project (corresponding to the 2nd proof described in the evaluation section). By means of this submission the students agree to follow the CE and reject the TE. From this moment these students may not appear as if they have not taken the subject.
- The students that do not submit the UML design of the project in the stipulated date reject the CE, so that they will be evaluated by means of the modality of TE. It is not possible to join the CE in the following intermediate proofs.
- The proofs of CE are not recoverable in any case, and they can not be repeated outside the dates stipulated by the teachers.
- Marks (of proofs of CE or practical projects or exams) are not saved from one course to another.

**First announcement. Students that opt for the CE.** They will be evaluated as follows:

□ Theoretical part:

- Written exam (50%). Individual exam. It corresponds to the 3rd proof described in the evaluation section. The mark of this exam only will be saved for the second announcement if it is equal or higher than 4.5 over 5.

□ Practical part:

- Practices of initiation in Java (10%). To be done in pairs. It corresponds to the 4th proof described in the evaluation section.
- Project (40%). To be done in pairs. It is divided in two parts:
  1. Design (10%). It corresponds to the 2nd proof described in the evaluation section.
  2. Implementation (30%). It corresponds to the 1st proof described in the evaluation section.

The requirements to pass will be:

- A minimum of 1/3 of the total in the theoretical part.

- A minimum of 1/3 of the total in the part of implementation of the project (or 1/3 of the total of the practical exam according to the case).
- A total mark (sum of the 4 proofs) equal or higher than 5.
- If the total mark is equal or higher than 5 but the minimum in some part has not been reached, the final mark will be 4.5 points (failure).

**First announcement. Students that opt for the TE.** They will be evaluated as follows:

□ Theoretical part:

- Written exam (50%). Individual exam. It corresponds to the 3rd proof described in the evaluation section. The mark of this exam only will be saved for the second announcement if it is equal or higher than 4.5 over 5.

□ Practical part:

- Individual realization of a software project that will suppose the remaining 50% of the final mark. This project will consist of the design (UML diagrams), the Java code and the generated documentation about the implementation details. The evaluation will take into account correct design, correct functionality, quality of the code and use of techniques of OOP. It must be submitted before January 6.

- Realization of an interview with the teacher with the aim of proving the authorship of the project. This interview will take place in the laboratory hours during the last week of the course. If the student does not accredit properly the authorship it will not surpass the announcement, and will have to do the corresponding project to the second announcement.

□ The requirements to approve will be:

- A minimum of 1/3 of the total in the theoretical part.
- A minimum of 1/3 of the total in the part of the project (or 1/3 of the total of the practical exam according to the case).
- A total mark (sum of the 2 proofs) equal or higher than 5.
- If the total mark is equal or higher than 5 but the minimum in some part has not been reached, the final mark will be 4.5 points (failure).

**Second announcement.** The students will be evaluated as follows:

□ Theoretical part:

- Written exam (50%). Individual exam. It corresponds to the 3rd proof described in the evaluation section. The mark of this exam will not be never saved for others convocatories.

□ Practical part:

It will depend on whether the student has delivered or not the project in the first call. For the students that have followed the CE in the first call, it will be considered that a student has delivered the project when, as least, he/she has submitted an UML design in which he/she has obtained a mark equal or higher than 0.6 of 1.

- The students that do not deliver the project in the first announcement or that have not surpassed the authorship interview, will have to do necessarily the extended project of the second announcement. In any case lose the notes of the parts of initiation in Java and UML design if they opted for the CE in the first announcement, that is to say, will be evaluated on 5.

- The practical part to be done for the students that deliver the project in the first call will depend on the mark obtained in the project in that call, as follows:

- *Mark  $\geq 1.5$  with CE or Mark  $\geq 2.5$  with TE.* They will keep the mark. However, they will be able to improve the mark of the project delivering a new version of the one of the first call together with the new functionalities to be done, that will be published in Fatic. In the same way, they will deliver a document that addresses the changes and updates in the project from the version delivered in the first call.
- *Mark between 1 and 1.5 by CE or Mark between  $5/3 < 2.5$  with TE.* Must do necessarily the extended project of the second announcement. They will not keep the mark of the project of the first call, but they will keep the marks of the parts of initiation in Java and UML design if they have opted for the CE in the first call.
- *Mark  $< 1$  with CE or Mark  $< 5/3$  by TE.* Must do necessarily the extended project of the second announcement. They will not keep the marks of the parts of initiation in Java and UML design if they have opted for the CE in the first call,

that is, they will be evaluated on 5.

□ Requirements to pass. The requirements to pass will be:

- A minimum of 1/3 of the total in the theoretical part.
- A minimum of 1/3 of the total of the project without taking into account the marks of the parts of initiation in Java and UML design if they have opted for the CE in the first call (or 1/3 of the total of the practical exam according to the case).
- A total mark (sum of all the proofs) equal or higher than 5.
- If the total mark is equal or higher than 5 but the minimum in some part has not been reached, the final mark will be 4.5 points (failure).

---

## Sources of information

---

Basic references:

- [1] □Absolute Java□. W. Savitch, 4th edition. 2010, Pearson.
- [2] □Introduction to Java programming□. Y. D. Liang, 8th edition. 2010, Pearson.
- [3] □Java: How to program□. P. Deitel, H. Deitel, 9th edition. 2011, Pearson.

Additional references:

- [1] □Programación orientada a objetos con Java: Una introducción práctica usando Blue□. D. J. Barnes, M. Kölling, 3rd edition. 2007, Pearson.
- [2] □The Java Tutorial. A short course on the basics□. S. Zakhour, S. Hommel, J. Royal, I. Rabinovitch, T. Risser, M. Hoeber, 4th edition. 2006, Prentice-Hall.
- [3] □Data Structures & Algorithms in Java□. M. T. Goodrich, R. Tamassia, 5th edition. 2010, Willey.
- [4] □Java Tools□. A. Eberhart, S. Fischer. 2002, Wiley.
- [5] □Java in a Nutshell□. D. Flanagan, 5th edition. 2005, O'Reilly.
- [6] □Thinking in Java□. B. Eckel, 4th edition. 2006, Prentice-Hall.
- [7] □Learning Java□. P. Niemeyer, D. Leuck, 4th edition. 2013, O'Reilly.
- [8] □How to Think Like a Computer Scientist. Java™ Version□, 4th version. Online: <http://www.greenteapress.com/thinkajava/>
- [9] □Java notes□. F. 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/>
- [13] □Ingeniería del Software orientada a objetos con UML, Java e Internet□. A. Weitzenfeld. 2005, Thomson.
- [14] □Open-oriented Analysis and Design with Applications□. G. Booch, R. Maksimchuk, M. Engel, B. Young, J. Conallen, K. Houston, 3rd edition. 2007, Addison-Wesley.
- [15] □The Unified Modeling Language User Guide□. G. Booch, J. Rumbaugh, I. Jacobson, 2nd edition. 2005. Addison-Wesley.
- [16] □UML Distilled: A Brief Guide to the Standard Object Modeling Language□. M. Fowler, 3rd edition. 2003, Addison-Wesley.
- [17] □Fundamentals of object-oriented design in UML□. M. Page-Jones. 2002, Addison-Wesley.

---

## Recommendations

---

**Subjects that it is recommended to have taken before**

---

