Learning objectives of the subject

To achieve that whoever studies this subject is capable of:
1. Facing successfully the work in a team that confronts the accomplishment of a program of moderate / high complexity;
2. Carry out object-oriented analysis tasks, object-oriented design using correctly software design patterns;
3. Develop automated program testing environments.
4. Add to the programs the level of persistence of data using relational databases. Be able to successfully face work within a team that confronts the completion of a program of moderate / high complexity. Be able to carry out tasks of: object-oriented analysis, object-oriented design using correctly software design patterns. Be able to develop automated testing environments for programs. Be able to implement a level of data persistence using relational databases.

Study load

<table>
<thead>
<tr>
<th>Total learning time: 125h</th>
<th>Hours large group: 26h</th>
<th>20.80%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hours small group: 13h</td>
<td>10.40%</td>
</tr>
<tr>
<td></td>
<td>Self study: 86h</td>
<td>68.80%</td>
</tr>
</tbody>
</table>
## Content

<table>
<thead>
<tr>
<th>Introduction to software tests. JUnit.</th>
<th><strong>Learning time:</strong> 8h</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Theory classes: 3h</td>
</tr>
<tr>
<td>Testable software design.</td>
<td>Laboratory classes: 3h</td>
</tr>
<tr>
<td>Unit tests: JUnit</td>
<td>Self study: 2h</td>
</tr>
<tr>
<td>Simulation of external components (&quot;Fakes&quot; and &quot;Mocks&quot;). Mockito Tool.</td>
<td></td>
</tr>
<tr>
<td><strong>Related activities:</strong></td>
<td></td>
</tr>
<tr>
<td>Lab session. Generation of tests for checking correctness of the code of a class.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td></td>
</tr>
<tr>
<td>Introduce the code testing activities in the context of the software development process. Introduce tools that facilitate these activities. Use JUnit as a tool that allows you to automate software tests</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterative software development processes. Introduction to Scrum</th>
<th><strong>Learning time:</strong> 3h</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>Theory classes: 2h</td>
</tr>
<tr>
<td>Concept of software development process. Main characteristics of iterative processes. Comparison with cascade processes. Introduction to Scrum.</td>
<td></td>
</tr>
<tr>
<td><strong>Related activities:</strong></td>
<td>Self study: 1h</td>
</tr>
<tr>
<td>Development of a program of moderate complexity within a team.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td></td>
</tr>
<tr>
<td>Briefly introduce the concept of software development process as a set of activities carried out by a team whose objective is the development of a program. Show the main characteristics of iterative processes as an alternative to cascade processes, highlighting the strong temporal overlap of analysis, design, coding and testing. Show details of one of the existing iterative processes: Scrum.</td>
<td></td>
</tr>
</tbody>
</table>
## Introduction to object oriented analysis

**Learning time:** 16h 30m  
Theory classes: 4h 30m  
Self study: 12h

**Description:**  
Object oriented analysis artifacts: repertoire of use cases and conceptual model of the problem. Requirements concept. Types of requirements. Requirements capture techniques. Use cases as a method for capturing requirements. Concept and parts of a use case. Building the conceptual model taking as basis the use cases. Identification of the classes. Identification of the attributes of the classes. Identification and complete definition of the relationships between classes.  
Examples

**Related activities:**  
Examples and project developed within a team

**Specific objectives:**  
Categorize the different types of requirements of a program. Be able to identify use cases. Be able to fully develop the most relevant use cases.  
Be able to construct the conceptual model from the use cases: concepts / classes, attributes and relationships.

## Introduction to object-oriented design

**Learning time:** 16h 30m  
Theory classes: 4h 30m  
Self study: 12h

**Description:**  
Object-oriented design artifacts: class diagrams, interaction diagrams, documentation of designed classes. Introduction to the interaction diagrams in UML: sequence diagrams and collaboration diagrams. System sequence diagram  
Techniques to identify and specify the methods of the classes of the program. GRASP patterns to identify methods and assign them to the relevant class. Use of GRASP patterns to construct interaction diagrams from the use cases.

**Related activities:**  
Examples and project carried out within a team

**Specific objectives:**  
Be able to build UML sequence diagrams. Be able to build the sequence diagram of a system. Be able to correctly use the GRASP patterns of assignment of responsibilities to the classes starting at the use cases for identifying the methods of the classes that are designed.
### Introduction to databases

**Description:**

**Related activities:**
Specific lab session and project carried out within a team

**Specific objectives:**
To know what a relational database is. Be able to design simple tables of relational databases. Understand the basic elements of an Entity-Relationship diagram. Know introductory elements of the SQL language. Connect a Java program with a SQL database

**Learning time:** 6h  
Theory classes: 2h  
Laboratory classes: 2h  
Self study: 2h

### Completing object-oriented design with design patterns

**Description:**
Introduction of software design pattern concept. GoF patterns. Classification.

**Related activities:**
Theory session

**Specific objectives:**
Understand the concept of software design pattern, its types and the benefits that its use entails in a design.

**Learning time:** 2h  
Theory classes: 1h  
Self study: 1h

### Study of some relevant design patterns

**Description:**
Study of some of the most relevant design patterns: Strategy, Decorator, Factory Method, Abstract Factory, Compound, Observer

**Related activities:**
Theory sessions  
Lab sessions for some patterns  
Project carried out in a group

**Specific objectives:**
To know the details of the studied patterns. Be able to correctly postulate the use of software design patterns in the design phase of a program. Be able to correctly implement the studied patterns.

**Learning time:** 29h  
Theory classes: 8h  
Laboratory classes: 8h  
Self study: 13h
Pattern aggregation: the model-view-controller pattern

**Description:**
Study of the model-view-controller pattern as an aggregation of several of the design patterns studied above.

**Related activities:**
Theory session and project developed within a team

**Specific objectives:**
Learn the details of the model-view-controller as the result of adding several of the design patterns studied in the previous topic. Implement this pattern aggregation correctly.

**Learning time:**
- Theory classes: 2h
- Self study: 1h

Project

**Description:**
content english

**Learning time:**
- Self study: 41h

Qualification system

Partial deliveries of activities related to the project: 10%
Exam: 20%
Project: 60%
Subjective assessment: 10%

Regulations for carrying out activities

The exam will be done on the computer. The use of electronic messaging systems of any kind will not be allowed.
Bibliography

Basic:


Others resources:

Hyperlink


https://www.infor.uva.es/mlaguna/is1/materiales/BookDraft1.pdf

Design patterns

https://www.oodesign.com/

Java design patterns - example tutorial

https://www.journaldev.com/1827/java-design-patterns-example-tutorial