Software Architecture Design and Implementation
COSC 2391/2401 Semester 1, 2016
Casino Style Spin the Wheel Game

Assignment 1: Single Threaded Single Player Implementation (25 marks)

This assignment requires you to implement a single threaded game engine and single player user interface for a casino style wheel game that is loosely based on Roulette.

The rules are simple, a player chooses their lucky number on the wheel before a random spin by betting points .. no odds/evens, no red/black just pick a number and win or lose!

For this assignment you are provided with a number of interfaces that you must implement to provide the appropriate behaviour as well as a simple console client which will help you test your game engine implementation independently of your GUI (to help you get started early). This will also ensure all functionality is separated from the GUI since your game code should operate independently using the console based client code.

NOTE: You may extend the provided console client code to facilitate testing but must ensure that the original unaltered code can still execute since we will use our own test client to check your code! i.e. do not change any of the interfaces etc.

AWT/Swing User Interface

You are to develop an AWT/Swing user interface that implements the following basic functionality:

Add one player
Place a bet (for the single player)
Spin the wheel showing intermediate output
Display results including updated player point balances

For the user interface you can make the following **assumptions**.

It is up to you how to design the layout and appearance of your interface and you should focus on clarity and simplicity rather than elaborate design. However you should include at least one each of the following.

A pull down menu
A dialog box
A toolbar
A panel which represents the wheel (this can be as simple as a single text label that is updated for each new number received from the `GameEngineCallback` methods).

Marking emphasis will be on the quality of your code and your ability to implement the required functionality.

All of your GUI code (MVC view(s) and controllers(s)) should be separate from, and use, your `GameEngineImpl` implementation (MVC model) via the `GameEngine` interface.

For assignment 1 you need only provide a single player/single-threaded implementation of the game engine[1]. In assignment 2 you will extend this code to add multi-threading, multiple player windows and networking capability to create a full concurrent and distributed multi player game.

[1] You will still need to respect the threading rules of Swing e.g. see `javax.swing.SwingUtilities.invokeLater(…)`

Do not worry about modelling a real Casino "Roulette" game with its additional rules. The focus here is on the implementation using a simple game approach.

NOTE: The interfaces are designed for multiple players to support assignment 2 and the methods should be implemented accordingly (`Client.java` gives an example) but your GUI need only support a single player.

Implementation Specifications

You must implement the provided `GameEngine`, `Player` and `GameEngineCallback` interfaces, in classes called `GameEngineImpl`, `SimplePlayer` and `GameEngineCallbackImpl`. You must provide the behaviour specified by the javadoc comments in the various interfaces and the generated javadoc html. The imports in `Client.java` show you which packages these classes should be in.

More specifically, you must provide appropriate constructors (these can be determined from `Client.java`) and method implementations (from the three interfaces) in order to ensure that your solution can be complied and tested ***without modifying*** the provided `Client.java` (although you can extend this class to thoroughly test your code and it easier to do this prior to writing the separate GUI code since you will minimise/partition the scope of your testing). A sample output trace is provided to help you write the correct behaviour, you do not need to follow the exact output format.

Other points:

1. You should aim to provide high cohesion and low coupling.
2. You should aim for maximum encapsulation and information hiding.
3. You should comment important sections of your code.
4. You should use assertions to test pre- and post- conditions (especially in `GameEngine.spin()`).

Submission Instructions

- You are free to refer to textbooks and notes, and discuss the design issues (and associated general solutions) with your fellow students on Blackboard; however, the assignment should be your own individual work.
- Where you do make use of other references, please cite them in your work. Note that you will only be assessed on your own work so the use of third party code or packages is not advised.

The source code for this assignment (i.e. complete compiled **Eclipse project**[2]) should be submitted as a .zip file by the due date. You will be provided with specific submission instructions on blackboard before the deadline.

Due at the end of week 6 at 11:59PM Sunday 17th April 2016

Late submissions are handled as per usual RMIT regulations - 10% deduction (2.5 marks) per day. You are only allowed to have 5 late days maximum.

[2] You can develop your system using any IDE but will have to create an Eclipse project using your source code files for submission and lab demo purposes.