# Problem Specification

The problem at hand is the need for a program that can take a user produced set of roads and traffic lights and simulate how traffic will act on the given configuration. This program should be used to develop efficient simulations that show how traffic acts given certain road configuration, this will help city planners identify build up points and areas where traffic control objects aren’t being used efficiently.

Additionally, this program will be used as a demo for the Programming 3 stall at JCU open day, as this will be a public demo code should be reasonably readable and fully commented.

# Classes

## Main

The main class should act as a primary controller for the simulation. It controls the primary JFrame and all contents within in. it is the first class to run and all other classes and frames are called from this class. Main has the following member fields

* private static Timer timer – timer use to countinuosly repaint the sim, also used for updating car positions.
* static JFrame mainFrame – the suitably named mainframe of the entire sim, all other Jframes and classes are controlled by controls on this frame. It is package-public for use it other classes that need to repaint it. Its static because it is used within the static main class
* private static Draw draw – Custom Jpanel, responsible for the visual element behind the simulation. Uses paint methods from car, road and children of road to define what is painted to the screen. It is private and static because this instance of draw is only used within Main.
* Private static JFrame createSim- Secondary JFrame in this program, used to edit and create new simulation configurations. It is private and static because this instance is only used within Main, and passed into the CreateSimPage.
* Private static CreateSimPage cs;- instance of CreateSimPage, used as a controller for the JFRame
* private static LS ls- Instance for LS, responsible for loading and saving configurations.

### Methods

#### main()

As discussed above Main is responsible with instigating other classes for use within the simulation controlled in this method. Main creates and shows the mainframe or homepage, and instigates any swing UI elements required for the GUI. Main also adds the appropriate action listens and points them to methods in the Main class.

#### load()

Uses the timer object to update positions of cars and change traffic lights. Also triggers a repaint every cycle to have the appearance on animated graphics. It static and private as it called in the static method main, and is only used within the Main class.

#### car()

spawns a new car object and sets its starting point the same of that of the first road in the Roadlist.index ArrayList. Set to private and static.

#### start()

Starts the timer object. Set to private and static.

#### stop()

Stops the timer object. Set to private and static.

#### csShow()

sets the createSim JFrame to visible. Set to private and static.

#### saveSim()

calls ls.save(). Set to private and static.

#### loadSim()

calls ls.load(). Set to private and static.

## Car

The car class is responsible controlling all aspects of a car instance. Elements like X and Y position, speed and the current road coinciding with position of the car.

* private double currentSpeed;- Sets the current speed of the car used as a modifier when adding to x or y position. Private for use within only this class
* private Road current; - the current road that the
* private double x, y;
* private Road next;
* private boolean direction;
* private Random random = new Random();

### Methods

#### updatePosition()

this is the primary method for moving the car along a road. This method performs a few functions. First it calls the method currentRoad() to set the cars current road and nextRoad() to find the cars next road.

#### speedUp()

This is a setter method that is called within updatePosition() used to add 0.5 to the cars current speed.

#### slowDown()

This is a setter method that is called within updatePosition() used to subtract 0.5 to the cars current speed.

#### Paint(graphics g)

Defines what the car looks like when painted.

#### currentRoad()

Uses the cars position to find the road it is currently on

#### nextRoad()

Use the cars current roads endpoint to find the next road in the sequence.

#### move()

changes the cars x or y position based on the roads orientation and the cars direction.

## Road

The road class is set to define the length of the road, and the starting x and y position within the orientation, with methods to figure out the finishing x and y position on the road. Additionally the road will store an orientation value, so the car knows to drive in X or Y direction.

* Double length- defines the length of the road, depending on orientation this will span in the X direction or Y direction. This variable is set to package-private as it is needed by some Jframes in different classes
* Double xStart, double yStart- defines the starting x and y position of the road. These use double instead of int to account for later iterations where acceleration and deceleration will be implemented. This member field has been set to package-private as it will be used by the car class to define which instance of road a car is on.
* Double xFinish, yFinish- defines the finishing x and y position of the road. Using a default constructor these variables they will always be set as their starting value (described above) plus the roads length so depending on orientation they will always be the lengths value away from the start. As with xStart and yStart these has been set to package private for use in other classes.
* String orientation- defines if the road is horizontal or vertical, this will be used by the car class the decide whether to move in the X or Y direction thus making it a public field.
* Boolean Selected- used to influence the paint method, so that users can see what road is selected when editing configuration.

