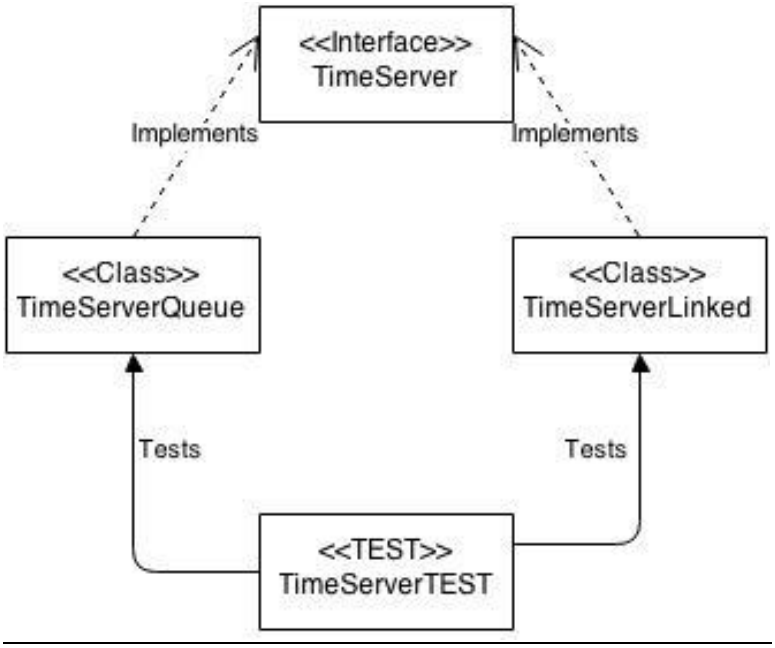
Steven Kiley

SE 450

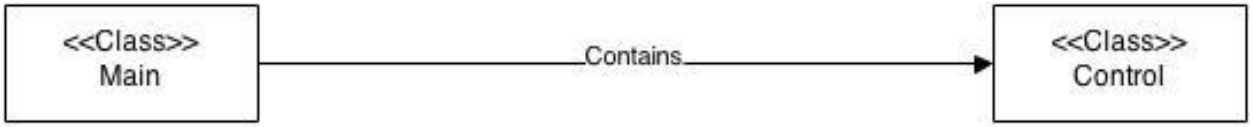
Traffic Control Project

CLASS DIAGRAMS BY PACKAGE

util package

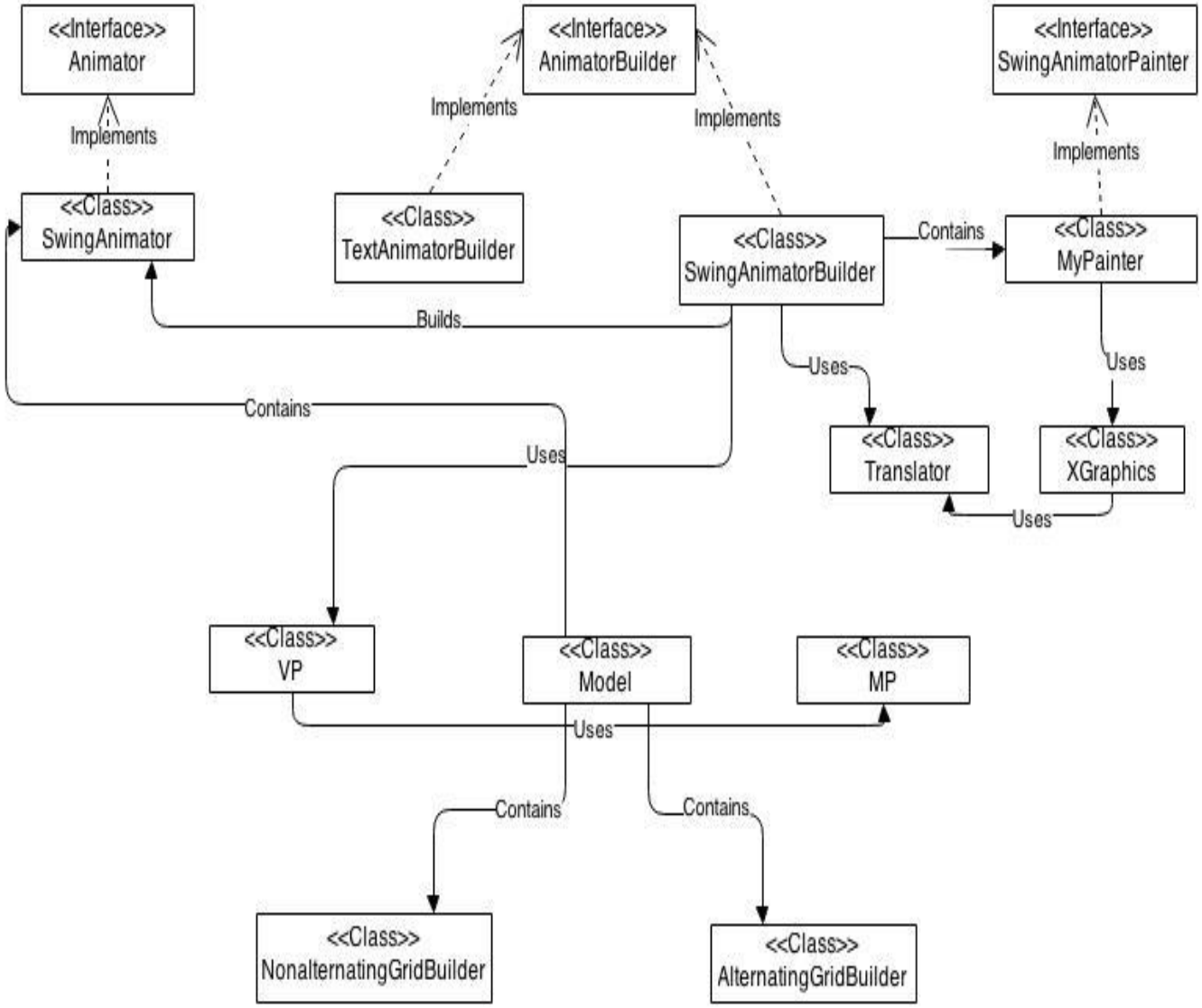


main package

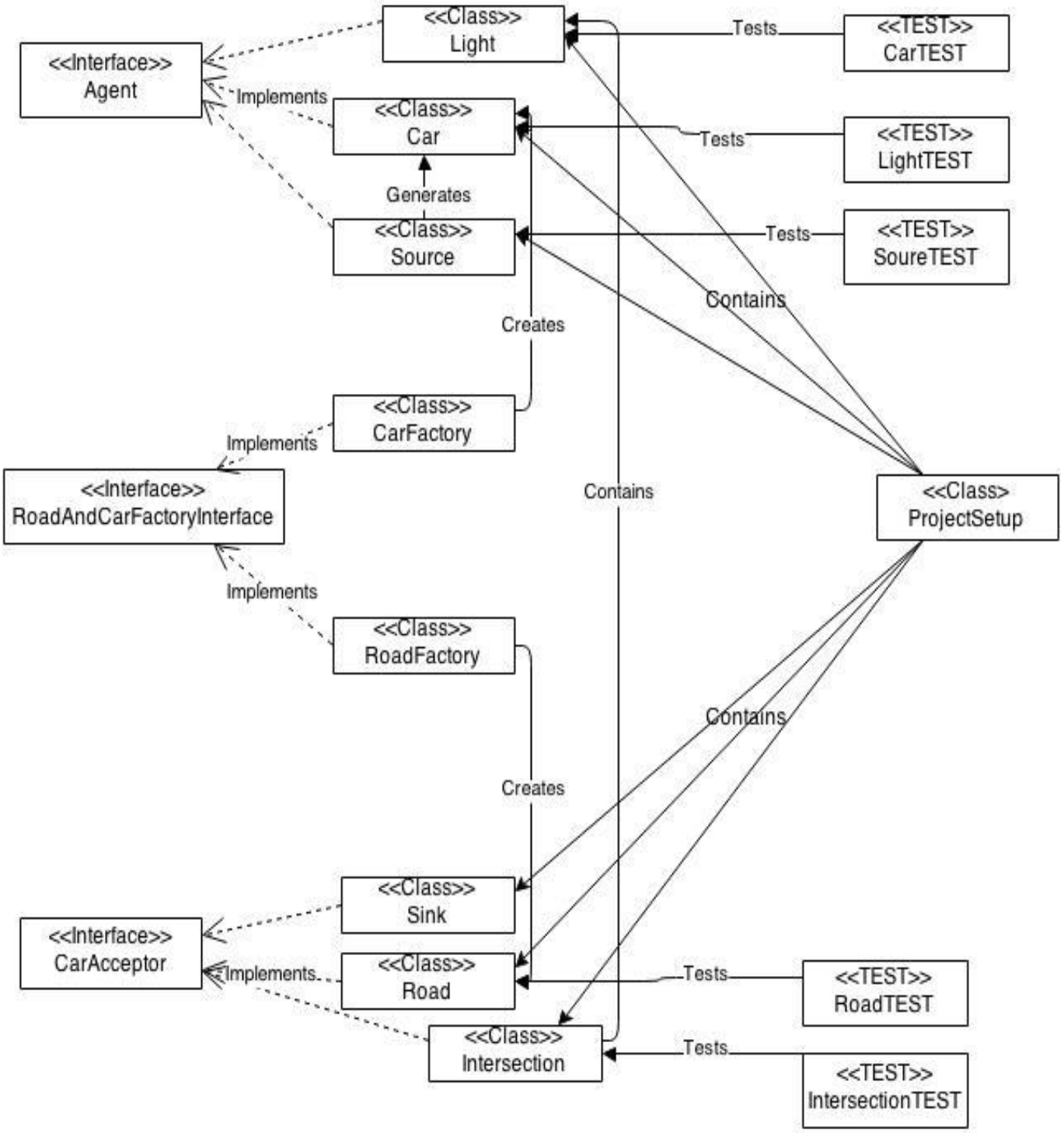


command package

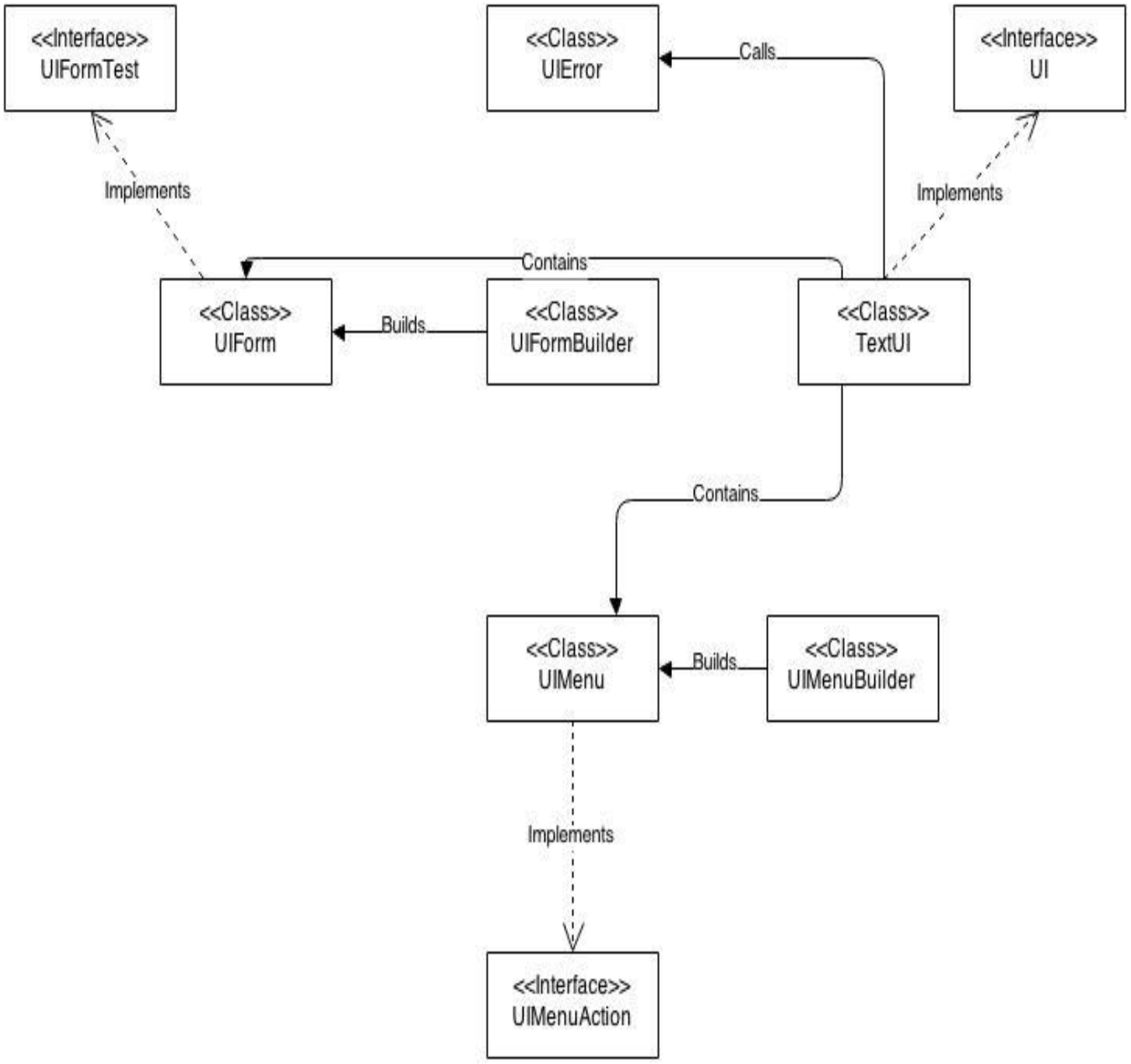
animator package



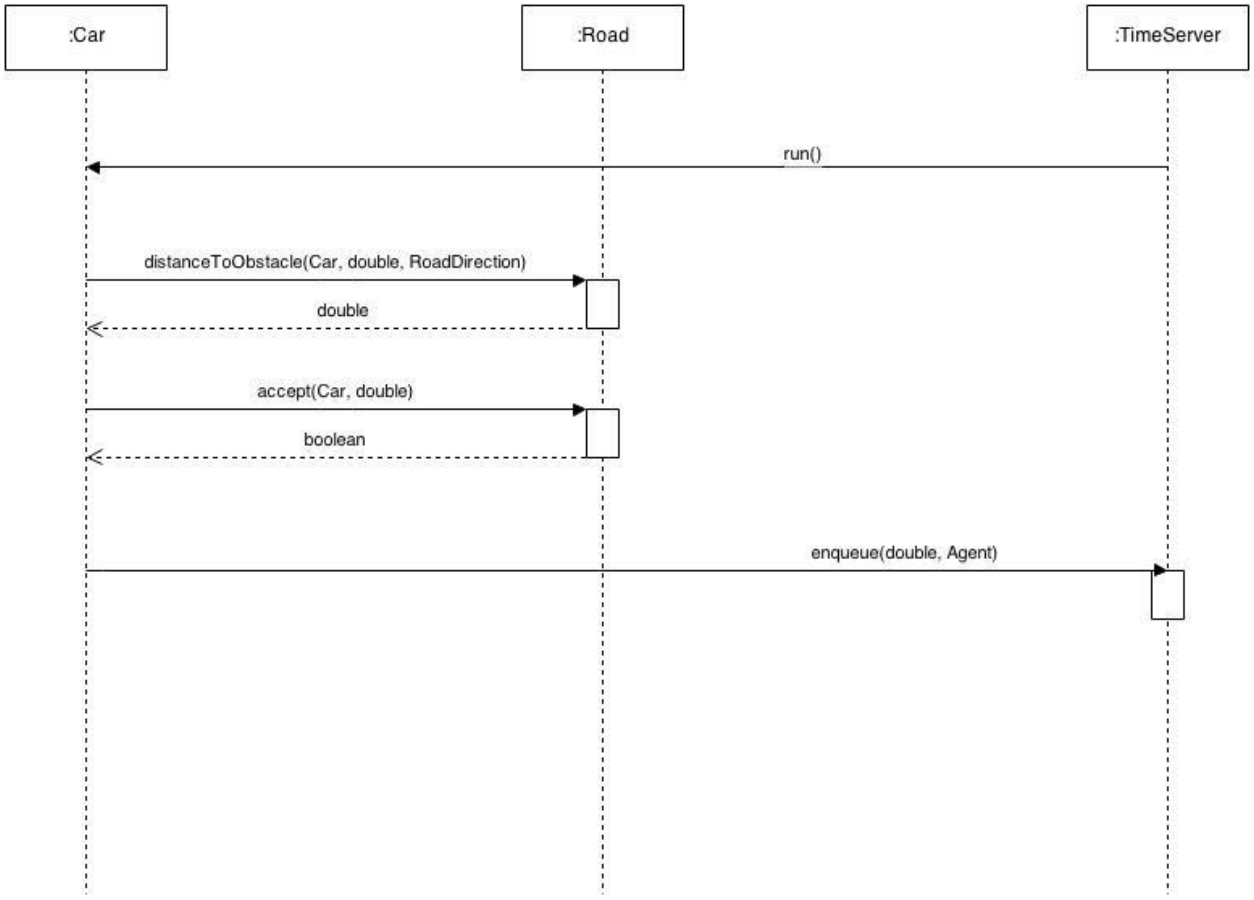
model package



userinterface package



SEQUENCE DIAGRAM



TIME SUMMARY

Week	1	2	3	4	Total
Design	5.75	13	8	0	26.75
Code	9.5	9.75	20.5	7.5	47.25

```
Bigbug  | 2.75 | 0    | 13.25| 4.75 | 19.75
                        Grand Total: 93.75 hours
```

NOTES ON PATTERNS

There are quite a few patterns used throughout the project. The most obvious are the Factory Pattern and the Command Pattern. The Factory Pattern is used in the "model" package to pass Car and Road objects (package private objects) to code outside of the package. In the "command" package you can see the bulk of the Command Pattern objects. These objects