REPORT

SHYAM PATEL-1741030

MEET MODI-1741071

JAY SHAH-1741084

GROUP NNUMBER:13

**TRAFFIC SIMULATION**

**PROBLEM STATEMENT:**

Suppose in a junction point three roads meet, for the traffics on the three roads, three signals P, Q and R are there. Only one signal can be turned on at a time to allow the traffic to pass. Following three functions are suggested in order to study the performance of the traffic control:

1. Each signal when turned on allows passing equal number of traffic, say, N in succession.
2. Different signals allow different number of traffics during their turn, say, signals, P, Q and R allow Q1, Q2 and Q3 number of traffics, respectively.
3. Only that signal will be turned on where maximum numbers of traffics are waiting till the numbers of traffics are at par with the other remaining queues. If all the queues contain the same number of traffic, signal P will be turned on.

**PROJECT BRIEF**

The system works as mentioned below, but for a junction with ‘4’ roads. Vehicles are entered manually and each lane is allocated time on the basis of the algorithm chosen in which ‘m’ number of vehicles can be freed, like a typical traffic signal. Also property can be assigned to the queue manually.

In case of an emergency vehicle, the program will search for the lane with the nearest emergency vehicle and dequeue the whole lane as soon as the emergency vehicle is detected so that the emergency can be helped. Until, all emergencies are taken care of, the signal will not resume the normal state.

The traffic lanes are represented by queues. Linear queue data structure is used for this purpose. Input for each lane would be given sequentially. After all the lanes are initialized, first a check will be performed for any emergency vehicles in all lanes. The emergency vehicles will be allowed to exit the lanes based on their proximity to the lane exit. First, it will detect the foremost emergency vehicle in all lanes and would allow the nearest vehicle to exit first. All lanes would be stopped after the first emergency vehicle exits, then it would detect the foremost emergency vehicle again in all the lanes. This process would continue till all the emergency vehicles clear out of all the lanes.

After all the emergency vehicles exit all the lanes, we check all the lanes based on what algorithm is used. If the number of vehicles in any lane follows below the minimum threshold limit (here if the ratio is >= 4); that lane will be cleared and therefore skipped in the round robin scheduling.

After the lanes are checked for threshold, the round robin scheduling (Modified) will be executed. Round robin scheduling algorithm will take all lanes as an input. The algorithm will allow vehicles in a lane to be cleared till a certain specified time (that is taken as input), after that the lane would be stopped and the vehicles in the next lane would be allowed to clear for the mentioned specific time (Time Limit). This process will continue till all the vehicles from all the lanes would be cleared and all the lanes are empty.

Priority Scheduling gives better performance, since the emergency can be taken care of with priorities and we can also include functionality for dequeuing a specific queue out of sequence. Priority Scheduling was taken into consideration during the course of the project but due to some conflicts with the timers, the class was scraped off from the final project.

**LIST OF DATA STRUCTURES USED:**

-Queue data structure

-Queue with array

**LIST OF ALGORITHMS USED:**

-Round Robin

-Time Scheduling Algorithm

-Shortest Job First

-Vehicle Based

**LIST OF PROGRAM FILES:**

-Main.java

-VehicleBased.java

-Queue.java

-TimeBased.java

-ShortestJob.java

-ShortestJobBased.java

**List of Concepts used:**

● Queue handling and queue operations.

● Round-Robin Scheduling Algorithm

● Timer class

● Basic Object-Oriented Programming with optimization

● Java Constructors and Methods

● MVC (Model, View, Control) Framework

● Exception Handling

● Multi-threading and working with Threads

**Algorithms/Psuedocode:**

**Psuedocode:**

Main class

Print(

1.Vehicle Based.

2.Time Based.

3.Shortest Job.

4.Exit.

Enter your choice.)

Scan choice;

Switch choice:

Case 1: Call vehicle based (Implements Modified Round Robin)

Break;

Case 2: Call Timer Based (Implements Modified Round Robin with Timer class)

Break;

Case 3: Call Shortest Job Based (Implements Shortest Job Based )

Break;

Case 4: Break;

End

**Algorithms:**

**ShortestJobBased Scheduling(Modified):**

Let queue[] be array of 4 queues

//Length of these queues will the number of vehicles present in that queue

//Now we will check if the shortest job exists or not

while(queue[0].VehiclesLeft() != 0 || queue[1].VehiclesLeft() != 0 || queue[2].VehiclesLeft() != 0 || queue[3].VehiclesLeft() != 0)