### Methods

#### setxFinish()

Takes the product of xStart and length as an input and sets uses it to set the current instances xFinish as the input.

#### setyFinish()

Takes the product of yStart and length as an input and sets uses it to set the current instances yFinish as the input.

#### Paint(graphics g)

Defines what the road looks like when painted.

## Traffic Light

This class controls the traffic lights within the simulation. At this point in development lights can only be red or green and only one light for each instance can be green. Traffic Light is a sibling class of road, so it will store all the same information as a road instance with a few added member fields this is done because like roads the car class needs to identify its current Road/Traffic light so that it can know when it needs to slow down of speed up. A traffic light has 4 lights, one for each direction a road can be attached.

* String leftLight
* String rightLight
* String topLight
* String bottomLight

These variables represent the colour of the light, directions are from a top down perspective. All light variables are set as public for use within other classes.

### Methods

#### change()

when called this method generates a random number bound at 4 and uses a switch statement to decide when to change a light. This method acts as a setter and doesn’t return a value instead it changes the current instances member fields based on which case of the switch statement is executed.

#### links(Road next)

called within cars updatePosition() it is used to where the car will go after an intersection. It takes the cars next road (if it is an traffic light) and returns an array list of all roads directly connected to the next.

Paint()

Overrides roads paint method to show what a traffic light looks like.

## RoadList

This classes only purpose is to store an ArrayList of Road and children of Road instances that will be used by other classes to find the starting and end points of the current simulation. This class has a single member field:

* ArrayList index<Road> - An array list that stores Road and child instances. This variable is public and static, so that it may be used across classes. It is static because we need to ensure that all classes are working with the same list of Elements, a static variable allows a single instigation to be used program wide without having to pass the instance into every class.

### Methods

#### addElement()

Takes a road instance as an input and adds it to the static variable *index.* Method is set as public for use within other classes

## CarList

This classes only purpose is to store an ArrayList of car instances that will be used by other classes to find car positons. This class has a single member field:

* ArrayList index<Road> - An array list that stores Road and child instances. This variable is public and static, so that it may be used across classes. It is static because we need to ensure that all classes are working with the same list of Elements, a static variable allows a single instigation to be used program wide without having to pass the instance into every class.

### Methods

#### addElement()

Takes a car instance as an input and adds it to the static variable *index.* Method is set as public for use within other classes

## Draw

A custom JPanel, used as a method to bring all paint methods from various cars and roads into a single panel making repainting easy

### Methods

#### paintComponent(Graphics g)

a special method called automatically, it loops through all current roads and cars and calls their paint method.

## Intersection

A child object of road, it is used as a connector between other roads.

### Methods

#### Links(Road next)

called within cars updatePosition() it is used to where the car will go after an intersection. It takes the cars next road (if it is an intersection) and returns an array list of all roads directly connected to the next.

## LS

The method that controls loading and saving csv files made from configurations within the application.

### Methods

#### save()

uses a StringBuilder to build each line of the csv file and then writes each line to a FileReader object. Loops through each road and builds a line for each. Filereader then writes the file to the path defined by a JFileChooser.

#### load()

again uses a JFileChooser to get file path. Uses a Buffered file reader to read file on line at a time, splitting by commas then created a new road, traffic light, or intersection object and adds then to the RoadList.index.

## CreateSimPage

The page responsible for editing and creating new simulation configurations. The Jframe has controls to add and remove roads and child objects. Uses mouse events to set a road to selected and modifies them from their

### Methods

#### load()

takes the passed in CreateSim jframe and adds all swing UI elements and actions listeners to it.

#### moveRoad()

moves the selected road based of a key input of either W, A,S or D

#### roadClick()

sets road.selected to true if mouse click x/y is the same as a road in the sim

#### refresh()

refreshes selected road based on textbox values

#### save()

saves the configuration and closes the Jframe

#### addRoad()

created a new road object and adds it to RoadList.index

#### addTrafficLight()

created a new TrafficLight object and adds it to RoadList.index

#### addIntersection()

created a new Intersection object and adds it to RoadList.index

#### repaint()

uses a time to repaint the JFrame

#### removeSelected()

removes the selected road from Roadlist.index