Matthew Stockert & Dominic Retli

April 5, 2018

CIS 450-001

Professor Wang

P3

Table of Contents

[Files Submitted 2](#_Toc510711945)

[Implementation 2](#_Toc510711946)

[tc.java 2](#_Toc510711947)

[Car.java 2](#_Toc510711948)

[Point.java 2](#_Toc510711949)

[Points.java 2](#_Toc510711950)

[Directions.java 2](#_Toc510711951)

[Report 2](#_Toc510711952)

[Screenshots 2](#_Toc510711953)

[Report 3](#_Toc510711954)

[Diagrams 3](#_Toc510711955)

[Contribution 3](#_Toc510711956)

# Files Submitted

P3\_450 – This report

tc.java – Main class

Car.java – Car object and thread for car

Point.java – Each Point on the intersection is an instance of this class

Points.java – A collection of Point.java

Directions.java – Each car will create an instance this class

# Implementation

## tc.java

* This is our main class. This class will create the Point objects as well as the separate threads for cars.
  + Main()
    - This method calls a method that creates Points representing the intersection.
    - This method calls a method that creates the threads representing each car.

## Car.java

* This class represents a car, as well as a thread, combined in one.
  + Car()
    - This is the constructor. It sets the Directions object for the car (this contains the original direction and the target direction).
    - This method will also sleep for the appropriate time given in the assignment details in order to create the different arrival times of each car.
  + run()
    - this method overrides the run() method of the Thread class. Car extends the thread class. In this method, the three methods described in the assignment are called: (arrive, cross, and exit)
  + ArriveIntersection()
    - This method is called to simulate the arrival of the car at the intersection.
    - The needed Point objects (semaphores) needed to complete the desired turn are obtained. From here, the first need point (representing a stop sign) is acquired. After that, another semaphore called “isNext” is acquired. This isNext semaphore is used to ensure that cars turn in a first come first server pattern, as described in the assignment details.
    - The remaining needed points are looped through and acquired.
  + CrossIntersection()
    - This method is called to simulate the crossing of the car at the intersection.
    - First, the appropriate stop sign semaphore is released.
    - Next, the thread will sleep for the appropriate time depending on the turn type (left, right, or straight turns have different times given in the assignment details).
  + ExitIntersection()
    - This method is used to release all of the semaphores, one by one, that were acquired by the Car as the turn is being complete. We do this one by one so that another car, going the same direction, does not have to wait until the car in front of it has completed its entire turn (as requested by Prof. Wang).
  + Print()
    - This method prints the details of each car action (arrival, cross, exit) with the appropriate time, car id, original direction, target direction, and action performed.

## Point.java

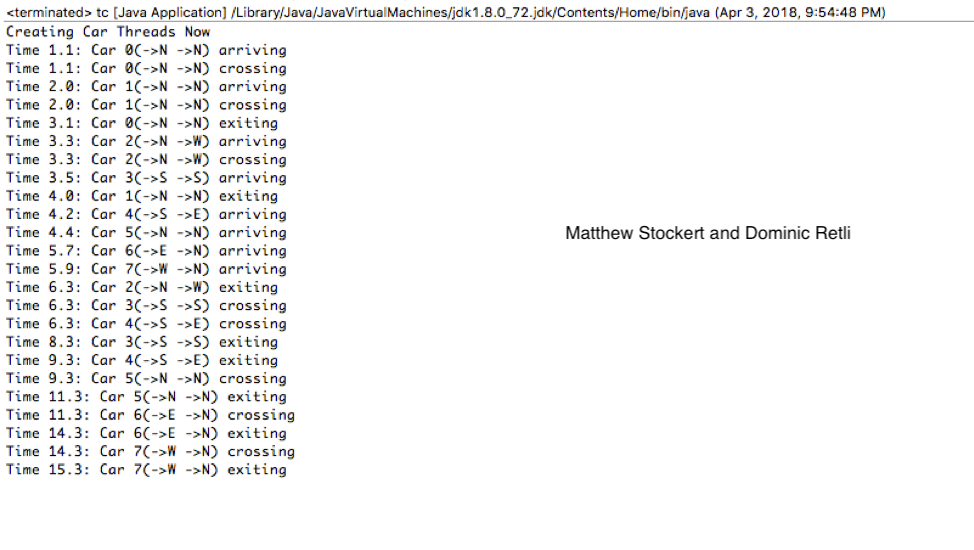
* This class is essentially a wrapper for the java Semaphore class
  + The Point class contains a Semaphore as a data member.
  + Each point was assigned an index1 and index2, in order to match with our diagram attached to the bottom of this report (See image of intersection for more details).
  + acquireLock()
    - This method will acquire the semaphore and stores the Direction object and cid of the Car obtaining the lock.
  + releaseLock()
    - This method will release the semaphore of the Point class.

## Points.java

## Directions.java

# Report

## Screenshots



## Report

## Diagrams

# Contribution