Call Checkshortest Job

For i=0 to 3,j=0 to 3

If queue[j] >0

If queue[i]/queue[j] >= 4

Return queue[j];

End if

Else

Return array[] ; // all the elements of array will be -1

If shortest job exists dequeue queue[j] first

Else dequeue all the queue using round robin

End while

Round Robin Scheduling:

ROUND-ROBIN​​(time)

Let A[1 ---- n] be the waiting queue

let b[1------n] be the arrival time of the function in ascending order

let c[1------n] be the running time of the function

T is the burst time of the process

time =0;

i=1; j=1; runtime = 0;

while A is not empty

if b[i] == time

enqueue function b[i] into waiting queue

i = i+1;

if c[j] == 0

output b[j] runtime= 0;

dequeue A for next function

if(runtime == T)

enqueue b[j] to A runtime = 0;

dequeue A for next function c[j] = c[j] -1;

time = time +1;

end while

**Input Data:**

Number of Sides:4

Limit of Vehicles if Vehicle Based/Time if Timed Based: Limit

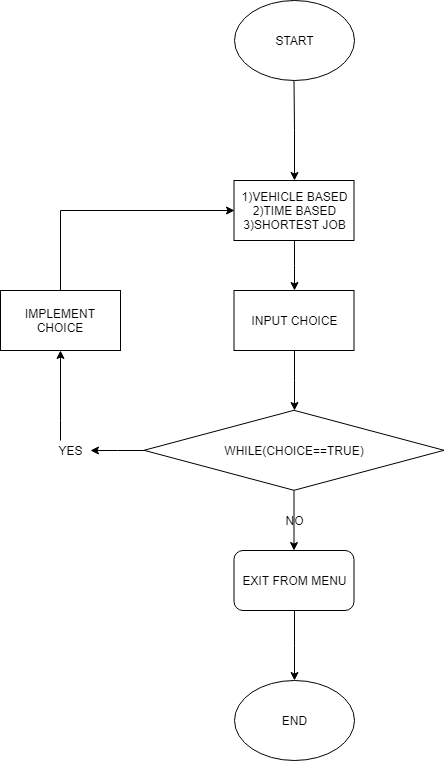
Enter Emergency Vehicles and in which lane : Represented by 0

**Output:**

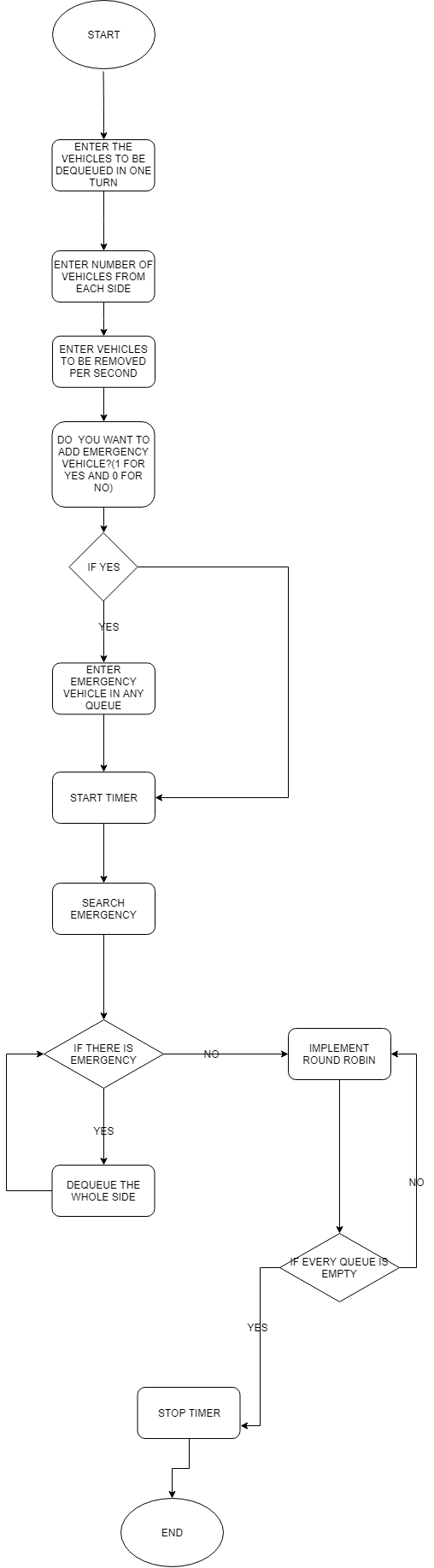
How the Queue is dequeued and Time taken…

