**COURSE:**

**ADVANCED SOFTWARE ENGINEERING(F21AS)**

**COURSEWORK 2023**

**GROUP REPORT#1**

**GROUP MEMBERS(Edinburgh CW 21):**

* Danish Ali
* Zohaib Amjad
* Saud Ali Khan
* Ernest Oganji

**REPOSITORY LINK:**

https://github.com/epospiky/road-intersection-traffic-simulation.git

**SUMMARY:**

**CODE:**

* **SAUD ALI KHAN:** Function Requirements Part 1,2
* **DANISH ALI:** Function Requirements Part 3,4
* **ERNEST OGANJI:** Functional Requirement Part 5
* **ZOHAIB AMJAD:** F Functional Requirement Part 6,7

**REPORT:**

**DANISH ALI:** Development Plan/Final Report.

**ERNEST OGANJI:** Managing Repo/Testing/UML Diagram

All the participants have worked on report part #5,6,7 explaining their part of the code individually.

**BUGS/ERROR:**

This program meets the specification

**UML DIAGRAM:**

Diagram

Description automatically generated

**Traffic Simulator**

The Road Traffic Intersection Simulation is a program that simulates traffic flow at a road intersection. The program reads in data from a csv file containing information about the vehicles (such as their plate number, car type, and length), and uses this data to simulate the behaviour of all vehicles as they move through the intersections.

**Classes**

**GUI** :

This contains the codes responsible for the graphical interface of the program.It is responsible for displaying the two tables - vehicles and statistics - and updating them as the simulation runs. The vehicles table displays information about the vehicles in the simulation, such as their plate number(Vehicle Number), car type, waiting time, waiting length, CO2 emissions, status (waiting or crossed), and segment number. The statistics table displays information about the simulation as a whole, such as the total waiting time and length of vehicles, the total crossing time, and the segment number.We have an input table which will take input from users to enter new vehicle details and adding the vehicle to the vehicles table.

The GUI class also contains the CO2 TextArea which displays the total co2 emitted by waiting vehicles.

**Vehicle** - The Vehicle class represents a single vehicle. Each Vehicle object contains a VehicleData object that stores information about the vehicle, like Vehicle number(plate number), car type, and current segment etc

It contains several methods that are used to get and set specific information about the Vehicle. Below are the methods in the class;

* getPlateNumber(): This getter method returns the plate number of the vehicle.
* setPlateNumber(): This sets the plate number of the vehicle.
* getVehicleType(): returns the type of the vehicle.
* setVehicleType(): sets the type of the vehicle.
* getCrossingTime(): returns the crossing time of the vehicle.
* setCrossingTime(): sets the crossing time of the vehicle.
* getDirection(): returns the direction in which the vehicle is traveling.
* setDirection(): sets the direction in which the vehicle is traveling.
* getWaitingLength(): returns the waiting length of the vehicle.
* setWaitingLength(): this sets the waiting length of the vehicle.
* getCo2Emission(): it returns the CO2 emission of the vehicle.
* setCo2Emission(): sets the CO2 emission of the vehicle.
* getVehicleStatus(): returns the status of the vehicle.
* setVehicleStatus(): it sets the status of the vehicle.
* getSegmentNumber(): it returns the segment number of the road on which the vehicle is traveling.
* setSegmentNumber():This sets the segment number of the road on which the vehicle is traveling.

**Statistics** - The Statistics class contains five instance variables: segmentNumber, totalWaitingTime, totalWaitingLength, averageCrossingTime, and totalCo2Emission. The constructor created in this class takes in values for these variables and sets them. The class also has getters and setters for each of these variables.

It has the populateStatistics method that takes in a list of Vehicle objects and returns a list of Statistics objects. It has a HashMap that keeps track of the number of vehicles in each segment. It then iterates over each segment and calculates the total waiting time, total waiting length, average crossing time, and total CO2 emission for that segment. It creates a Statistics object with these values and adds it to a list of Statistics objects. It then finally, returns the list of Statistics objects. Below is the function of the methods in this class.

getSegmentNumber(): It returns the segment number of the Statistics object.

setSegmentNumber(): It sets the segment number of the Statistics object.

getTotalWaitingTime(): returns the total waiting time of the Statistics object.

setTotalWaitingTime(): sets the total waiting time of the Statistics object.

getTotalWaitingLength(): returns the total waiting length of the Statistics object.

setTotalWaitingLength(): sets the total waiting length of the Statistics object.

getAverageCrossingTime(): returns the average crossing time of the Statistics object.

setAverageCrossingTime(): sets the average crossing time of the Statistics object.

getTotalCo2Emission(): returns the total CO2 emission of the Statistics object.

setTotalCo2Emission(): sets the total CO2 emission of the Statistics object.

The Vehicle class also has a method called getWaitingTime() that returns the total waiting time for the vehicle (i.e. the amount of time the vehicle has spent waiting at a red traffic light or behind other vehicles). This method is used to calculate the statistics for the simulation.

Overall, the Vehicle class is responsible for simulating the behavior of a single vehicle in the simulation and updating its state as the simulation progresses.

**ADD-VEHICLE:**

The Add-vehicle table is used to take inputs from user so new vehicles can be add to the vehicles table.It has a top-level container which provides an input window for the user. The JLabel class is used to create text labels for the various components in the GUI, such as "Segment", "ID", "Type", "Crossing Time", "Direction", "Crossing Status", "Length", and "Emission Rate".

The JComboBox class is used to create drop-down lists of selectable options. In this code, JComboBox objects are created for selecting the type of vehicle ("Car", "Bus", "Truck"), the direction ("Left", "Straight", "Right"), the segment ("S1", "S2", "S3", "S4"), and the crossing status ("Waiting", "Crossed").

Jbutton has used to create two buttons:

* ADD
* CANCEL

I’ve used string[] to store the selectable options for the combo boxes.

### How to use the program

To use this program, you will need to provide a CSV file which has information about the vehicles. The file should be named vehicles.csv as this is the name of the file used in developing the program, and should have the following columns:

* Plate Number(eg TD2344)
* Car Type(eg Truck, Car, Truck etc)
* Length(in meters)
* Direction(Left, Straight, Right)
* Segment Number(S1,S2,S3 and S4)

Once you have the CSV file, you can run the program in a Java IDE such as Eclipse or IntelliJ IDEA. The program will display a GUI containing 3 tables: one showing information about the vehicles currently in the simulation, one showing statistics about the simulation (such as total waiting time and CO2 emissions) and the other for adding new vehicles to the simulation.

To start the simulation, click the "Start" button in the GUI.