Formula 9131 Grand Prix

Introduction

Assignment 2 has two components: a programming task and a unit test. These components are assessed independently but BOTH are required. The programming component is worth 20% of the marks for your final assessment in this unit and an associated test held during tutorial time is worth 5%. You must attempt both components to be awarded any marks for Assignment 2.

This document specifies the programming component of Assignment 1. This component is due by 5pm Saturday of Week 12 (1st June, 2019). **Heavy penalties will apply for late submission.** This is an individual assessment task and must be your own work. You must attribute the source of any part of your code which you have not written yourself. Please note the section on plagiarism in this document.

The assignment must be done using the BlueJ environment.

The Java source code for this assignment must be implemented according to the **Java Coding Standards for this unit.**

Any points needing clarification may be discussed with your tutor in the lab classes.

Completion of this assignment contributes towards the following FIT9131 learning outcomes:

1. *design, construct, test and document small computer programs using Java;*
2. *interpret and demonstrate software engineering principles of maintainability, readability, and modularisation;*
3. *explain and apply the concepts of the "object-oriented" style of programming.*

Specification

In this assignment you will write a program to simulate the running of a Formula 9131 Grand Prix championship. This is similar to (but not the same as) the Formula I Grand Prix championship. This section specifies the required functionality of the program. **Only a text interface is required for this program**; however, more marks will be gained for a game that is easy to follow with clear information and error messages to the player.

The aim of the *Formula 9131 Grand Prix* is for a group of drivers to compete against each other in a series of car races and win the *Formula 9131 Grand Prix* racing championship

Grand Prix championship

The *Grand Prix* championship simulation begins with a message requesting the championship organiser to enter the number of races in the championship. There can be from 3 to 5 races in a championship. The championship then proceeds with the running of each race.

Before each race the championship organiser selects the venue from a list of possible venues. There are eight possible venues and each venue will be used only once in a championship. A race at each venue will have a fixed number of laps.

The list of drivers entered in the championship are read from a file. At the beginning of a race each driver is allocated to a grid (starting) position. The grid positions are determined from each driver's ranking with the first grid positions allocated to the drivers with the highest rankings. The layout of a starting grid for 10 cars is shown in Figure 1.

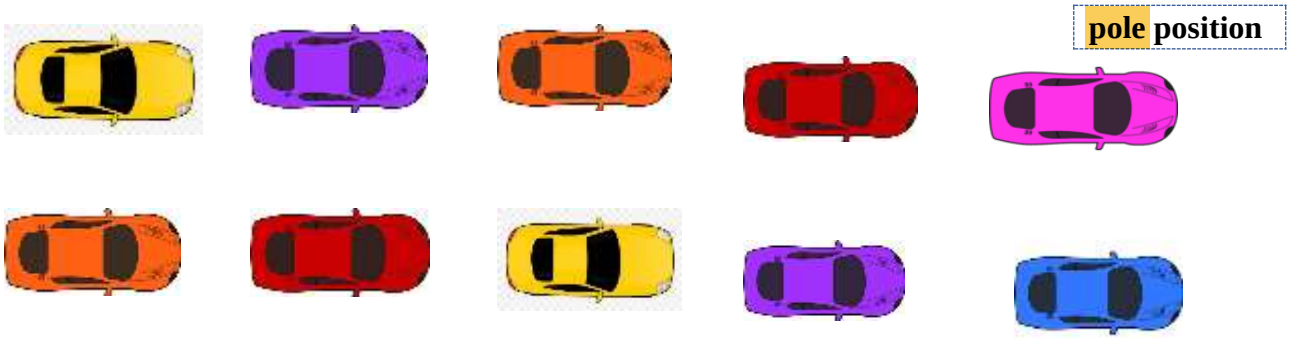


Figure 1 Grid layout for 10 cars

The grid position is simulated by a time penalty. The driver at the pole position (the best position) will have no penalty and the others will be given a time penalty determined by their grid position. These are shown in Table 2. The time penalties are awarded at the start of the race.

Table 1 Rankings, grid positions and time penalties for the race start

Name	Ranking	Grid position	Time penalty (seconds)
Senna	1	1st row (pole position – pink car)	0
Lauder	2	1st row (next to pole – blue car)	3
Hamilton	3	2nd row (behind pole – red car)	5
Prost	4	2nd row (purple car)	7
…remaining drivers	5 or greater	back of the field	10

The race begins and each driver attempts to complete the required number of laps in the fastest time. After each lap each driver's time is calculated and added to their accumulated time for the race. Each driver will be given the average lap time as read from the file for the venue; however, during each lap there are random events that can affect the lap times and whether the driver is eliminated from the race and the championship. These are displayed to the screen as they occur. At the end of each lap the name and time of the leading driver is displayed. The leader will be the driver with the lowest accumulated race time.

After all laps have been completed the drivers with the four lowest total times for the race are awarded points as shown in Table 2. It is possible that two or more drivers will have the same race time. If this happens then the places and subsequent points are randomly allocated. The results of the race are then displayed to the screen.