**Flowchart:**

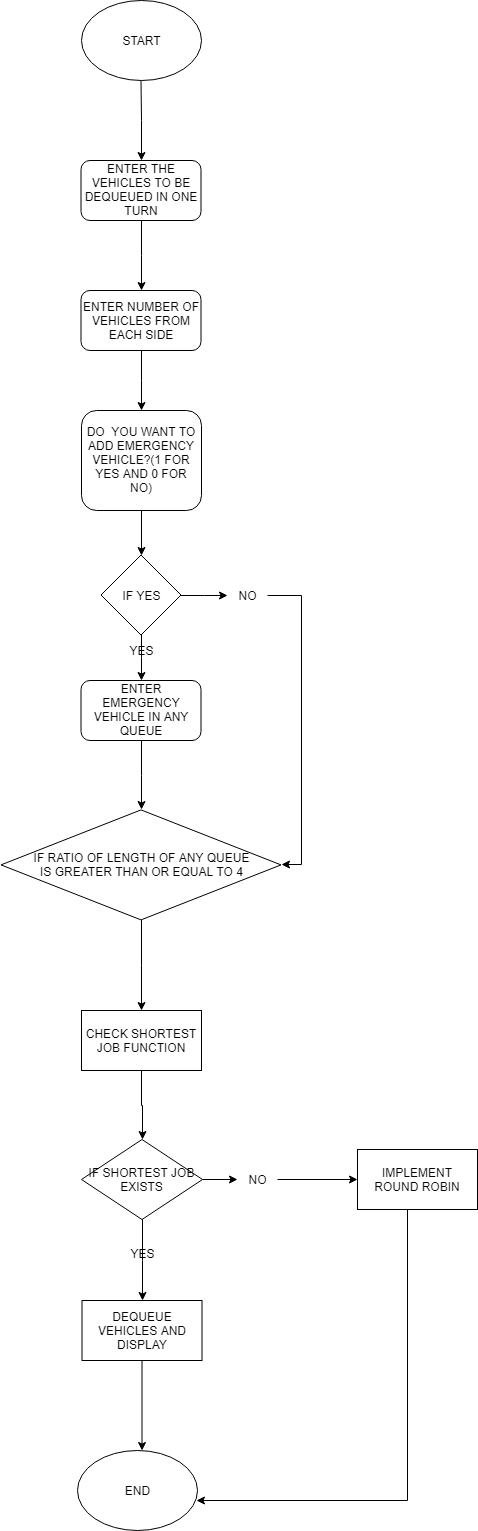
**Main class:**



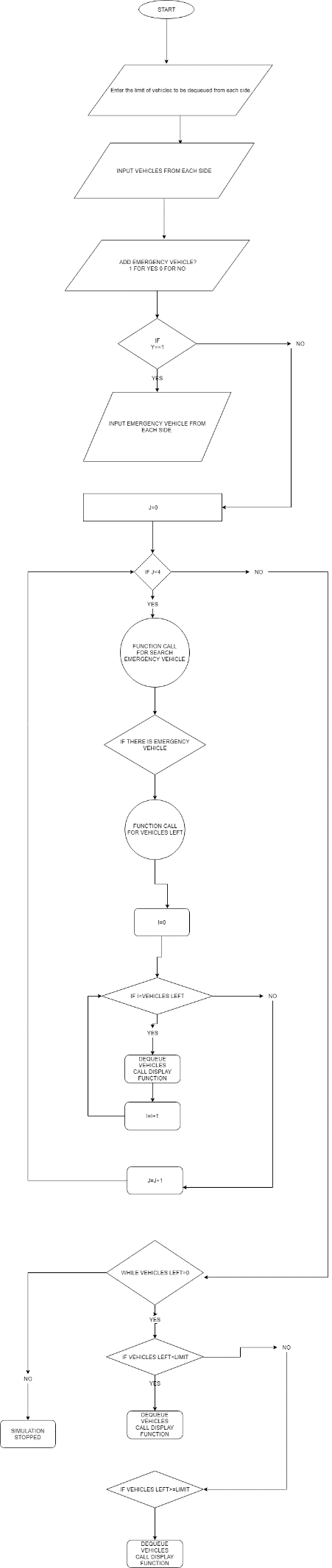
**Timer Based:**



**Shortest Job First:**



**Vehicle Based:**



**References:**

● “Round Robin Scheduling (RRS)”, technopedia.com

● “Operating System Scheduling algorithms”, tutorialspoint.com

● “Timer Class in Java”, journaldev.com