COSC1295 Advanced Programming
Computer Science and Information Technology
School of Science, RMIT

Assignment 1 – Semester 1 2017

Due: 8th April 2017

Introduction

This assignment is worth 15% towards your final grade. You are required to implement a basic Java program using Java SE 8.0. This assignment is designed to:

- Practise your knowledge of design classes in Java;
- Practise the implementation of various kinds of classes in Java;
- Practise the use of polymorphism;
- Basic project management/development skills, e.g. using GitHub, collaboration and documentation.

This assignment can be submitted as a group of two (no more than two). We encourage group submission.

Academic Integrity

The submitted assignment must be your own work. No marks will be awarded for any parts which are not created by you. More on http://www.rmit.edu.au/academicintegrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

General Implementation Details

- Although you are not required to use more than one class per task, you are required to modularise classes properly. No method should be longer than 50 lines.
- Your coding style should be consistent with Java coding conventions (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)
- You should include comments to explain your code.
- It is not necessary to use dynamic data structures to store the input data (i.e. it is fine to define a fixed size data structure taking into account the maximum possible amount of input data).
- Your programs will be marked with Java SE 8.0. Make sure you test your programs with this setting before you make the submission.

Problem Description

This assignment involves implementing a system for managing a mini game event – Ozlympic Games.

All participants including athletes and officials have a unique ID. Their personal information is also stored including name, age and the state (of Australia) they represent. Assume gender is NOT recorded and NOT relevant.

There are three sports in the Ozlympic: **swimming**, **cycling** and **running**. There are four types of athletes, **swimmers**, **cyclists**, **sprinters** (who can compete in swimming, cycling and running respectively) and **superAthletes** who can compete in all three games.

Points will be rewarded to the winners of games. In each game, a first place worth 5, a second place attracts 2 points and a third place is 1 point. No points for the rest. Each athlete might have points carried over from the previous games. Each athlete should have a ***compete()*** method which will randomly generate a time between 10 to 20 seconds for running, 100 to 200 seconds for swimming and 500 to 800 seconds for cycling.

Each game has a unique game ID such as "S01", "C02" ..."R05". Each game has one official as the referee. Assume each game can have at most 8 athletes to compete. A game will be cancelled unless there are more than 4 participants. Official is the one that can summarise the score each game.

A user can predict the winner for each game. User's prediction is limited to only one athlete in one game. If the prediction is correct, then a congratulation message will be generated.

Driver Class

The driver class is where user interaction occurs, and it should utilise other classes to manage the games. It should support the following functionalities:

- Select a game to run
  NOTE: you can re-run a game meaning a game does not check whether it has happened before or not. However when a game is run again, the user prediction of that game should be reset.
- Allow the user to predict the winner of that game
- Start the game and award points to top 3 athletes according to their finishing time.
- Display a congratulation message if the user prediction is correct.
- Display the final result of all games including the name of the referee for each game.
- Display the points of athletes.

Important: Software requirement may change. Please pay attention to the course announcement.

Part 1: Design (3 marks)

Design an object design for each of these sections. You should take advantage of object-oriented concepts such as composition, inheritance, method overriding, abstract classes, interfaces wherever appropriate.

Class hierarchies, relationships and components must be conceptualised in a relevant manner, based on the problem description and special conditions listed in this document. Furthermore, you must be able to explain how your program design will perform in certain scenarios and circumstances.

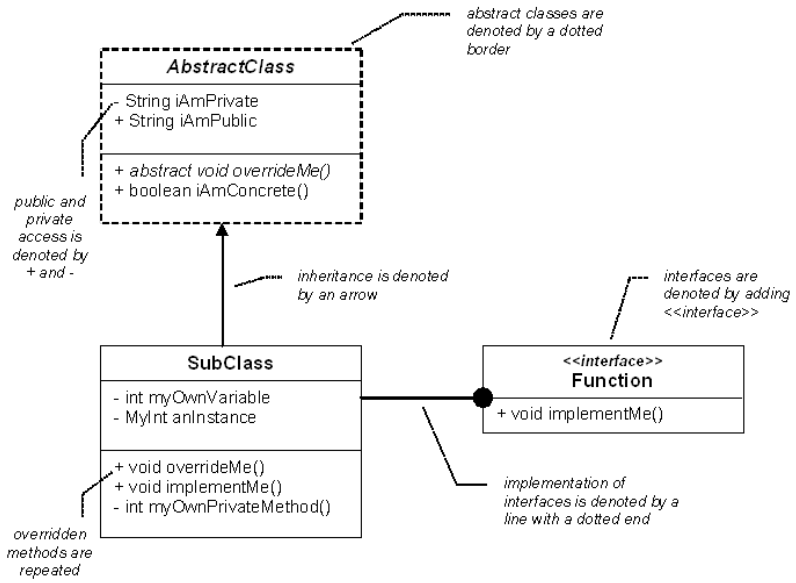
It may be necessary for your design to provide more functionality (for example, accessors and mutators) than that documented here for the mechanics of your design to work properly.

