Software Construction HS 2021

Instructor: Prof. Dr. Alberto Bacchelli — Assignment 2

Teaching Assistants: Enrico Fregnan, Robin Baettig, Jan Bauer, Deborah Jakobi, Matthias Mylaeus, Joel Ruettiman, Lukas Zehnder — Week 05

To correctly complete this assignment you **must**:

- Carry out the assignment with your team only (unless otherwise stated). You are allowed 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 to all the artifacts submitted for evaluation.
- Provide solutions to the exercises. Each solution will consist of source code and its changes and/or explanations (*e.g.*, of decisions taken):
  - The explanations must be written in a PDF file with the name: *Group[id on OLAT]-a[AssignmentNumber].pdf*[1]
  - Source code and its changes, as well as explanations must be pushed to the master branch of your GitHub repository by **Nov 09, 2021 @ 16:00**[2] and the tutor who follows your group has to be tagged in the commit they need to consider.

---

[1] *e.g.*, a correct name would be: *Group1-a2.pdf*.

[2] 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 1 - A Checkers Game – Getters and Setters: OUT! (45 pts)

In class we discussed the importance of *information hiding*, *data abstraction*, and *encapsulation* in object-oriented programming. From this discussion, it derives that having getters and setters (also known as accessors and modifiers) or public variables for objects is not a good practice.

1. Where are *getters/setters* or *public variables* used[3] in your source code? Refactor *nine*[4] of these cases, so that they are not necessary anymore and describe your refactoring/redesign.

   If you find some rare cases of *getters/setters* or *public variables* statements that should/cannot be refactored, you can convincingly explain why (we recommend to consult with your tutor before going this way); these cases count toward the nine you have to refactor (**45 pts**).

Exercise 2 - A Checkers Game – 20%-Time (30 pts)

1. Google (used to?) ask their employees to spend 20% of their time at Google on a project that their job description does not cover. As a result of the 20% Project, Google now has services such as Gmail and AdSense.

   This is your occasion to have similar freedom. You can decide what to do next to your game:[5] It can be an extension/improvement from any perspective, such as improved code quality or novel features.

   Define your own requirements and get them approved by your tutor (especially in terms of load). Afterwards you must implement the requirements. (**22 pts**).

2. During the analysis and design phases of this extension use responsibility driven design and UML (push to the repository the *single* PDF file including all the produced documents) (**8 pts**).

Exercise 3 - A Checkers Game – A Design Pattern IN! (15 pts)

Choose one design pattern among those that we cover in class until the lecture on Nov 01 (included), excluding the *Singleton* pattern. For the chosen design pattern, you must have a corresponding implementation in your code. If not, refactor your code to include it. Then, complete the following points:

1. Write a natural language description of *why* and how the pattern is implemented in your code (**5 pts**).
2. Make a sequence diagram of how the pattern works dynamically in your code (**5 pts**).
3. Make a class diagram of how the pattern is structured statically in your code (**5 pts**).

[3] not only *declared* but also *used* by other objects
[4] If you have less than nine of these *getters/setters* or *public variables*, good job! You have less work to do!
[5] Consider that this exercise is worth one-third of the points of this assignment, so plan its load accordingly.