Table 2 Points for race places

Place	Points
1st	8
2nd	5
3rd	3
4th	1

After all the races have been completed the championship driver is the one with the highest number of accumulated race points. The championship results are then displayed to screen.

The following are the race and championship rules:

1) The venues are read from a file "venues.txt" at the start of a championship. Each venue will have a name, the number of laps at that venue, an average lap (circuit) time in seconds and the chance of rain occurring as a percentage.

2) The average lap times for the venues range from 60 to 90 seconds. Each driver will drive their car at the average lap time for the venue but their lap time may vary due to random events which occur during each lap.

3) The driver details are read from a file "drivers.txt" at the start of a race. Each driver will have a name, ranking, special skill, eligibility to race and accumulated race points. At the beginning of a tournament each driver is eligible to race and has zero accumulated race points.

4) Each of the drivers has one of three special skills: braking, cornering, and overtaking. This means they can sometimes complete a lap in a shorter time.

    a) Braking and cornering skills can be used every lap and each driver with this skill will have a random time reduction of between 1 to 8 seconds.

    b) An overtaking skill can be used every third lap and each driver with this skill will have a random time reduction of between 10 to 20 seconds.

5) During each lap, for each driver there is a chance of one of the following problems occurring.

    a) 5% chance of a car developing a minor mechanical fault resulting in a 20 second addition to the lap time.

    b) 3% chance of a car developing a major mechanical fault resulting in a 120 seconds addition to the lap time.

    c) 1% chance of a car developing an unrecoverable mechanical fault resulting in the driver taking no further part in the race.

HD Level

If all the above functionality is completed then you may attempt the following to achieve an HD level.

At the end of a race the four leading drivers are given rankings 1 to 4 and all other drivers are given a ranking of 5. These ranking are used to determine the starting positions for the next championship. The drivers' details are then saved back to the file.

Additional functionality for bonus marks:

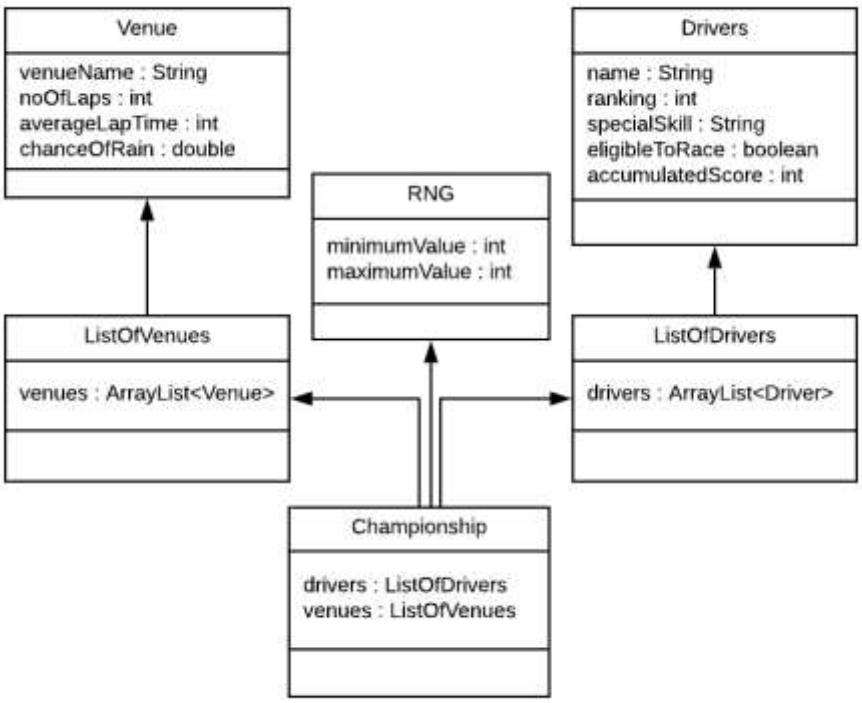
If all the above functionality is completed then you may attempt the following for bonus marks. Note that bonus marks will only be awarded if all the above is complete.

The cars start the race with dry-weather tyres. On the second lap of a race each driver has a 50% chance of changing to wet-weather tyres in case of rain. This adds 10 seconds to the driver's time for that lap.

The chance of rain on any lap will depend on the venue. This will range from 1 to 20%. If rain occurs then the drivers of cars with dry-weather tyres will have 5 seconds added to their lap time.

Program design

Your program must consist of at least the following classes: *Championship, Driver, ListOfDrivers, Venue, ListOfVenues, and RandomNumber.* For more scalable design you may choose to include classes such as *FileIO*, *Input*, and *Validation*. The following is the proposed class diagram for the program. The remainder of this section gives details of these classes. There are suggested fields for each class. If you include any extra fields then you must be able to justify these. You may want to include comments in your program to state any assumption made.



Championship:

The main class of the program. It will allow the user to start the game. It generally handles all the inputs and outputs to the user unless another utility class has been included to do this. It has only two attributes which store objects of the classes which have an arraylist of venues and drivers.

