Beijing-Dublin International College

SEMESTER 1 RESIT EXAMINATION - (2017/2018)

School of Computer Science

COMP2005J Object Oriented Programming
Programming Exam

Prof. Pádraig Cunningham
Dr. Seán Russell*

Time Allowed: 120 minutes

Instructions for Candidates:

Answer all questions.

BJUT Student ID:________________ **UCD Student ID:**________________

Honesty Pledge:________________________________ **(Signature)**

Instructions for Invigilators

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

Project and Submission Instructions

- Create a project in eclipse where the project name is your student number.
- All code should be in the **default** package.
- There should be **only one** main method and it should be in a class named `Main`.
- There are bonus marks for providing good JUnit test cases for the classes you have written.
- 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 `"goodluck"` (without the quotes).
- The classes you define should have good encapsulation. If you need to access an instance variable you should add a getter or setter method.
- You should create the zip file for submission by following these instructions
  - Right-click on your project and choose **Export**
  - In the menu that appears choose **General** → **Archive File** and click **next**
  - Make sure that only your project is selected with a tick (this means all files)
  - Click on browse to choose the location of the file will be saved and its name
  - Choose the name of the zip file as your UCD student number
  - Click **Finish**

The questions in this exam will all be based around formula 1 racing. All questions will be associated with this concept.

Question 1: Drivers

Drivers race each other to win points in each race and at the end of the season the driver with the most points wins the driver championship. Each driver has a unique number that is painted on their car. A shortened version of their name is shown on the screen when the full name will not fit. Drivers usually use their full name (first name and family name) but are often referred to by what country they are from.

a. Implement a `Driver` class in Java to represent a single driver. The class should have a constructor and any required getter methods.

(10%)

b. In addition to tracking the number of points that a driver has received, we also track the number of laps of the races that they have completed. The number of laps completed and points won need to be increased as the season progresses based on the results of the races. Implement `addPoints` and `addLaps` in the class. Both methods should take a single integer as a parameter and return no values.

(10%)

c. Override the `toString` method in the `Driver` class, when called this method should return a String containing the drivers number, followed by their shortened name and finally followed by their full name, the code of the country they are from, the number of points they have and the number of laps they have raced. For example it should look like this `"44 HAM Lewis Hamilton from GBR has 0 points and has raced 0 laps"`.

(10%)

d. Write a **static/class** method called `readDriverData` in the class `Main`. This method takes no parameters and should read all of the driver data from the file `"d.txt"`. Each line of the file represents a single driver and the data is in the following format `<firstName> <familyName> <shortName> <number> <country>`.

After reading the data from the file a Driver object should be created for each line and added to a suitable data structure. The driver objects will be needed in the next questions so the data structure should be returned when the file is fully read.

The `readDriverData` method should be called in the main method and the returned data structure remembered in a variable. Each driver object should be printed to the screen in the main method.

The output of the code should look like this, but the order may depend on the data structure that you use:

```
2 VAN Stoffel Vandorne from BEL has 0 points and has raced 0 laps
3 RIC Daniel Ricciardo from AUS has 0 points and has raced 0 laps
5 VET Sebastian Vettel from GER has 0 points and has raced 0 laps
7 RAI Kimi Raikkonen from FIN has 0 points and has raced 0 laps
...
40 DIR Paul DiResta from GBR has 0 points and has raced 0 laps
44 HAM Lewis Hamilton from GBR has 0 points and has raced 0 laps
55 SAI Carlos Sainz from ESP has 0 points and has raced 0 laps
```

(15%)

(Question Total 45%)

Question 2: Teams

Drivers are grouped together into teams. Each team has a name and can have many drivers but usually only have two. The team with the drivers that received the most points (added together) at the end of the season wins the team championship.

a. Implement a `Team` class in Java to represent a single team. The class should have a constructor and any required getter methods. There should be a method called `addDriver`, that allows a new driver to be added to the team.

(10%)

b. Implement a method `getPoints` in the `Team` class. This method should calculate the number of points the team has based on the points that it's drivers have.

(5%)

c. Override the `toString` method for the `Team` class, when called this method should return a String containing the Teams name, followed by the the details of each driver in the team on a separate line. For example it should look like this:

```
Ferrari
```

(5%)

d. Write a **static/class** method called `readTeamData` in the class `Main`.This method takes no parameters and should read all of the team data from the file `"t.txt"`. Each line of the file represents a single team and the data is in the format `<teamName> <drivernumbers>`.

The team name will be between 1 and 3 space separated words and there will be either 2 or 3 integers representing driver numbers after this. After reading the data from the file a Team object should be created for each line and added to a suitable data structure. The correct drivers should be added to each team and the team objects will be needed in the next questions so the data structure should be returned when the file is fully read.

The `readTeamData` method should be called in the main method and the returned data structure remembered in a variable. Each team object should be printed to the screen in the main method.

The output of the code should look like this:

(10%)

(Question Total 30%)

Question 3: Championships

Drivers compete in a championship, the driver with the most points winning.

a. Implement a class in Java called `Championship` to represent the driver championship. This class should have data structures for remembering drivers. The data structures should be constructed in the constructor of the class.

(10%)

b. Add the method `addDriver` to the `Championship` class. This method should take a single parameter of a Driver object and add that to the data structure to be remembered. The method does not need to return anything.

(10%)

c. Override the `toString` method in the `Championship` class. This method should print the current championship table for the drivers.

Note: to do this correctly, the drivers must be sorted based on the number of points that they have, remember there are methods in `Collections` to do this.

The output should look like this:

```
Current Driver Championship Table
Position  Number  Name  Country  Points
1         5       VET   GER      25
...
25        40      DIR   GBR      0
```

(15%)

d. Add code to the main method to create a championship object and add all of the drivers and teams to the championship.

(5%)

e. Add a method called `readRaceData` to the `Championship` class. This method should take a single parameter, a string containing the name of the file to be read. This method should read the results of a race, each result will be on a single line in the following format `<position> <driverNumber> <numberOfLaps> <points>`.

The position is either an integer of a String (if the driver did not finish) and all others are integers. After reading each line, the method should find the correct driver, and then add the points they have won and the laps they have completed. The method should not print anything or return any values.

Add code to the main method in the `Main` class to perform the following steps.

- Load the race data from each of the files `1.txt` to `20.txt`.
- After reading the data from each file the updated championship tables should be printed to the screen.

The output should look something like this:

```
...
Results after race 19
Current Driver Championship Table
```

```
Position  Number  Name  Country  Points
1         44      HAM   GBR      345
2         5       VET   GER      302
...
Results after race 20
Current Driver Championship Table
Position  Number  Name  Country  Points
1         44      HAM   GBR      363
2         5       VET   GER      317
...
```

(15%)

(Question Total 55%)