Assignment 3

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098
October 25, 2016

1. Introduction

The assignment simulates car racing around a race track. I need to demonstrate my ability to use conditional statement, constructor and be able to interact with multiple classes. The output will consist of overall winner for each lap, with the final lap raining.

2. Pseudocode

Class RaceTrack:

```
  Initialise private int averageLapTime
  Initialise private Boolean isRaining

  Define RaceTrack
    Set averageLapTime
    Set Boolean value isRaining

  Define getLapTime
    Return averageLapTime

  Define setRainingStatus
    Set Boolean value to isRaining

  Define determineRaceLeader
   If car1 total time <= car2 and car 3 total time, Then
    Return car1
   Else if car2 total time <= car1 and car 3 total time, Then
    Return car2
   Else
    Return car3
```

Class Car:

```
 Intialise private int id
 Intialise private int fuel
 Intialise private int lowFuelBoost
 Intialise private int highFuelBoost
 Intialise private int fuelConsumptionPerLap
 Intialise private int pitStopTime
 Intialise private int rainSlowDown
 Intialise private int totalTime
 Intialise private RaceTrack raceTrack

 Define Car
  Set id
  Set fuel
  Set lowFuelBoost
  Set highFuelSlowdown
  Set fuelConsumptionPerLap
```

Set pitStopTime
Set rainSlowDown
Set totalTime

Define completeLap
Add LapTime from object raceTrack to totalTime

If fuel < 50, Then
Add highFuelSlowdown to totalTime
Else
Subtract lowFuelBoost from totatlTime

If isRaining from object raceTrack is TRUE, then
Add rainSlowDown to totalTime

Subtract fuelConsuptionPerLap from fuel

If fuel < fuelConsumptionPerLap, Then
Add pitStopTime to totalTime
Set fuel to 100

Define getTotalTime
Return totalTime

Define getId
Return Id

Class RaceSimulator:
Initialise new car1
Initialise new car2
Initialise new car3

car 1 completelap silverstone
car 2 completelap silverstone
car 3 completelap silverstone
Print id of the car, from determineRaceLeader of object Silverstone

car 1 completelap silverstone
car 2 completelap silverstone
car 3 completelap silverstone
Print id of the car, from determineRaceLeader of object Silverstone

Set setRainingStatus TRUE
car 1 completelap silverstone
car 2 completelap silverstone
car 3 completelap silverstone
Print id of the car, from determineRaceLeader of object Silverstone

