Software Engineering Methods 2014

Instructor: Dr. Alberto Bacchelli — Assignment 1

Teaching Assistants: Friso Abcouwer, Rob van Bekkum, Moritz Beller, Thijs Boumans, Aimee Ferouge, Jan Giesenberg, Michael de Jong — Week 03

The goal of this assignment is to apply Responsibility Driven Design, Software Modeling using UML, and Advanced Object Oriented Programming principles. To correctly complete this assignment you **must**:

- Carry out the assignment with your team only–unless otherwise stated. You are free to discuss solutions with other teams, but each team should come up its own personal solution. A strict plagiarism policy is going to be applied on all the artifacts submitted for evaluation.
- Your team has to complete the assignment by following the SCRUM methodology:
  - You have to submit a sprint plan for this assignment, using the template "Sprint Plan Template" available on Blackboard, by **Sep 16, 2014 @ 13:00**.
  - You have to submit a sprint reflection for this assignment, using the template "Sprint Reflection Template" available on Blackboard, by **Sep 23, 2014 @ 13:00**.
- Solutions to this assignment will consist (depending on the exercise) in changes to the source code of your project and in explanations (*e.g.*, of decisions taken):
  - The explanations must be written in a PDF file with the name: *Group[devhub id]-[AssignmentNumber].pdf* (a correct name would be: *Group1-1.pdf*).
  - Changes and explanations must be pushed to the master branch of your Devhub repository by **Sep 20, 2014 @ 23:55**.[1]

Exercise 1 - The Core (14 pts)

1. Following the Responsibility Driven Design, describe the *main* classes you implemented in your project in terms of responsibilities and collaborations (**4 pts**).
2. Why do you consider the other classes as less important? Following the Responsibility Driven Design, reflect if some of those non-main classes have similar/little responsibility and could be changed, merged, or removed. If so, perform the code changes; if not, explain why you need them (**4 pts**).
3. Draw the class diagram of the aforementioned main elements of your game (do not forget to use elements such as parametrized classes or association constrains, if necessary) (**2 pts**).
4. Draw the sequence diagram to describe how the main elements of your game interact (consider asynchrony and constraints, if necessary) (**4 pts**).

Exercise 2 - Theory in practice (14 pts)

1. What is the difference between aggregation and composition? Where are composition and aggregation used in your project? Describe the classes and explain how these associations work (**2 pts**).
2. Draw the class diagrams for *all* the hierarchies in your source code. Explain why you created these hierarchies and classify their type (*e.g.*, "Is-a" and "Polymorphism"). Considering the lectures, are there hierarchies that should be removed? Explain and implement any necessary change (**2 pts**).
3. Where do you use *if* or *case* statements in your source code? Refactor them (*e.g.*, using double dispatch) so that they are not necessary anymore, and describe your refactoring. If they cannot be refactored (very rare case!) explain why (**10 pts**).

[1]Solutions sent within the first 24 hours after the deadline will be given 50% of the points they would normally get. Solutions sent after 24 hours from the deadline will not be graded.

Exercise 3 - Play with your friends (60 pts)

1. Extend your implementation of the game to support multiplayers. The game has to allow at least two players playing using two computers connected through the network. Define your requirements and get them approved by your teaching assistant. The implementation and process will be based on the same criteria used for the working version (see previous rubrics on blackboard) (**50 pts**).
2. During the analysis and design phases of this extension use responsibility driven design and UML (push to the repository any document produced) (**10 pts**).