Software project report

Introduction

During the course of this semester the module of Software project has tasked the class with a three stage project. This report is written to expand and explain upon my methods and execution of my attempt of the project.

The general overview of the project is that we work in a company that has self driving taxis and the aim is to make travelling safer and more available while also saving commuters time.

Stage 1

Summery

Given a list of pre-booked rides in a city and a fleet of self-driving cars and a file of rides and their allocated rides. Computate a score based on certain parameters as listed below.

- The car score points based on the distance of the ride
- If the car is late to finish no points is given
- If the car starts and finishes in time a set bonus will be given

Design

In my final code the structure of the classes are as follows

1. Main

This class was predefined by the problem so that another system can feed the files into the program and have our program score the cars. Most of the code was already given beforehand and as such I will not be deeming it as my handiwork.

2. Simulation

This class in my program is designed to do the scoring and acts more as the main hub of the system. From here the program will send the file location to world and ride class and allocation class to do its functions, then proceed to prompt the car class to calculate the distance of the ride before evaluating the execution before giving a score and outputting it for the user to see.

3. WorldAndRide

This class will take the file and break down the data so that the program knows how large the place is, the number of cars, the bonus to be given, and then send the individual ride to the ride class.

4. Rides

Thai class breaks the ride data given by the world and ride class and stored the individual rides

5. Allocation

Take in the allocation files and split the data by lines and feeds them to the ride class before storing an array the resulting ride data

6. Cars

This class takes the information from the ride class and calculates the distance of the ride and if the car was early before returning it to the simulation class to be scored.

7. RideData

This takes data from the allocation class and stores the rides a certain car has which will be later used in the simulation class

Algorithm

This section into the car distance calculation and the simulation calculation

Simulation calculation

The simulation class takes both files before sending them to the appropriate classes. Then it takes the array list of rides and rides data from the allocation and world and rides classes. It also stores the set bonus from the world and ride class. From here on out it will iterate through the ride data calling on the rides per ride data and ask the corresponding car to check whether it reached the starting point early followed by asking the car to calculate the distance. From here it takes the distance then determines whether it finished on time before giving the score of the rides.

Car Algorithm

The car will take the location, destination and starting time from the simulation. Firstly it will check if the current position is the right position. If not it will move to the right position taking account of the time taken. The car will then check if started at the right time and return the results as a boolean value to the simulation class. This is then proceeded by the calculation of the difference between the location and the destination. This result was made into an absolute value then returned to the simulation class.

Testing/Evaluation

For stage one the testing and the evaluation i used mostly the test files that were made available to us through the module. This allowed us to see where the program made the errors and make relevant changes. I do have 2 of my own tests added for my own testing evaluation which can also be found on git.

Stage 2

For this stage I was to come up with the allocation file when the program was given the world and ride file.

Algorithm

I had to program split the work by iterating the rides over the number of cars which was given out as the ride order afterwards. This system looking back at it is very rudimentary and needs to be reworked in future as the rides are not optimal.

Design

Has a world and ride class to take in the file and break down the rides. Then the rides and stored in a array then sent to the simulation where the rides are iterated over and looped around the respective number of cars before the cars are printed out with their ride numbers

Reflection

From the completion of stage one i have come to realise a few issues with the way i had handled things and how i should move onward from this experience. Some of these include but are not limited to

- I had a lacking understanding of error types. After receiving errors via testing i came to a stand still trying to figure out what the error message means which lead to a huge time loss.
- Time allocation. I need to manage my time more efficiently to be able to complete the work with enough time to spare for re-evaluation
- Make variables easy to understand . After a long time of not looking at a certain part of the code it took me a minute before I came to understand what I was doing in my own code.
- The code for stage two needs way more work than I put in. this is also due to my time management which I have to look to correct in the future.

Moving forward I hope to correct my errors to make a more efficient system for coming projects.