PV260 Software Quality

Assignment 2 - Code Refactoring

Spring 2015

1 General Information

1.1 Dates

- Assignment start: 17.3.2015
- Dry run submission possible till: 24.3.2015 23:59
- Assignment deadline: 31.3.2015 23:59

1.2 Submission

Please submit your solution as a .jar containing only source files to the Homework vault (Odevzdávarna) called *Assignment 2: Refactoring (Groups 01, 02).* The name of the archive should be as follows: *lastname1-lastname2-assignment2.jar.* The tasks should be solved in the groups of two (exceptionally in groups of three). Include names of the group members in `@author` tag in every class. Only one solution per group should be submitted.

1.3 Evaluation

The maximum points for this is assignment is 15. The points will be distributed based on both fulfilling the functional requirements and compliance of the code with SOLID and Clean Code principles. Additional points can be awarded by completing the optional bonus tasks.

1.4 Project

This assignment works with the following project:

- TRON game: `http://www.mediafire.com/download/ucj53886o8a1164/Tron.zip`

2 Mandatory Tasks

The following tasks are not meant to be completed in any particular order.

2.1 Task 1

- Make it possible to add more players to TRON easily (so 3,4,5... people can play at the same time each with their own uniquely colored bike)
- Addition or removal of players should be as simple as possible, take into account the collision detection, controls configuration, rendering etc.
- Addition of player will be done offline, that is to change number of players the source of the game will be altered and the whole code recompiled. There is no need to create any kind of game menu.

2.2 Task 2

- Adapt current code to make it possible to use as much of it as possible as a generic engine for implementing other games.
- Provide a way for these new implementations to bind with the engine, e.g. interfaces that need to be implemented.
- Make the current TRON game such implementation of the engine.

2.3 Task 3

- Add the possibility to control any player by mouse (e.g. turn left or right using left and right mouse buttons respectively)

2.4 Task 4

- Separate the state of the game from its graphical representation, that is have one class / package represent the Model and other the Presentation.
- The model must contain and provide access for the player bikes, events such as collision between two players happened and so forth.
- There is no dependency from Model to Presentation, only from Presentation to Model
- The model itself should not handle frame timing (Thread.sleep()), that is the responsibility of calling some update method on the model is outside of the model, preferably in the extracted engine code from previous tasks.

3 Bonus Tasks

- As a proof of concept implement second game on top of the game engine you have created (e.g. Pong).
- Create differential tests for the game. That is create a test which will run your code and the original code at the same time and let the user play the game, it then checks every frame that states of both games are completely identical. If any difference occurs the test fails and reports the difference.