Project 3

CMSC 335 6381

Professor Mujeye, Douglas

Wanner HernandezR

March 8, 2021

**Assignment Details**

As a new engineer for a traffic congestion mitigation company, you have been tasked with developing a Java Swing GUI that displays time, traffic signals and other information for traffic analysts. The final GUI design is up to you but should include 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

Some of the details of the simulation are up to you but the following guidelines will set the guardrails:

1. The components listed above should run in separate threads.
2. Loop through the simulation with button(s) providing the ability to start, pause, stop and

continue the simulation.

1. You will need to use basic distance formulas such as distance = Speed \* time. Be sure to be consistent and define your units of measure (e.g. mile/hour, versus km/hour)
2. Assume a straight distance between each traffic light of 1000 meters.
3. Since you are traveling a straight line, you can assume Y = 0 for your X,Y positions.
4. Provide the ability to add more cars and intersections to the simulation through the GUI.
5. Don’t worry about physics. Assume cars will stop on a dime for red lights, and continue through

yellow lights and green lights.

1. Document all assumptions and limitations of your simulation.

**UML Diagram:**

**Chart

Description automatically generated with medium confidence**

**Classes:**

* CMSC335Project3
* Car
* Intersection
* Time

**Variables:**

* Class CMSC335Project3
* setButtons
* JButton
* Car
* JTable
* JLabel
* CarSlider
* carArray
* Intersection
* Class Car
* xPosition
* yPosition
* atLight
* isRunning
* speed
* Class Intersection
* JLabel
* currentLight
* isRunning
* Suspended
* threadName
* Green, Yellow, Red
* Class Time
* isRunning
* timerPattern
* timeFormat
* date
* timeText

**Methods:**

* CMSC335Project3
* JLabel timeText, JLabel trafficAtext, JLabel trafficBtext, JLabel trafficCtext, JButton start, JButton pause, JButton stop, JSlider car1Slider, JSlider car2Slider, JSlider car3Slider, JSlider car4Slider, Intersection A, Intersection B, Intersection C, isRunning = Thread.currentThread().isAlive(), setSize, setVisible, setLocationRelativeTo, setDefaultCloseOperation, simIsRunning.set(false),
* Car
* xPosition, yPosition, isRunning = new AtomicBoolean(false), atLight = new AtomicBoolean(false), suspended = new AtomicBoolean(false), threadName, thread, atLight.get(),
* Timer
* this.isRunning = Thread.currentThread(), String timePattern = "hh:mm:ss a"; , getTime(), timeFormat.format(date), CMSC335Project3.timeText.setText(getTime())
* Intersection
* isRunning.set(true), currentLight = COLORS[i], isRunning = new AtomicBoolean(false), suspended = new AtomicBoolean(false), getColor(), thread = new Thread(this, threadName), suspended.set(true)

**User’s Guide**

1. All of my work is saved in a file pull up the file “CMSC335\_Project2
2. Make sure to have Netbeans application preferer Apaache Netbeans 11.2.pkg/ 12.0.pkg if you are using other application just pull up the classes.
3. Open src 🡪 Main 🡪 Java 🡪 Com 🡪 Mycompany 🡪 CMSC335\_Project3. To see all classes.
4. After opening the Application open the project “CMSC335\_project3 and run the file. It should be able run compile the classes and get you the result desire.
5. Make sure to select the x-Position, y-Position and speed you desire so you could get the distance where the car stop you want. I added where the car stops on what light and the distance.

**Compiled:**

**Graphical user interface, text, application

Description automatically generated**

**Test case Input/ Images:**

**Test case 1:**

Current time: 05:59:13 PM

Intersection A: Green Intersection B: Green Intersection C: Green

|  |  |  |  |
| --- | --- | --- | --- |
| Car | X-Pos | Y-Pos | Speed km/h |
| Car 1 | 294 | 0 | 180 km/h |
| Car 2 | 834 | 0 | 180 km/h |
| Car 3 | 1918 | 0 | 180 km/h |
| Car 4 | 1333 | 0 | 180 km/h |

**Image :**

Graphical user interface, text, application

Description automatically generated

**Test case 2:**

Current time: 06:01:34 PM

Intersection A: Yellow Intersection B: Yellow Intersection C: Yellow

|  |  |  |  |
| --- | --- | --- | --- |
| Car | X-Pos | Y-Pos | Speed km/h |
| Car 1 | 889 | 0 | 180 km/h |
| Car 2 | 1429 | 0 | 180 km/h |
| Car 3 | 2513 | 0 | 180 km/h |
| Car 4 | 1928 | 0 | 180 km/h |

**Image:**

Graphical user interface, text, application

Description automatically generated

**Test case 3:**

Current time: 06:02:44 PM

Intersection A: Red Intersection B: Red Intersection C: Red

|  |  |  |  |
| --- | --- | --- | --- |
| Car | X-Pos | Y-Pos | Speed km/h |
| Car 1 | 1214 | 0 | 180 km/h |
| Car 2 | 1679 | 0 | 180 km/h |
| Car 3 | 2763 | 0 | 180 km/h |
| Car 4 | 2253 | 0 | 180 km/h |

**Image:**

**Graphical user interface, text, application

Description automatically generated**

**Test case 4:**

Current time: 06:06:11 PM

Intersection A: Red Intersection B: Red Intersection C: Red

|  |  |  |  |
| --- | --- | --- | --- |
| Car | X-Pos | Y-Pos | Speed km/h |
| Car 1 | 1214 | 0 | 180 km/h |
| Car 2 | 13 | 0 | 180 km/h |
| Car 3 | 2 | 0 | 180 km/h |
| Car 4 | 2253 | 0 | 180 km/h |

Output: System error because I made changes on Car 2 & 3.

**Image:**

**Graphical user interface

Description automatically generated**

**Lesson Learned:**

This was a challenging assignment for me. It took a lot longer than I would have liked due to numerous bugs and also because this was my first time implementing multiple threads. It didn’t help that I was also incredibly busy with my job at the time. I eventually was able to create something that I am proud of, but I still fell short of the expectations of this assignment. There is no synchronization of keys used between the job threads. In its place, are numerous logical conditions to determine when a new thread should be initialized and when it should sleep. I will go back and properly implement synchronization for the final project.

Overall, I like what I did with the GUI. I think my programs are starting to resemble something someone might actually make for professional use. It was very hard and time consuming to figure out how to deal with this project and create the wanted result. All in all, I feel like this assignment taught me a lot about how multithreading works.