are used with the "userinterface" package to allow changes to the default simulation parameters. Other patterns of note are Observer/Observable in which the animator is alerted to changes in the model so it may alter the graphical output accordingly. The whole thing acts as a sort of Model-View-Controller, as there is a data based model(conveniently called "Model", a view (the "Animator"), and a controller that acts as a go between (via the Command Pattern). Builder Pattern is used several times in the project, in both GridBuilder files and in the lovely Assignment 2 files I borrowed named UIFormBuilder and UIMenuBuilder. As a special note, there is a section of code where I use a somewhat complicated switch. It would have been preferable to set this up with a Strategy Patter, but unfortunately I did not have time to do this and stuck with the simpler and quicker switch statement.

SUCCESSES AND FAILURES

This project took a tremendous amount of work, but the end product runs fairly well. It took a few attempts at building the project to get things in a good place, and there was much crying and gnashing of teeth. My first attempt was very ill advised. It was a purely iterative approach (one you said in class NOT to do) that resulted in a Car objects that navigated their roads heroically, but were patently unable to handle a stop light. The whole thing was clunky and had major functionality problems. I didn't understand this, because I had taken a lot of time to design things. How could they not work? Well, if we're speaking of "lessons learned" then I learned that you cannot design something if you don't really know how it works. My lack of understanding regarding Observer/Observable and the use of Swing in Java made my plans meaningless. Naively, I thought I could design the animation files from scratch myself. This was a mistake and cost me quite a bit of time. After repenting and deleting my hopelessly tangled files, I decided to do two things. First, I spent some time running through some online