# 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.

# Classes

## Main

The main class should act as a primary controller for the simulation. Because of the console nature of the program (at this point in development) it is responsible for creating all instance of classes used within the simulation. Instances for Road, Car, Roadlist, and TrafficLight are defined with set values for testing purposes within main.

### Methods

#### Main()

As discussed above Main is responsible with instigating other classes for use within the simulation controlled in this method. At this point the simulation uses 2 roads, one vertical and one horizontal split up by a single traffic light, as well as a single car that travels that route. This method uses a infinite while loop that contains a try/catch statement that will try call 2 methods car.updatePosition() and trafficLight.change() which are responsible for moving the car and updating the traffic lights. The catch statement catches any IndexOutOfBounds errors which occur when car.updatePosition() attempts to access the next element in the list of road objects when it has already reached the end, this catch breaks the loop and ends the simulation.

## 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.

* String type- This is unused in this version, as there is only a single car n the simulation for testing purposes thus making it private.
* double X and Y- marks the x position of the car, this is used to identify when the car is moving vs still and that the car is moving in the correct direction with respect to the current road’s orientation. Both have been set to public for use within main and testing classes
* double currentSpeed- the current speed the car is travelling, this is used to decide if the car needs to be moved or not. Variables uses a double to make implementing closer to reality acceleration and deceleration.
* Road current- By default this is initialized as null, although it uses the private setter method currentRoad() to set a value. This variable stores the current Road/Traffic light instance that the car instance is on. This variable is only used by methods within the Car class, making it private.

### 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, next it will check whether the car is at the end of current road by comparing the cars X and Y to the roads xFinish and yFinish. If this is the case the method will then check if the next Road is a traffic light and that the corresponding light is red or green and based of the light the car will stop and wait or continue driving. Finally the method will add 1 to x or y depending on the current roads orientation provided that the car is moving. This is a public method that is used within the simulation loop in main.

#### speedUp()

This is a setter method that is called within updatePosition() used to set the cars currentSpeed to 60. Method is set to private as it is only needed within Car class.

#### slowDown()

Similar to the speedup() method, this is a setter method that is called within updatePosition() used to set the cars currentSpeed to 0. Method is set to private as it is only needed within Car class

## 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 private as other variables primarily work with Start and finish locations to define positioning.
* 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 public 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 public 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.

### 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.

## Traffic Light

This class controls the traffic lights within the simulation. At hist point 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 left, 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.

## 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 TrafficLight 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