Design & Analysis of Algorithms

CS5592 SPRING2018

PROJECT REPORT

Emergency Vehicle Dispatch System

Instructor: M. Amin Kuhail, PhD.

Email: kuhailm@umkc.edu

Department of Computer Science & Electrical Engineering

University of Missouri, Kansas City

TEAM MEMBERS:

MAHESH JAMALLAMUDI (16234558)

GitHub URL:

https://github.com/mahesh9595/DA_Final_project

PROJECT DESCRIPTION

Project 2 – Emergency Vehicle Dispatching System

- We are trying to design an emergency vehicle dispatching system. Let's assume that there are three different types of emergency vehicles: 1 (Ambulance), 2 (Fire Truck), 3 (Police car).
- Let's also assume that every request only needs one emergency vehicle.
- Here's an example of the table of EmergencyVehicles:

ID	Type	ZipCode
1	1	64151
2	1	64151
3	1	64151
4	2	64151
5	3	64151
6	3	64151
7	3	64149
…	…	…

(e.g. from the table above, we can see there are 3 ambulances, 1 fire truck, and 1 police car at 64151.

- Here's an example of the table Request

ID	VehicleType	ZipCode	VehicleID
1	1	64151	?
2	2	64149	?
3	1	51234	?
…	…	…	?

(e.g. from the table above, we can see there is a request for an ambulance at zip code 64151. The question mark means we still haven't assigned a vehicle for the request.

- Figure 3 shows the table distance, which consists of information telling us how far apart are neighboring zip codes.

ZipCode1	ZipCode2	Distance
64150	64151	4
64151	64152	2
64152	64153	3
…	…	…

(e.g. from the table above, we can see that the zip code 64150 is 4 miles away from 64151).

Note: the distance between zip codes that are not neighboring is not readily available.

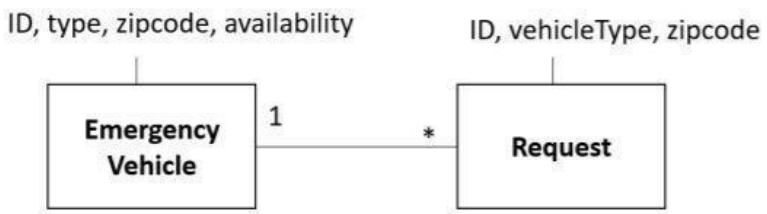


Figure 2

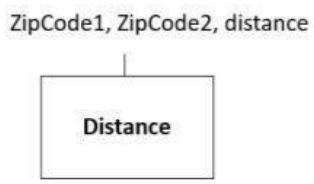


Figure 3

Technical Requirements

- Implement an algorithm that processes requests one by one. For each request, the algorithm should try to find the closest available emergency vehicle
- Implement the algorithm in a language you are comfortable with.
- Make up the data for the project. It should be similar to the examples I gave. You can generate the files at run time, or have it stored in files, and you read it at run time.
- The algorithm should produce the vehicle matching, e.g.:

ID	VehicleType	ZipCode	VehicleID	Distance
1	1	64151	1	2
2	2	64149	15	1
3	1	51234	20	3
...	...	...	...	...

PROJECT APPROACH

Language Used: JAVA

Software: ECLIPSE JAVA OXYGEN

Operating System: Windows

Briefing:

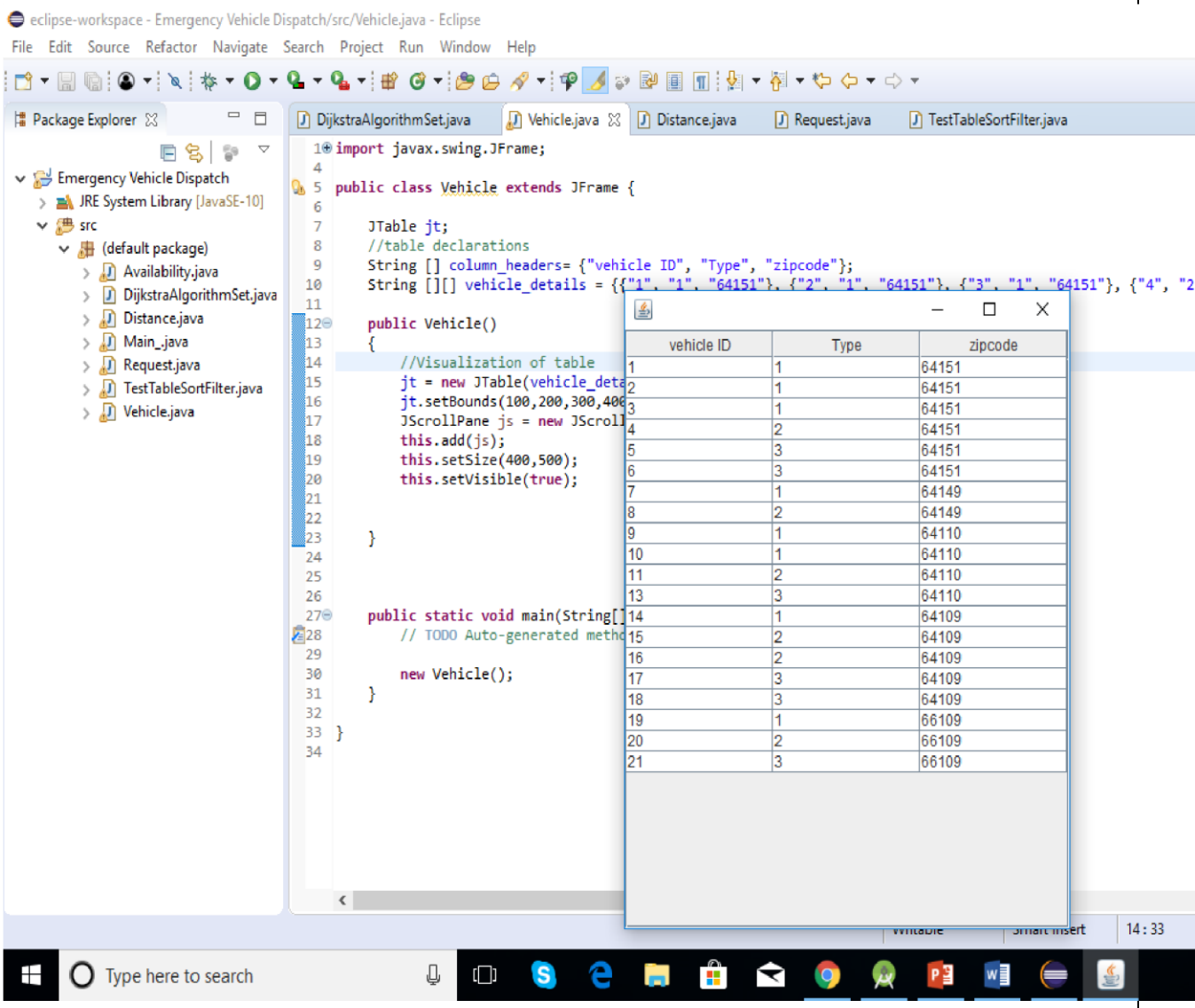
- Main objective of the project is to dispatch an Emergency Vehicle to the required area.
- From the given data we can say that Vehicle type -1 is for Ambulance, 2 is for Fire Truck, and 3 for Police Car.
- So, the data of Vehicles like how many vehicles and what vehicles are present in respective areas are manually inserted using JTables.
- We also make some distance assumptions between the zip codes we have selected, and these are also manually inserted in tables using JTables.
- If there is no vehicle which is requested in respective area, then one has to dispatch from another location which is nearer to requested location.
- For this purpose, we use Dijkstra's algorithm to find the minimum distance.
- After calculating the minimum distance, a vehicle is dispatched from the near location to the requested location based on the availability.

The order I followed in the project is

1. CREATING TABLES:

As mentioned above I created a table for vehicle data which consists of vehicle data like vehicle ID, type and respective zip codes and another table for distances between the zip codes.

