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

Assignment 2 – Semester 1 2017

Due: 9am 22th May 2017

Introduction

This assignment is worth 20% 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 exception in Java;
- Practise your knowledge of testing and debugging in Java;
- Practise the implementation of file handling, database handling in Java;
- Practise the implementation of Graphic User Interface in Java;
- 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. You may form a group different to that for Assignment 1. However maintaining the same group is highly recommended.

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 is to continue the development of Assignment 1 – Ozlympic Games.

The scenario of this game is still the same as that in Assignment 1. 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.

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.

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 of each game.

User prediction is not necessary in this assignment.

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

Part 1: Design (2 marks)

Update the class diagram of your design. New classes such as the helper classes for handling exceptions, reading files and so on also need to be added to the diagrams.

You may use a design different to that in your Assignment 1 submission.

You will need to be able to answer the following questions with respect to your design:
1. Explain the changes if you use a different design compared to your assignment 1
2. Explain how the new classes are organized
3. Explain the process by which your program will interact with user and external data source to run a game.

Class Diagram Format is the same as specified in Assignment 1.

Part 2: Exceptions (3 marks)

Possible errors in your program should be handled by the Exception mechanism of Java. That include:

- **`GameFullException`** when trying to add an athlete to a game which already has 8 athletes registered

- **`WrongTypeException`** when trying to add an athlete to a wrong type of game e.g assigning a swimmer to a running game. Note, super athletes can participate in all three types of games. This exception is also for attempts of assigning an athlete as an official or assigning an official to a game.

- **`TooFewAthleteException`** when trying to run a game, which has less than 4 athletes registered.

- **`NoRefereeException`** when trying run a game which has no official appointed.

Other type of exceptions that may occur during file opening, database connection, data management need to be handled properly.

Part 3: External Data Sources (4 marks)

Your program should be able to handle external data sources.

It can read an ASCII file "`participants.txt`" which looks like this

```
Oz1123, officer, Derek, 21, WA
Oz3434, sprinter, Mary, 35, VIC
Oz0091, super, Hannah, 24, NSW
Oz1234, swimmer, Beck, 30, TAS
```

You can assume that
1. the state information is always present and correct.
2. possible states are `ACT`, `NSW`, `NT`, `QLD`, `SA`, `TAS`, `VIC` and `WA`.
3. the type information is always correct if present
4. possible types are `officer`, `sprinter`, `swimmer`, `cyclist` and `super` (for super athletes)
5. no one has more than two types or registered with two states
6. age is an integer and is always correct.

Your program should only read in valid data entries

```
Oz1123, officer, Derek, 21, WA
, sprinter, Mary, 35, VIC                  // ignored, missing ID
Oz1123, officer, Derek, 21, WA             // ignored, duplicates
Oz1234, , Beck, 30, TAS                     // ignored, missing type
```

Your program also writes to an ASCII file "`gameResults.txt`" which looks like the following:

```
C01, Oz1123, 2017-04-06 09:20:19.77
Oz3321, 675, 5
Oz3403, 703, 2
Oz1206, 733, 1
Oz3241, 751, 0
Oz2109, 753, 0
Oz0917, 779, 0

S02, Oz1123, 2017-04-07 11:09:46.31
Oz0812, 103, 5
Oz1436, 123, 2
Oz3267, 134, 1
Oz5132, 145, 0

C03, Oz1123, 2017-04-08 21:07:34.14
Oz1206, 583, 5
Oz3403, 599, 2
Oz0917, 625, 1
Oz3241, 658, 0
Oz3321, 661, 0
Oz2109, 703, 0

R04, Oz1125, 2017-04-08 22:41:01.52
Oz0302, 10.6, 5
Oz1436, 11.5, 2
Oz1164, 12.7, 1
Oz2313, 13.4, 0
Oz1321, 14.9, 0
Oz0029, 15.2, 0

R05, Oz1125, 2017-04-08 22:43:15.27
Oz1321, 10.9, 5
Oz2313, 11.4, 2
Oz0302, 11.6, 1
Oz1164, 12.1, 0
Oz0029, 13.2, 0
Oz1436, 14.5, 0
```

The above file stores the results of previous games, including game ID, official ID and the time stamp. When a new game is finished, its results will be appended at the end of the file. The game ID will automatically increase by one as C03, R04, S05, S06…

When an **embedded** SQLite or HSQLDB database connection is found, then your program should read the participants data from the database. The game results need to be stored in the database as well. The database has one table for the result, which stores 1 - Game ID, 2 - Official ID, 3 - Athlete ID, 4 – Result, 5- Score. Note this database schema is not ideal, but it is fine for this assignment.

Your program should show an error message on the user interface if it cannot find a database connection nor "`participants.txt`" on the local file system.

Part 4: GUI (8 marks)

Your program should have a graphical interface to facilitate user interaction. It should support the following functionalities through **mouse operations**:

- Create a new game either running, cycling or swimming and select a group of athletes for that game;
- Assign an official to that game and run the game;
- One game can only run once. The result will be stored in "`gameResults.txt`" or the database.
- A new game may have the same athletes and official as a previous game. However the game ID will be different.
- Display the results, e.g. time + the point for each athlete, after the game completes;
- Display all finished games including the name of the referee and three winners for each game;
- Display the points of all athletes.
- Real-time game simulation. The real time position of each athlete in a game will be animated as the game progresses, for example as a bar, or a moving icon. Note this is a challenge task. You should only attempt this requirement after successful completion of other parts.
- Errors and exceptions need to be handled gracefully through the GUI as well, e.g. running a game without an official, assigning a swimmer to a cycling game, no database and file.

You are free to design your own GUI as long as it supports the above functionality. The GUI should be user friendly and visually pleasant. You may choose different Java GUI packages. Using JSP and servlets is permitted but does not attract extra marks.

Your code must be runnable on RMIT lab machines and core teaching machines without installing additional packages.

Part 5: Proper Coding Practice (3 marks)

Your source code needs to following the Java coding convention
`http://www.oracle.com/technetwork/java/codeconventions-150003.pdf`

Your program should enable proper tests e.g. with unit test embedded.

Your program is properly documented and can generate a reasonable HTML Javadoc.

Proper messages are shown on the command-line console to reflect frontend GUI operations.

Start-up

Your main start-up class should be named `Ozlympic`. Once it is invoked, a graphical interface should appear to allow user to operate as described above.

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 handling GUI and class diagram, Member 2 handling the rest. 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 **`assignment2`** 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 9am Monday 22th May 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.