Venue:

This class describes a single venue for the race. The attributes describe the name, the number of laps needed at the circuit, the average lap time per lap, and the probability of rain expressed as a percentage.

ListOfVenues:

This class stores an arraylist of Venue objects where races can be help.

Driver:

This class describes a single driver in the championship. The attributes describe the name, the ranking, a special skill the driver has, the driver's eligibility to race, and the accumulated time. The driver's eligibility to race only changes when they are no longer able to race due to a major malfunction or a crash. Once changed, they can no longer compete in the current race but they can compete in further races in the championship.

List of Drivers:

This class stores an arraylist of Driver objects who race in the championship.

Test Strategy

Using the template discussed in Week 7's lecture, your assignment must also provide a detailed test strategy for the Driver class only.

Hints and Suggestions

Your assignment may want to use pass by reference to achieve a well designed program. Additionally, should your program wish to accept only String inputs from the user, you may want to look up the Integer.parseInt() method which belongs to the Integer class. This can be used along with exception handling to ensure your program doesn't crash.

Your program must use a dynamic collection (i.e. ArrayList).

Your program must use exception handling so that it doesn't crash no matter what the user inputs.

Assessment

Assignment 2 has two components: a programming task and a unit test. These components are assessed independently but BOTH are required. Note you must attempt both components to be awarded any marks for Assignment 2.

Programming Task Assessment (20%)

Assessment for this component will be done via an interview with your tutor.

The marks for your program code will be allocated as follows:

- 10% - Test strategy for the Driver class.
- 35% - Object-oriented design quality. This will be assessed on appropriate implementation of classes, fields, constructors, methods and validation of the object's state and adherence to Java coding standards
- 55% - Program functionality in accordance to the requirements.

You must submit your work by the submission deadline on the due date (a late penalty of 20% per day, inclusive of weekends, of the possible marks will apply - up to a maximum of 100%). There will be no extensions - so start working on it early.

Marks will be deducted for untidy/incomplete submissions, and non-conformances to the FIT9131 Java Coding Standards.

Please note that your program code will be submitted to a code similarity checker.

Interview

You will be asked to demonstrate your program at an "interview" following the submission date. At the interview, you will be asked to explain your code/design, modify your code, and discuss your design decisions and alternatives. Marks will not be awarded for any section of code/design/functionality that you cannot explain satisfactorily (the marker may also delete excessive in-code comments before you are asked to explain that code).

In other words, you will be assessed on your understanding of the code, and not on the actual code itself.

Interview times will be arranged in the tutorial labs in Week 12. The interviews will take place in week 14. It is your responsibility to attend the lab and arrange an interview time with your tutor. **Any student who does not attend an interview will receive a mark of 0 for the assignment.**

Unit Test Assessment (5%)

This assessment will be a written assessment and will be held during the tutorial times in Week 12. Keep in mind that students are expected to have finished ALL their code by the submission deadline in Week 12, and more importantly to have done their own work.

The unit test will be a closed book test wherein students will be required to WRITE code to answer a few questions. The questions will cover material from Weeks 1 – 10 of the lectures. Students should bear in mind that BlueJ or any other electronic device will not be permitted, so students need to know HOW to write their own code in order to finish the unit test.

Submission Requirements

The assignment must be uploaded to Moodle by 5pm Saturday of Week 12 (1st June, 2019).

The submission requirements for Assignment 2 are as follows:

A .zip file uploaded to Moodle containing the following components:

- the BlueJ project you created to implement your assignment.
- a completed **Assignment Cover Sheet**. This will be available for download from the unit's Moodle site before the submission deadline. You simply complete the editable sections of the document, save it, and include it in your .zip file for submission.

The .zip file should be named with your Student ID Number. For example, if your id is 12345678, then the file should be named 12345678_A2.zip. Do not name your zip file with any other name.

It is your responsibility to check that your ZIP file contains all the correct files, and is not corrupted, before you submit it. If you tutor cannot open your zip file, or if it does not contain the correct files, you will not be assessed.

Marks will be deducted for any of these requirements that are not complied with.

Warning: there will be no extensions to the due date. **Any late submission will incur the 20% per day penalty.** It is strongly suggested that you submit the assignment well before the deadline, in case there are some unexpected complications on the day (e.g. interruptions to your home internet connection).

Extensions

All requests for extensions must be sent to Judy Sheard (judy.sheard@monash.edu). Requests must follow the faculty guidelines, include the required forms and be submitted as soon as possible. In addition, when emailing for an extension, please remember to include all code completed until that time. This assignment cannot be completed in a few days and requires students to apply what we learn each week as we move closer to the submission date.

Plagiarism

Cheating and plagiarism are viewed as serious offences. In cases where cheating has been confirmed, students have been severely penalised, with penalties ranging from losing all marks for an assignment, to facing disciplinary action at the Faculty level. Monash has several policies in relation to these offences and it is your responsibility to acquaint yourself with these.

Plagiarism (http://www.policy.monash.edu/policy-bank/academic/education/conduct/plagiarism-policy.html)