tutorials of Swing and basic animation in Java. Second, after completing the tutorials and took a second look at the files provided in the example project. It took me a while to trace everything that was happening in them, but I was able to, eventually understanding them enough to adapt them to what I was doing with my new Car, Road, and Light objects.

The second attempt was decent but I ran into a problem. My animation was happening fast. Like, very fast. Just flashes on the screen. The ultimate solution was to use sleep() to make the thread wait each time the timeserver executed the run() method of an Agent. Unfortunately, by the time I'd figured that out I had made a mess of the elegant files I had originally created. This called for a third and final reboot of the project. Here I was able to create a successful animation and simulation.

What Works and What Doesn't

Using the default parameters (located in ProjectSetup.java) the simulation runs wonderfully and the animation thereof is pretty good. There are no compiling errors or crashes at runtime using the default parameters. The cars in the animation do stop pretty far apart from each other, the backup from a traffic light doesn't take many cars to fill up the whole road because of this. I attempted a few things to alleviate this but couldn't figure it out. I was also getting an error where the cars updated position was returning as a negative number, which threw an error. I wasn't able to fix this so I just decided to add an if statement that turns negative numbers into a zero. This causes occasional irregularities but otherwise is not a problem.

The menu options to change the default parameters for the simulation all seem to work. Honestly, I have not had much time to test them so there may be values for which there are problems.

Overall, the project works much more than it doesn't. I was very pleased that the animation works, the stoplights change colors, the cars move and slow down and stop. There isn't anything that is glaringly "broken".