University of Westminster
School of Computer Science & Engineering

5COSC019W Coursework (Semester 1)

	Dr D. Dracopoulos
Weighting:	50%
Description	Coursework
Learning Outcomes Covered in this Assignment:	- LO1 Identify and justify good practices in object-oriented software development. Communicate the aspects of object-oriented programming which are advantageous when compared to non-object-oriented paradigms; - LO2 Acquire detailed knowledge of concepts of object-oriented programming and apply characteristics, tools and environments to adapt to new computational environments and programming languages which are based on object-oriented principles; - LO3 Design and implement applications based on an object-oriented programming language, given a set of functional requirements. Use APIs which have not been exposed to previously, in order to develop an application requiring specialised functionality; - LO4 Design and Implement graphical interfaces using an object-oriented programming language; -LO5 Apply appropriate techniques for evaluation and testing and adapt the performance accordingly.
Handed Out:	26/10/2021
Due Date	14/12/2021 13:00
Expected deliverables	Java Source code and any Resources (images, etc)
Method of Submission:	Electronic submission on BB via a provided link close to the submission time. The file you upload should have the following naming format: wNNNNNNNN.zip (where wNNNNNNN is your university ID login name)
Type of Feedback and Due Date:	Individual feedback via Blackboard within 2 weeks of submission **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website :http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

5COSC019W OBJECT ORIENTED PROGRAMMING - Assignment
Deadline 14/12/2021, 13:00

Dr Dimitris C. Dracopoulos
Email: d.dracopoulos@westminster.ac.uk

Description

Your task is to create a Java program which simulates the manipulation of a Formula 1 racing car championship.

For the GUI part you are NOT allowed to use drag and drop tools (such as those found in Netbeans, etc.) to create the graphical user interface for any part of this coursework! All graphical code should be manually written in Java Swing and no tool which generates code automatically should be used. Any submission which uses for any part of this assignment drag and drop tools will not receive any marks for these parts.

Implement a class `Formula1ChampionshipManager` which extends interface `ChampionshipManager`. The `Formula1ChampionshipManager` class maintains a number of drivers and cars (constructors, i.e. manufacturer of the car) which compete in the Formula 1 car championship. Each driver belongs to exactly one constructor team (e.g. Ferrari) and each constructor team has a single driver (e.g. Mercedes has a single driver called Hamilton)

The details for the implementation of the system are given in the steps below: **It is important to follow exactly the specifications and your implementation must conform to these:**

1. Design and implement classes `Driver` (abstract class), `Formula1Driver`. The classes should include appropriate methods and hold information about the name of the driver, their location, the team they belong to and various statistics about the drivers.
   `Formula1Driver` should include statistics such as how many first position, second positions and third positions an instance of it has achieved in the season. The number of points that a driver currently has, and the number of races participated so far in the season should also be included.

   The points awarded for each driver in a race (and for the all the calculations in this assignment) are according to the following scheme:

   ```
   1:25 2:18 3:15 4:12 5:10 6:8 7:6 8:4 9:2 10:1
   ```

   i.e. the driver who got the first position in the race wins 25 points, the second 18, the third 15 and so on. A driver must finish the race to be awarded points and also finish it in the first 10 positions. *(5 marks)*.

2. Implement a class `Formula1ChampionshipManager` which extends interface `ChampionshipManager`. The `Formula1ChampionshipManager` class maintains a number of drivers who play in the Formula 1 championship. *(5 marks).*

   The class should create a menu based on text input (i.e. console and NOT graphical components) and give the user the choice of:

   - Create a new driver (who is added in the championship). The driver should be associated with a unique team (car manufacturer) *(4 marks).*
   - Delete a driver and the team that the driver belongs to from the Formula 1 championship. *(2 marks).*
   - Change the driver for an existing constructor team (e.g. change the driver for the Ferrari team). *2 marks*
   - Display the various statistics for a selected existing driver. *(4 marks).*
   - Display the Formula 1 Driver Table, i.e. display all the drivers competing in the Formula 1 championship, the team they belong to and some of their statistics, in descending order, according to the points they have in the current season. Thus, the driver who has the maximum number of points should be displayed first, the driver being second in the championship should be displayed next, etc. In the case that two drivers have the same number of points the driver who has won the most first positions in races should appear first. *(8 marks).*
   - Add a race completed with its date and the positions that all the drivers achieved. The statistics of all the drivers who participated and the Formula 1 championship table are updated automatically. *(7 marks).*
   - Saving in a file of all the information entered by the user so far. *(8 marks).*
   - The next time the application starts it should read all the information saved in the previous file (resume/recover the previous state of the program) and continue its operation based on that with the user being able to enter new information or change the existing information. *(9 marks).*

3. Start a graphical user interface (GUI) based on Java Swing from the text menu (i.e. console) which is able to do the following:

   - Display the list (table) of all the drivers and their statistics in descending order of points. *(4 marks).*
   - Give the user the possibility of sorting the previous table according to points won by drivers (ascending order). *(4 marks).*
   - Give the user the possibility of sorting the previous table according to the largest number of first position won in races (descending order). *(4 marks).*
   - Add a button which every time it is pressed it generates one random race with random positions achieved by the existing drivers. This automatically updates the Formula 1 championship table by adding the race (points, positions and other statistics). The positions should be entirely random and not hardcoded in your source code. The button should generate a different race with different driver positions every time it is clicked. The user should be able to see the randomly generated race with the driver positions (in addition to the table of standings), in order to be able to verify the correctness of your code for the updated information of the table. *(8 marks).*

- Add a button which is similar in functionality with the previous questions (i.e. it generates the results of a full race and adds them in the statistics) with the following modification. The drivers are starting the race in a randomly calculated position (e.g. Vettel in starting position 1, Hamilton in starting position 2 and so on). These starting positions should be random and not hardcoded. The results of the race are probabilistic and according to the starting position, more specifically: the driver starting in position 1, has 40% probability to win the race, the driver starting in position 2 has 30% probability to win the race, the drivers starting in positions 3 and 4 have a 10% each to win the race. All of the drivers in positions 5 to 9 each have a probability of 2% to win the race and all other drivers have a 0% chance to win the race. The rest of the positions (2–10) are determined completely randomly *marks (8)*.
- Add a button which displays all the completed races sorted in ascending order of date played (both randomly generated or manually entered using the text menu functionality described above). This should display all the races that took place in the season, included races inserted and generated in previous runs of the application (assuming that the user saved the information entered using the text menu functionality above). *(7 marks)*.
- Add a button and a textbox which can be used to search for all races that a given driver participated. The full details of the races should be displayed (i.e. both the dates and the positions of the driver in the matching races). *(5 marks)*.

Marking Scheme: The marks achieved for each part of the program are indicated in the description of the task above. In addition to these the following will be taken into account:

- *Code readability* (structure, comments, variable naming, etc.): 3%
- *Implementation* (e.g. quality, efficiency, etc.): 3%

The maximum for work which does not compile is 30% (i.e. a mark in the range 1–30% will be awarded).

Submission of assignments using a different method other than Blackboard will not be accepted and zero (0) marks will be awarded in such cases.

Deadline: Tuesday 14th of December 2021, 13:00.

Submission Instructions

Files to submit: all your source code files (the files with extension `.java` files NOT the `.class` files). This should include test classes, i.e. classes which were implemented for the sole purpose of testing other classes. **You should NOT copy and paste your code into notepad, Word or other applications but simply submit in a zip file all your Java source, i.e. the files with extension** `.java`.

Alternatively you can use Netbeans to create a zip file of your whole project (go to the menu `File->Export Project->to Zip`)

Referencing code: Any code taken from other resources (i.e. a textbook or internet) should be referenced in comments within your code (full textbook details or full web URL), identifying

the exact code that you used it as part of your application and the exact portions of the original source code that you reused.

You should submit via BlackBoard's Assignment functionality (do NOT use email, as email submissions will be ignored.), all the files described above. A single zip file with the name `wNNNNNNNN` (where `wNNNNNNN` is your university ID login name) containing all the above files could be submitted alternatively. You can create such a file by using the main menu in Netbeans and choose `File->Export Project->to Zip` or use other tools in your operating system.

Note that Blackboard will allow to make a submission multiple times. Make sure before submitting (i.e. before pressing the `Submit` button), that all the files you want to submit are contained there (or in the zip file you submit).

In the case of more than one submissions, only your last submission before the deadline given to you will be marked, so make sure that all the files are included in the last submission attempt and the last attempt is before the coursework deadline.

Request to mark submissions which are earlier than the last submission before the given deadline will be ignored as it is your responsibility to make sure everything is included in your last submission.

The following describes how to submit your work via BlackBoard:

1. Access `https://learning.westminster.ac.uk` and login using your username and password (if either of those is not known to you, contact the Service Desk, tel: +44 (0) 207 915 5488 or log a call via `https://servicedesk.westminster.ac.uk`.).
2. Click on the module's name, `MODULE: 5COSC019W.2021 OBJECT ORIENTED PROGRAMMING` found under `My Modules & Courses`.
3. Click on the `Assessment->Submit Coursework->Coursework`.
4. Click on `View Assignment`.
5. Attach your zip file containing all the Java source code files, by using the `Browse` button.
6. Create a Word or PDF file with the following information:
   - *Comments*: Type your full name and your registration number, followed by:
     "I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."
7. Attach the file with the statement above.
8. Check that you have attached both the zip and the statement file.
9. Click the `Submit` button.

If Blackboard is unavailable before the deadline you must email the Registry at `fitzregistry@westminster.ac.uk` with **cc:** to myself and your personal tutor before the deadline with a copy of the assignment, following the naming, title and comments conventions as given above and stating the time that you tried to access Blackboard. You are still expected

to submit your assignment via Blackboard. Please keep checking Blackboard's availability at regular intervals up to and after the deadline for submission. You must submit your coursework through Blackboard as soon as you can after Blackboard becomes available again even if you have also emailed the coursework to the above recipients.