SENG 201 Group Project Report

HanByeol Yang(hya62, 14868742),

Byoungsoo Kim(Joshua) (bki42, 68493559)

Introduction

Over the past few weeks working on the SENG 201 Project, There are many decisions to make and think a lot about. This report will explain why the decision has been made and why the result has happened.

Structure of the Application and Design Decision

In the case of User Interface(UI), we implemented Command Line Interface(CLI) as well as a Graphical User Interface (GUI) for GUI design. Moreover, since there is a decision that there is a possibility of using both UIs, the interface class called UserInterface is created and implemented to both GUI and CLI. In the case of GUI by creating a class Gui, There are no more necessary to call GameEnvironment at each GUI concrete class. However, since the class Gui implements the interface UserInterface and other subclasses inherit the class Gui, but this class does not need to specify the setUp method which is declared at the interface, the setUp method at Gui needs to be abstract thus Gui class became an abstract class.

Products such as Ahtletes and Items are randomly generated at the Class Market. Market call generateAthlete methods at enumeration class Athletes. This method will choose one keyword that is represented the kind of athletes or Items, and the class Market creates the athletes and items. Class Item and class Athlete has subclass Class Item and Athlete implement interface Product and this is because these two class has the same trade process as product.

Class Team represent ‘Club’ in project requirement. The class Team has methods and most of them work for managing Team(Club). When the class Team get athletes or items, it cast them from Product to Ahtlete or Item.

GameManager is represented as Stadium. It controls all sequences of a match with plenty of random calls.

GameEnvironment is a facade class of this application. This class help user to interact with the application using the interface. All requests from the user interface will go through GameEnvrionment.

To conclude the design pattern, the application is used a Facade pattern as a Structural Design Pattern, a Builder Pattern as a Creational Pattern to create Items and Athletes, and an Observer pattern as a behavioural of the player’s Team, Athletes, and Items.

Unit Test Coverage

The result of JUnit Test coverage is as follows.

Item and its subclasses have 100 per cent of coverage. Class Athlete itself has 94% of coverage because one method is not tested and this is because this is for getMethod to get an image of athletes. However, Class Athlete's subclasses got 100 per cent of coverage results such as all different characters. Class Market is tested during testing Class Item and Athlete as athletes and items are from Market and trading will be processed through Class Market. All the methods were tested and showed 100 per cent of coverage. Class Team was tested separately, but also tested while testing Class Athlete and Class Item as three of them are interacting with each other. The test coverage for Class Team is 73.2 per cent as it contains some methods that will be used in matching with opponent such as getting score, getting winning score and getting how many game the player played.
Hence, all the basic classes and their smaller subclasses were tested and the results meets the requirements. The rest of the Classes that were not covered were tested by black-box testing.

Feedback, Retrospective of Project

This project gave us a great chance to learn how to work in a team, communicating with a team member for separated tasks and also designing and constructing a program with Java language. We experienced some poor communication between team members as everybody has different opinions and thinking working style, but we ended up understanding and managed to finish this project.

When we look back on the project, there has not been any big trouble between team members, for example, one person being lazy or absent in the meeting. Even though the achieving speed was slower than we expected, the task was separated well like one person doing command line and the other person GUI, and all of us put many efforts on what we were in charge of.

As one person had more experience with java than the other one, there were some situations that experienced one was spending time teaching. Therefore, for the next project, improving more skills before starting a project will boost up the teamwork. And also, analysing requirements and problems and better communication will improve the quality of the project.

Throughout this project, HanByeol spent around 20 hours per week and Joshua Kim spent around 17 hours per week since the project was released.

Finally, we both agreed that HanByeol contributed 60% because he was taking one less course during this semester and Joshua Kim did the rest, 40%.