Brian Vegh

CMSC 335

December 15, 2020

Project 3 Documentation

**USER GUIDE**

**Description:**

**Java Swing application that uses event handlers, listeners and incorporates Java’s concurrency functionality and the use of threads. This application displays time, traffic signals and other information for traffic analysts. The GUI design includes viewing ports/panels to display the following components of the simulation:**

1. Current time stamps in 1 second intervals

2. Real-time Traffic light display for three major intersections

3. X, Y positions and speed of up to 3 cars as they traverse each of the 3 intersections

**Details of the Simulation:**

1. The components listed above run in separate threads.

2. Ability to loop through the simulation with button(s) providing the ability to start, pause, stop and continue the simulation.

4. Assume a straight distance between each traffic light of 1000 meters.

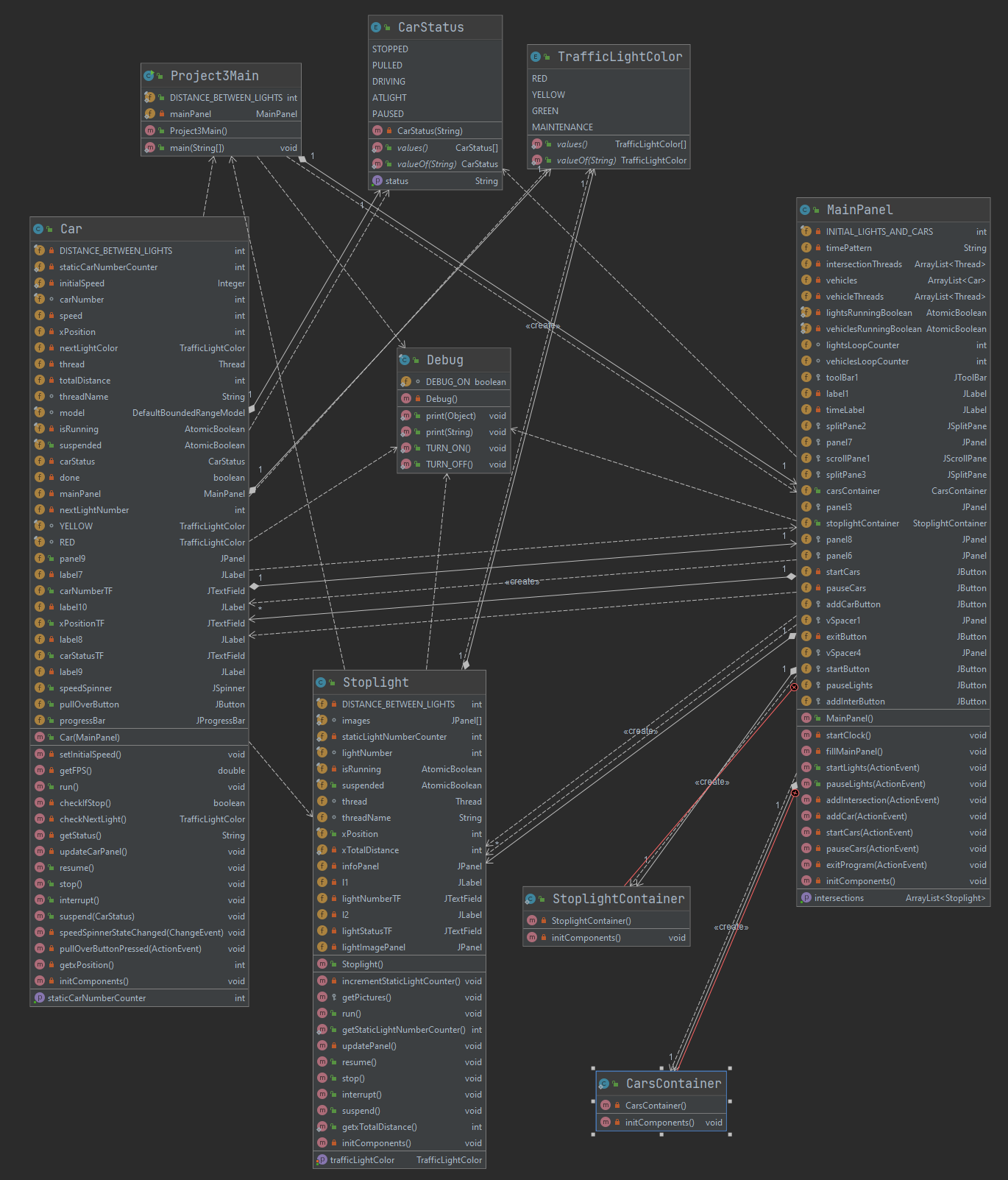
5. Assume Y = 0 for your X,Y positions.

6. Provides the ability to add more cars and intersections to the simulation through the GUI.

**Operating Instructions:**

1. Open the program source folder.
2. Click on the “Run Project3.bat” file to execute the program and open the program’s Javadocs.

**UML DIAGRAM – main package**



**TEST PLAN**



**LESSONS LEARNED**

**GUI Thread Hierarchy:**

When dealing with concurrent threads, they must be passed by reference. Any class that is executing concurrently must have synchronized getters and setters if there is a chance that other classes will try to access a variable at the same time. By synchronizing methods, I was able to prevent deadlock and thread false updates of values within concurrently running objects.

**Creating Runnable JPanel components:**

Because each Swing component primarily executes code on the Event Dispatcher Thread, to achieve concurrency, I created each threaded runnable Panel as an independent class. This way, every Stoplight can be addressed directly, and be allowed to invoke its own individual SwingWorker, and be paused and resumed independently. Rather than linking many java class files to a single GUI Panel, each Stoplight and Car operates within its own Panel. This also allows for mobility of code; I could implement the Car panel or the Stoplight panel anywhere a JPanel can be created, and it will run perfectly fine. If I wanted to connect them to a racing game, I could add them on the bottom of another application very easily, without having to recode any GUI.

**RELEVANT PROJECT IMAGES**

