CP2406 Programming 3 – Assignment 2 – Car Traffic Simulator Working document

Problem specification:

The problem is to design a car traffic simulator, at this stage the simulator only needs to include basic functionality utilising the core classes that will be needed to represent the basic objects such as cars, roads and traffic lights. The final program will have functionality to create and modify maps that comprise of a 2 dimensional layout of roads and traffic lights. However, for this stage of the program, the simulator will just comprise of 2 roads, a car and a traffic light. The simulation will be contained inside the main class for the first part of the assignment and produce some statistics about the simulation while it is running.

Problem decomposition:

In regard to access control, the road, traffic light and car classes will have private member fields. If this data needs to be accessed from outside the class will be returned using, get methods for each member field.

Road:

* The road class will need to be able to track car objects and where traffic lights are positioned. This will be implemented by dividing the road into segments. Each traffic light object will have an associated static road and segment in that road where it is located. Each car (or vehicle) object will move from segment to segment which will allow its speed to be roughly calculated.
* Member fields:
* Speed limit to represent the maximum speed that cars will be allowed to travel at.
* Number of segments, as the length of a road is set out by the project requirements of being between 6 to 15 (units) times the length of a car (2 to 5 times the bus length). Let 1 unit be the length of a road segment, thus roads will range from 6 to 15 segments long. This is an intrinsic characteristic
* Orientation, either horizontal or vertical. This isn’t an important variable for the initial part of the simulation but will be needed later to graphically position the road objects.
* Road ID, this will be a 2 dimensional value to represent the position of the road graphically. But, for initial functionality this isn’t important as the simulation will only consist of 2 roads so the position can be represented 1 dimensionally for now.
* Next Road ID, this will indicate what this road will connect to, later on multiple roads may be connected. If this field = 0, then current road is the last road.
* Methods:
* Overloaded constructors to manage the assignment of variables to the object. Its method signature will consist of the method name (same as the class name, Road) and the multiple parameters that it takes.
* Methods to return the values of the roads speed limit, number of segments, orientation and position of the road. The method signature for these methods is just the name of the method, as they take no parameters. These methods can be private as they return the value of their associated parameter.

Car:

* The car class will need to be able to represent a car object with its position (what road and segment of road is the car currently on), the car will also have some internal characteristics, the cars length and width. The length and width of the vehicle will be altered if the vehicle is a bus or a motorbike (this will be accomplished with an overloaded constructor that will be implemented later). The length of a car will be used to determine the values of length for all other objects.
* Member fields:
* Type, this field will allow for the implementation of motorbikes and busses later on and will identify what type of vehicle it is.
* Road number, to identify which road the car is currently on.
* Segment in road, which segment the car is currently located.
* Length of road, will be needed later for calculation of speed.
* Methods:
* Two overloaded constructors to allow a default vehicle (car) to be created and its attributes assigned and another which will allow a bus or motorbike to be created.

Traffic light:

* The traffic light will need to switch between go and stop modes. It will be located on a road and a position on that road.
* Member fields:
* Status, a Boolean value to represent the red or green status of the traffic light. This may be changed later to also implement functionality for a yellow status.
* Road number, to identify where the traffic light is located.
* Segment in road, which segment of the road is the traffic light located.
* Methods:
* Get status, returns the status of the traffic light. True to represent green light and False to represent red light.
* Get road number, returns what road the traffic light is located on.
* Get segment of road, returns the segment in the road that the traffic light is on.
* Change status, this inverts the status of the traffic light.

City:

* The city will need to store a layout of roads and traffic lights. Possibly by storing and reading data about the layout of the city from an external file probably in JSON form.
* Member fields:
* Roads, a collection of road objects and their location information.
* Traffic lights, a collection of traffic lights objects and their location.
* Methods:
* Get number of roads, returns the total number of roads on the map.
* Get number of traffic lights, returns the total number of traffic lights.
* Add road, creates a new road object and stores it in the city file.
* Add traffic light, creates a new traffic light object and stores it.

The road, car, traffic light and city classes will also need to have associated unit test classes. Which

Main:

* The main class will act as the simulator for the first part of the assignment. Which will just create 2 roads, a traffic light and a car object. The program will display information about the objects while the simulation runs.

Methods:

* A drive method, which will drive all cars in the simulation if able to and update each car object. This method will also handle the car changing road.
* Check car status method which will display the position of each car in the simulation.
* Update traffic lights which will take a rate of change variable, this represents the percentage chance of the traffic light changing its status.

Part 2 of this assignment involves implementing the basic functionality of assignment 1 and integrating it into a self-contained GUI. As well as allowing the user to create and modify the layout of roads and traffic lights and control the spawn rate of cars.

I have had significant trouble regarding the functionality required for assignment 2, which still have not and probably will not be satisfied. Functionality has been implemented to read and store road and traffic light information from/ to a csv file, but this could still be improved to encode the position on screen that these roads would be drawn. As the program is now, it iterates over each line in the csv, drawing roads consecutively instead of associating positional data of the roads in the csv. There is no functionality to allow a user to create or edit a map and the data is currently only able to be read/ stored in one file. Finally, the program still outputs through the console and is unable to update the GUI element, there is significant bottom layer code for this functionality (the repaint() function is quite extensive but is only able to be called during its initialiser and doesn’t update with the simulation).

There are several reasons why the required functionality has not been achieved for part 2 of this assignment. The primary reason being the inability to repaint the main window as the simulation runs, this problem arises because the main function that the simulator ran from in part 1 of the assignment is a static function and the repaint function is not, therefore it cannot be directly called from main. I have tried using an external Boolean value to control this, but a non-static value cannot be updated from a static context and so the problem doesn’t disappear. I have also tried creating an instance of the Main class from within the main function and calling main.repaint() in order to keep the main function as the simulation loop, but this also didn’t work. The partially working solution that is currently implemented, is that the simulation loop was taken out of main and embedded in its own function, which is called from the run simulation button. The issue with this is that the stop simulation button does nothing. Another issue is that while the start button is able to initiate the (console output only) simulation, the button remains pressed until the simulation totally ends (when there are no more cars left) because of this, the JFrame also doesn’t respond to the exit button as it is still running code associated with the button press. Furthermore, this second stage of the assignment should implement a city class to control and handle the layout of the map on the backend side of things.

This leaves me at an impasse for this assignment and I am unable to achieve the required functionality, while there is a lot of functionality implemented (such as the save/ load functions from a csv file) that lays the foundation for a complete simulator, I was unable to utilise this functionality to fully achieve the requirements for the assignment.