3. Class Diagram

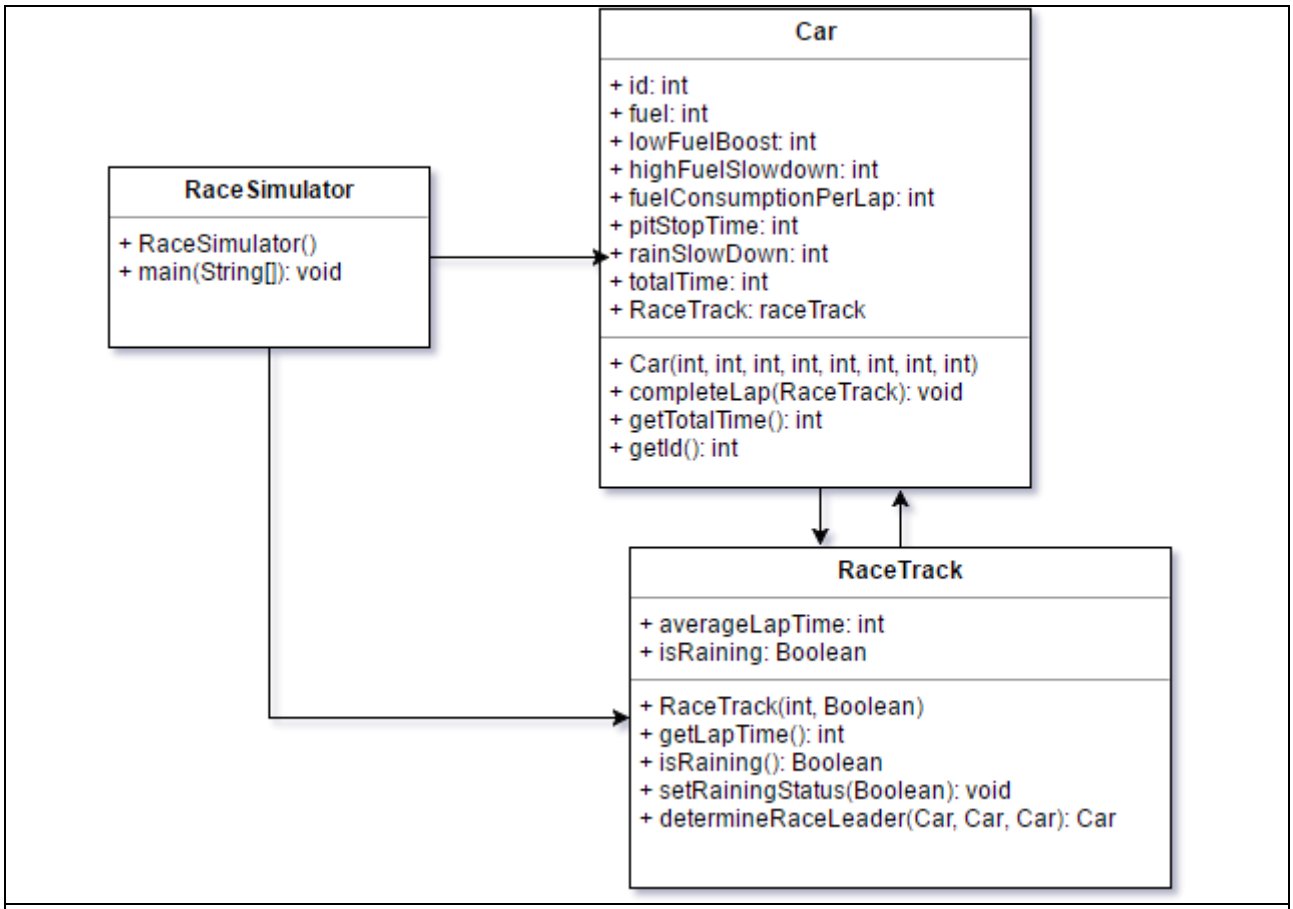

Figure 3-1. RaceSimulator class diagram

4. Description

Class RaceTrack is constructed with two variables, integer averageLapTime and Boolean value isRaining. This is assigned to the private field averageLaptime, and isRaining.

Method Car determineRace leader returns car1, if the total time of car 1 is less then or equal to car 3 and car3. car2 if the total time of car 2 is less then or equal to car1 and car3. And for all everything else, car3. This method has clear problem of being biased to car1. When cars have same lap time. For example, if car 1 and car 2 have the exactly same lap time the winner will always be car 1.

This problem could have been overcome with the following implementation:

1. Initialise new private integer field winnerCase and assign 1: having single winner, 2: having car1 and car2 being winner, 3: car1 and car3 winner, 4: car2 and car3 winner, 5: Every car being winner and return this field instead of Car from the method. By utilising Switch statement in the RaceSimulator, we could have been able to call multiple car winner ids. However, this would go over the given specification asking the programmer to return Car from the method determineRaceLeader.

2. Make determineRaceLeader to accept extra Car: determineRaceLeader(Car car1, Car car2, Car car3, Car winner), which the car winner is the winner from the previous lap. When there is a tie between all cars the method would return the previous winner. This will follow the

given specification but still have multiple problems. For example, when two cars are a tie and the winner of the car being the slowest for that lap.

However, realistically the lap time can never be the same for two cars so these extreme cases will never occur in practice. So the given method in my code is more than sufficient.

Class Car
The most interesting method here is completLap method. Checking for the amount of fuel left is at the end of the function so that any car will always start below the necessary fuel for the lap. It was interesting to observe difference in the intermediate lap winner depending on the placement of this function. This is one of the reason refuelling strategy is so important in F1. In real-life competition, each cars will be optimised in terms of fuel consumption for each race track and will not be refuelled 100% to minimise high fuel slow down time.

Class RaceSimulator initialises appropriate values from the specification and drive our program.