Beijing-Dublin International College

SEMESTER 1 FINAL EXAMINATION - (2018/2019)

School of Computer Science

COMP2005J Object Oriented Programming Programming Exam

Prof. Pádraig Cunningham
Dr. Seán Russell*

Time Allowed: 180 minutes

Instructions for Candidates:

Answer all questions.

BJUT Student ID:_____ **UCD Student ID:**_____

I have read and clearly understand the Examination Rules of both Beijing University of Technology and University College Dublin. I am aware of the Punishment for Violating the Rules of Beijing University of Technology and/or University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I accept the punishment thereof.

Honesty Pledge:_____ **(Signature)**

Instructions for Invigilators

Students are permitted to bring any printed notes and textbooks into the examination.

Information

- The name of your project in eclipse should be your UCD student number
- All questions are based on the same program. Each question adds some new functionality to the code. You should not move to the next question until you have finished the one before it.
- You should only submit working code. If your code does not compile or work correctly you should comment it out. You will **fail** if you submit code that does not compile.
- To improve your grade you should submit well designed code with light comments and correct indentation.
- The main method of the program should be in a class named `Main`.
- There are bonus marks for providing good JUnit test cases for the classes you have written. This is not required to get 100%, but it will improve your score.
- There are files with some input data for the program on moodle. They are in a zip file called InputFiles.zip. Download these at the beginning of the exam. Internet access will be turned off after 10 minutes. If you have not downloaded the input files you will have to create them yourself. These files should be put in the project folder in eclipse, not in the src folder with your code. The password for the zip file is `"rugby"` (without the quotes). There is also a java file `League.java`, this contains code that you will need to use in completing the exam and should be copied to your src folder.
- The classes you define should have good encapsulation. If you need to access an instance variable you should add a getter or setter method.

Rules

- You are not allowed bring any device (phone, laptop or tablet) into the exam with you. Even having them is against the exam rules and will result in you failing the module.
- You should only open the programs listed below on your computer, any other programs open will be considered **CHEATING**.
  - Eclipse
  - The Java API in a browser
  - Moodle to download the input files of to submit your work at the end
- Any student caught trying to read the code of another student will also be considered **CHEATING** and will fail the exam

Question Topic

The questions in this exam will all be based around the management of a rugby league. In the questions you will be required to create classes to represent the teams, games and overall league.

Question 1: Score, Team and Game

In the game of rugby there are 4 ways to score, each giving a team a certain number of points. Teams can score a **try** (worth 5 points), a **conversion** (worth 2 points), a **penalty** or a **drop goal** (both worth 3 points).

For each team in the league, we will remember their name as well as the city that they play in. In addition, the team will remember all of the games that they have played in.

Games take place between two teams at one of the teams home city. Teams are usually called *home* and *away*. As they types and number of scores for each team effect the number of points that they earn from a game, these must be recorded for both teams.

Teams are awarded points in their league for playing in games. When a team wins a game they are awarded 4 points, when both teams have the same score, they are both awarded 2 points and when a team loses they are awarded 0 points. There are additional points awarded to teams based on how they have played, any team can get an extra point if they have scored 4 trys. This is called a try bonus point. The losing team may be awarded a point if their score is within 7 of the winning team, this is called a losing bonus point.

a. Implement an enumerated type called `Score` in Java to represent the types of scores. There is no need for any variables or methods in the enumerated type.

(5%)

b. Implement a `Team` class in Java to represent a single team. The class should have a constructor (accepting the name and city as parameters in that order) and getter methods for name and city.

(5%)

c. Implement a `Game` class in Java to represent a single game. The class should have a constructor accepting the names of home and away teams and the location as parameters (All Strings, in that order) and getter methods for all values.

(5%)

d. Implement the following methods in the `Game` class:

- `addScore`. This method should take a String (Team name) and a score object as parameters (in that order). The method should check if the score was for the home team or the away team and remember the required details.
- `getScore`. This method should take pass a team name as a parameter and return an integer representing the sum of the values of each type of score for that team.
- `getPoints`. This method should take a team name as a parameter (String) and return an integer representing the number of points that are awarded to that team from this game (including try bonus points and lowing bonus points).

- `isTryBonus`. This method should take a team name as a parameter (String) and return a boolean value, `true` if the team earned a try bonus point and `false` if they did not.
- `isLosingBonus`. This method should take a team name as a parameter (String) and return a boolean value, `true` if the team earned a losing bonus point and `false` if they did not.

Each of these methods should take no parameters and return an int value.
Note these methods can be completed by calculating the relevant values as each score is added, or by remembering all of the scores and calculating the answer when requested.

(15%)

e. Write a **static/class** method called `readTeamData` in the class `Main`. This method should take two parameters, a data structure to put the teams in and a string containing the name of the file to load. Each line contains the name of a team (all only one word) followed by the name of the city that they play in (this may be multiple words).