You will need to be able to answer the following questions with respect to your design:

1. Explain how your design will be able to store the information of games, athletes and user predictions.
2. Explain how your class hierarchy will forbid a user from creating a "generic" type of participant (i.e. not a athlete nor an official)
3. Explain the process by which your program will maintain a game and give correct score to athletes according to their performance.
4. Explain how a user prediction can be checked with the actual game results.

Class Diagram Format

The diagram below summarises one way to express common items in a class diagram:



Part 2: Command-line Implementation (12 marks)

This part is an implementation of the design from Part 1 for command-line interaction.

The driver class should display a simple menu system. The menu should be numbered, with the user selecting an option by entering the number that corresponds to the function.

The menu may look like the following:

```
Ozlympic Game
==================================
1. Select a game to run
2. Predict the winner of the game
3. Start the game
4. Display the final results of all games
5. Display the points of all athletes
6. Exit

Enter an option: _
```

User would enter a number from 1 to 6 to select an option. If the input is outside of that range, an error message should be displayed and the menu should be redisplayed. When a valid option is entered, the program should execute the corresponding feature and return to the main menu once the command is completed. Your program should exit if user selects the exit option.

All input data should be read from a hard coded data collection. The implementation of this data collection does not affect the main program. The main program does not know whether the data is from a text file, a database, or a piece of Java code. It is actually unnecessary for the main program to know that.

All output data should be printed to the standard output.

Start-up

Your main start-up class should be named `Ozlympic`. Your code must provide some data, which are imported through a generic interface as mentioned above. The data should include a few different types of games ready to run.

Submission Guidelines

You can choose to complete this assignment in a group of two. There is no extra mark for individual submissions.

For group submissions, we expect about 50:50 split of coding between two group members, for example Member 1 defining all athlete classes, Member 2 writing the command-line interface. One member should write a package, which contains the necessary classes that can be called by the program written by another member. Each class should have ONLY one author. The author's name should be stated in the first few lines of that class definition.

All submissions should have a **Github** repository, which records the progress of the development and so on. In the final submission, each group should nominate the contribution of each member, discuss possible issues and/or comment on your knowledge gain and your learning experience through this group work. Members of a same group may receive different marks.

Submission Details

Submissions should be made via Blackboard as zip file of your project.

In addition you need to make a version (release) named **`assignment1`** in your Github repository. You can have folder for your designs and documents in your GitHub as well.

You can submit your assignment as many times as you want before the due date. The last submission will overwrite the previous ones.

Bonus mark percentage can be added for early submissions based on the mark you can achieve:
Bonus marks are available only for assignments that achieve 50 out of 100.
20% Bonus will be added if you submit 5 days earlier.
10% Bonus will be added if you submit 3 - 4 days earlier.
There is no bonus mark for submissions made one or two days earlier or marked below 50/100.

Submission due date: by 11:59PM Saturday 8th April 2017.

You should archive your design documents and code for Part 2, and submit them as a "zip" file (no RAR or 7-Zip).

Design documents should be in one of the following formats only and should be named as **design** (e.g. design.doc, design.pdf, design.gif, design.jpg or design.png)

- Microsoft Word format (must be compatible with Microsoft Word on RMIT CS terminals)
- Adobe PDF format
- GIF, PNG or JPEG image format (please do NOT submit BMP files)

Please do NOT submit compiled files (*.*class* files), or you will get zero.

You will get zero if the submitted code cannot be compiled.

You may get zero if the submitted code is NOT the same as your version in Github.