# Problem Specification

**What is the problem?**

The problem given was to build a traffic simulator for a city and implement a graphical user interface using swing. The program will need to be able to have a customizable map, of varying sizes, and allow the user to build the city during the ‘map editing’ section of the simulation. When running the vehicles must be able to automatically enter the map, and travel through the city with random destinations without hitting another vehicle and maintaining all Australian road rules.

**How will the program be used?**

The program will be used to help design and plan out a city. This will be done by displaying useful information to the user while the simulation is running, such as the total number of cars on the simulation, and the average speed of each car in the simulation.

**How will the program behave?**

When the simulation is running the program will update incrementally, with all vehicles updating their speed first (accelerate, deaccelerate and stop vehicle) followed by changing the road direction if require. Once they have been calculated the vehicle will move based of them, moving after each tick. Whiles the simulation is happening, cars can only enter from sides of the maps and most obey all Australian speed limits, and give way / brake appropriately when needed and maintain a safe distance between other vehicles.

**Graphical User Interface (GUI)**

For the gui a JMenu bar will be used to navigate the program, with a status bar to display what is happening throughout the program and errors that occur (entering incorrect values, map not found).A BorderLayout will be used to position these items, with the menu bar at the top and the status bar at the bottom. To display the map a gridLayout, assigned to the center, was used with each road piece being placed in the appropriate part of the grid. JOptionPanes will be used to get user input and a JSlider will be used to select the entry rate of vehicles.

**Map design / movement**

While the initial map design was left the same, using a grid system (e.g. 4x4 / 5x5 / 10x10) for each road piece, with each road piece having directions for how the vehicle will move (e.g. up, turn left or right). The scale of each peace was also adjusted from 20 to 100, allowing more possible locations for the car to go, smoothing out the animations from how it initially was.

**Map Editing**

To edit the map a JButton class was created (MapPieces) which sets a JPanel of radio buttons visible (and not visible when finished / exited) to border east which is used to assign an image icon to the JButton based on the radial selection when the button is clicked. When the button is clicked the appropriate road piece is then added to the roadArrayList to be used later in the simulation.

**Data design (types of lists & saving)**

To run the simulation all data during the map editing stage (Road pieces and traffic lights) are saved into an Array list (which are saved to separate txt files if chosen to be saved). These lists are then passed to the simulation, with vehicle list changing throughout the simulation (location, location on road and speed) and the traffic light list and road list effecting how the vehicle moves throughout the simulation.

# Class Design

**Road**

The road class was decided to consist of 3 constructors, name, orientation and location which allows the simulation class to see what type and where the road piece, as well as what way the piece is facing (e.g. for 3-way intersections and where the exists are).

**TrafficLight**

The traffic light class was built with multiple constructors to facilitate a variety of types of light cycles and road types. The location and road location determines where the traffic light is on the map and road piece. The roadName attribute is used to determine how the traffic light responds (e,g, 3-way and 4-way intersections had different light cycles) with the traffic light cycle and number used to track where the lights are on the intersections, and what part of the rotations they are through. Methods for both 3-way and 4-way intersections have been added to cycle through the different combinations turning each light green in red from the simulation using the changeColour method, which changes the lights colour. For ease of coming a 2 colour light system has been utilized until a prototype of the program is running, where amber will be added to the traffic light colors. To stop traffic lights changing colour straight away, a colour timer was added with a method that ensures that the TrafficLight waits a preset amount of time before being allowed to change.

**Vehicle**

The vehicle class was built using the id constructor to allow references to a specific vehicle (to add it to the map), and the name to references what type of vehicle it is, in combination with length based on the size of the vehicle. To determine where the vehicle class will be located, 3 constructors where used, location and road Location like the previous classes, and road Direction to determine which way the vehicle is going, and what side of the road, and where on the road for intersections. Whiles the road Side was initially used in code, it was released that it was redundant, and road direction could be used in the same way, and will need to be removed from the class. The chosen direction constructor is used when the vehicle needs to choose which direction to go (such as 3-way and 4-way intersections) and will correspond with which exit it should take on the road. For now the Speed constructor has been simplified to a 1 to 5 scale for speed to allow the cars to move throughout the map, however will need to be redesigned to something more precise later on. To change the speed of the vehicle to methods were made, accelerateVehicle and decelerateVehicle, with the methods increasing the speed (up to 5) or decreasing the speed (down to 1) respectively. A stop vehicle method was use to set the speed to 0, which will stop the vehicle moving. To move the vehicle a basic method was added which adds the speed of the vehicle to the road location of the vehicle, which moves it across each road piece. Each vehicle in the simulation has been given a length based on the road length (e.g. bus Length = 1000/5) to be used in the simulation to keep vehicles evenly spaced (considering the vehicle in front from the back of the vehicle).

**Simulation**

Whiles the simulation class was initially used to run the simulation by extending the timer class, this was changed back to running it out of the main class, due to errors occurring when trying to paint from the simulation class and extending JFrame and would need to redesign how my simulation is run to not have extend both timer and JFrame.

Before the simulation begins the current loaded map is converted from the MapPieces class to the MapPiecesRun class which changes each map piece button to a JLabel for the simulation to run, with each position being calculated based on the dimensions of the grid divided by the map size.

The simulation was updated at a quick rate (1ms) to allow the cars to smoothly transition. The simulation class functions by running through a serious of array lists based on a variety of conditions(location, direction, road location and road name) to determine where the vehicle should be located, and how (or if) it should move locations. Before moving a vehicle, it will consider other vehicles locations (and length) that are in front of the vehicle (road location of current map piece and next map piece) and traffic lights. If nothing causes the vehicle to deaccelerate or stop, the vehicle will accelerate up to 5 (max speed limit. Before the simulation begins moving vehicles, the code for the traffic light rotation is run, and then checks for how many vehicles then user entered, then allows them enter through each section of the road by running 4 methods, enterTopMap, enterBottomMap, enterLeftMap, enterRightMap which checks for road pieces that are valid entry points, then checking to make sure there is no cars at the entry points using checkVehicleList, then adding the vehicle if no vehicles are there to the road using addVehicle.

Once vehicles are running the simulation moves around the vehicles based on various conditions , calculating speed based on what direction it is going and whether or not there is any vehicles in front of it, where the vehicle wants to go (based on a random number generator) and having to give way, and traffic lights. When this is done the vehicles are then update and the painted to the appropriate location. Once the simulation has run through and all cars have left the map or the user stops the simulation.