Each line of the file should be read, and a team object created to represent each team. These should then be placed in the data structure.

(10%)

f. Implement the following methods in the `Team` class:

- `addGame` - This methods should take a game object as a parameter and store it within a data structure in the object.
- `getNumGames` - This should return the number of games the team has played in
- `getNumWins` - This should return the number of games the team has won
- `getNumLosses` - This should return the number of games the team has lost
- `getNumDraws` - This should return the number of games where both teams had the same score
- `getNumTryBonus` - This should return the number of games where this team has scored 4 trys or more
- `getNumLosingBonus` - This should return the number of games where this team lost by 7 points or less
- `getNumPoints` - This should return the total number of points awarded to this team in all of it's games in the league

All methods beginning with get should take no parameters and return an int value.
Note these methods can be completed by calculating the relevant values as each game is added, or by remembering all of the games and calculating the answer when requested.

(20%)

g. Write a **static/class** method called `readGameData` in the class `Main`. This method should take three parameters, a data structure containing the teams (previously loaded), a data structure to put the games in and a string containing the name of the file to load.

Each game is represented by two lines of data in the file, the first line contains the details for the home team and the second line contains the details for the away team. Each

line starts with the name of the team, and is then followed by the type and number of scores they got in the game (each score represented as a single upper case character). E.g. `"Blues P T T C T C T C P"`, this shows that the blues scored 4 tries, 3 conversions and 2 penalties. There may be any number of scores (including none) and there will be 4 different types of characters present `"T"` meaning try, `"C"` meaning conversion, `"P"` meaning penalty and `"D"` meaning drop goal.

Each line of the file should be read, and a game object created to represent each game that took place and added to both teams. All scores should be added to the each game. These should then be placed in the data structure.

(10%)

(Question Total 70%)

Question 2: League

The `League` class has been provided for you, however the author was a little lazy so there is a small bit of code missing.

a. In the main method of the `Main` class, create 2 league objects to represent different leagues. The first league is called the `"Pro14"`. The teams are stored in the file `"pteams.txt"` and games in the files `"p1.txt"` to `"p17.txt"`. These files should be loaded and the teams and games added to the league. The second league is called the `"Six Nations Championship"`. The teams are stored in the file `"steams.txt"` and games in the files `"s1.txt"` to `"s4.txt"`. These files should be loaded and the teams and games added to the league. For both of these leagues you should call the `printLeagueTable` method to output the current standings in the league and the `printGames` method to output the results of all of the games.

(10%)

b. The creator of the `League` class was lazy and forgot to implement the part of the code that sorts the teams into the correct order (highest points at the top) before they were output. Add this functionality to the `sortLeagueTable` method so that the league table is printed in descending order of points.
Note: you should make no other changes to the `League` class.

(20%)

(Question Total 30%)

Submission

Create a single **zip** file containing your entire project.

How to Submit

Your entire project should be submitted as a zip file. To do this, please follow these instructions:

a. Right-click on your project and choose Export

b. In the menu that appears choose `General -> Archive File` and click next

c. Make sure that only your project is selected with a tick (this means all files)

d. Click on browse to choose the location of the file will be saved and its name

e. Choose the name of the zip file as your UCD student number

f. Click Finish

g. After you have completed this open the zip file to make sure that all of your code is in it. There will be no second chances to submit your work.

Output

The output of the code should look something like this:

```
Table for the Six Nations Championship
Team            P    W    L    D    TB   LB   PTS
Wales           4    4    0    0    0    0    16
England         4    3    1    0    3    0    15
Ireland         4    3    1    0    2    0    14
Scotland        4    1    3    0    1    1    6
France          4    1    3    0    1    1    6
Italy           4    0    4    0    0    0    0
Record of games played in the Six Nations Championship
Home       Away       Location        H     A
France     Wales      Paris           19    24
Scotland   Italy      Edinburgh       33    20
...
England    Italy      London          57    14
Ireland    France     Dublin          26    14
Table for the Pro14
Team            P    W    L    D    TB   LB   PTS
Leinster        17   15   2    0    11   1    72
Glasgow         17   12   5    0    11   2    61
Munster         17   12   5    0    8    2    58
Benetton        17   10   6    1    5    3    50
Ulster          17   10   5    2    4    1    49
Connacht        17   9    8    0    5    6    47
Blues           17   9    8    0    6    5    47
Scarlets        17   9    8    0    5    4    45
Edinburgh       17   8    9    0    5    5    42
Ospreys         17   8    9    0    3    4    39
Cheetahs        17   6    10   1    7    3    36
Kings           17   2    15   0    5    7    20
Dragons         17   4    13   0    0    3    19
Zebre           17   3    14   0    5    1    18
Record of games played in the Pro14
Home       Away       Location        H     A
Blues      Leinster   Cardiff         32    33
Ospreys    Edinburgh  Swansea         17    13
...
Blues      Kings      Cardiff         26    19
Dragons    Ulster     Newport         15    28
```