

270086 - CAP - Advanced Programming Concepts

Coordinating unit: 270 - FIB - Barcelona School of Informatics
Teaching unit: 723 - CS - Department of Computer Science
Academic year: 2019
Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Teaching unit Optional)
ECTS credits: 6
Teaching languages: Catalan

Teaching staff

Coordinator: - Jordi Delgado Pin (jdelgado@cs.upc.edu)

Prior skills

Students should have enough knowledge of data structures and algorithms and object oriented programming and design.

Requirements

- Prerequisite IES
- Prerequisite PROP

Degree competences to which the subject contributes

Specific:
CES1.1. To develop, maintain and evaluate complex and/or critical software systems and services.

CES1.3. To identify, evaluate and manage potential risks related to software building which could arise.

CES1.7. To control the quality and design tests in the software production

General:
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Teaching methodology

Teaching the course is structured in lectures and laboratory sessions.

Teachers will use lectures to introduce the essential contents of the course. In the laboratory sessions the contents of the course will be brought to the computer by carrying out practical problems. The laboratory classes will be a continuation of the lectures, where new concepts will be implemented as they appear in lectures.

Learning objectives of the subject

1. Review the concepts of object-oriented programming in order to ensure a common knowledge base from which to proceed with the specific subject matter of the course and to know a dynamic and object oriented programming language such as Smalltalk.
2. Learn the basics of computational reflection, so that the student is able to understand how these concepts are implemented in different programming languages
3. Knowing how computational reflection is implemented in Java and Smalltalk: accessing and modifying members of classes at runtime, creating code at runtime, etc..
4. Being able to develop a computer program of small-middle size that uses reflection techniques
5. Learn the basic concepts of prototype based programming, so that the student is able to understand how these concepts are implemented in different programming languages
6. Learn a language based on prototypes so that the student is able to get the general concepts in a concrete implementation (JavaScript)
7. Knowing what a closure is and some techniques associated with their use. Understanding what Continuations are and using them to implement several control structures

<table>
<thead>
<tr>
<th>Study load</th>
<th>Hours large group:</th>
<th>Hours medium group:</th>
<th>Hours small group:</th>
<th>Guided activities:</th>
<th>Self study:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total learning time:</td>
<td>150h</td>
<td>30h</td>
<td>0h</td>
<td>30h</td>
<td>84h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.00%</td>
<td>0.00%</td>
<td>20.00%</td>
<td>56.00%</td>
</tr>
</tbody>
</table>
Review of object oriented programming and Introduction to Smalltalk

Degree competences to which the content contributes:
Description:
Philosophy and origins. Encapsulation, members private / (protected) / public, inheritance, polymorphism, early / late binding. Patterns. Smalltalk, learning a Smalltalk environment

Reflection: General concepts

Degree competences to which the content contributes:
Description:

Reflection in Java and Smalltalk

Degree competences to which the content contributes:
Description:

Prototype based programming

Degree competences to which the content contributes:
Description:
Classes vs. prototypes. Inheritance vs. delegation

Control Structures: Continuations

Degree competences to which the content contributes:
Description:
Continuations are the result of being able to reify and eventually modify the execution stack of a program while it is running, but from the same language. We will explore the possibilities that this offers and the associated techniques.
## Planning of activities

| Review of Object Oriented Programming: Generalities | Hours: 2h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 0h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.</td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>1</td>
</tr>
</tbody>
</table>

| Introduction to Smalltalk                         | Hours: 18h  
Theory classes: 6h  
Practical classes: 0h  
Laboratory classes: 6h  
Guided activities: 0h  
Self study: 6h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.</td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>1</td>
</tr>
</tbody>
</table>

| Reflection: General concepts                      | Hours: 16h  
Theory classes: 2h  
Practical classes: 0h  
Laboratory classes: 2h  
Guided activities: 2h  
Self study: 10h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.</td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>2</td>
</tr>
</tbody>
</table>
## Reflection in Java (java.lang.reflect) and Smalltalk

**Description:**
The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.

**Specific objectives:**
3, 4

### Hours
- Theory classes: 8h
- Laboratory classes: 8h
- Guided activities: 0h
- Self study: 20h

## Practical exercise

**Description:**
Submission date: End of semester.

**Specific objectives:**
4

### Hours
- Guided activities: 0h
- Self study: 0h

## Theory test

**Specific objectives:**
2, 3

### Hours
- Guided activities: 2h
- Self study: 10h

## Prototype based programming

**Description:**
The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.

**Specific objectives:**
5, 6, 7

### Hours
- Theory classes: 6h
- Laboratory classes: 6h
- Guided activities: 0h
- Self study: 14h
Continuations

**Hours:** 28h  
Theory classes: 6h  
Practical classes: 0h  
Laboratory classes: 6h  
Guided activities: 2h  
Self study: 14h

**Description:**  
The student should pay attention to the lecture and he/she should work through the exercises suggested by the lecturer.

**Specific objectives:**  
7

Final test

**Hours:** 12h  
Guided activities: 2h  
Self study: 10h

**Specific objectives:**  
5, 6, 7

Qualification system

Grading the course will consist of two theoretical tests (T1 and T2), one mid-course and the other at the end, and one medium-sized practical work (Practice).

Then, the evaluation method would be:  
0.6 * Theory + 0.4*Practice  
where:  
Theory: 0.5 * T1 + 0.5 * T2

Teamwork:

Evaluated using a simple rubric that each group tutor group uses to rank different aspects of teamwork of every member of the group.
Bibliography

Others resources:

Hyperlink

http://download.oracle.com/javase/6/docs/technotes/guides/reflection/index.html

https://developer.mozilla.org/es/JavaScript

http://www2.parc.com/csl/groups/sda/projects/reflection96/docs/rivard/rivard.html