Both the table data is based on the assumption of distances, no. of vehicles, etc.…

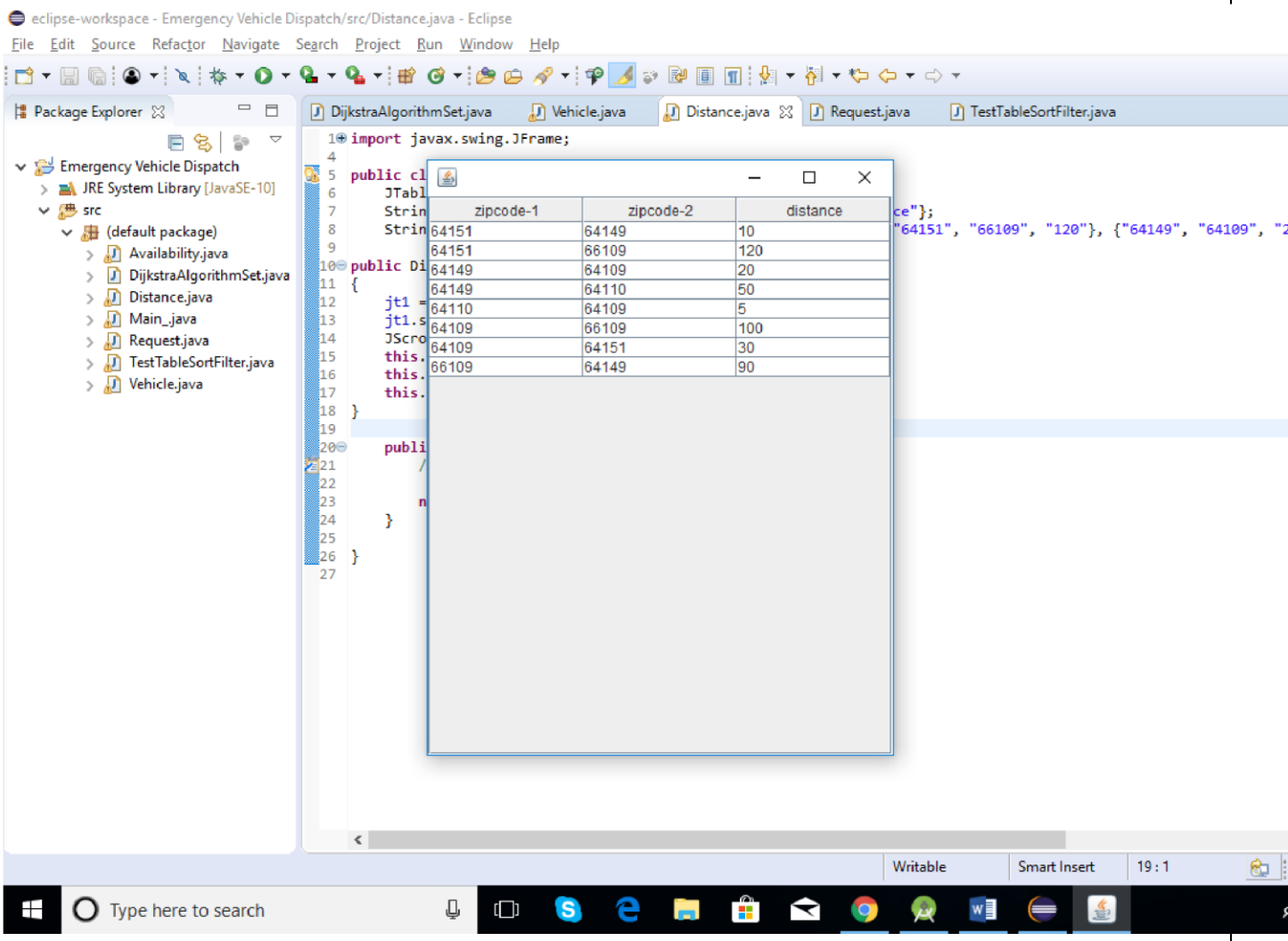


Example:

In zip code 64151 we have 3 Ambulances, 1 Fire Truck, and 2 Police cars. Table 2 is created with the same logic has table one implementing JTable In this table, we have distances between two zip codes.

Example:

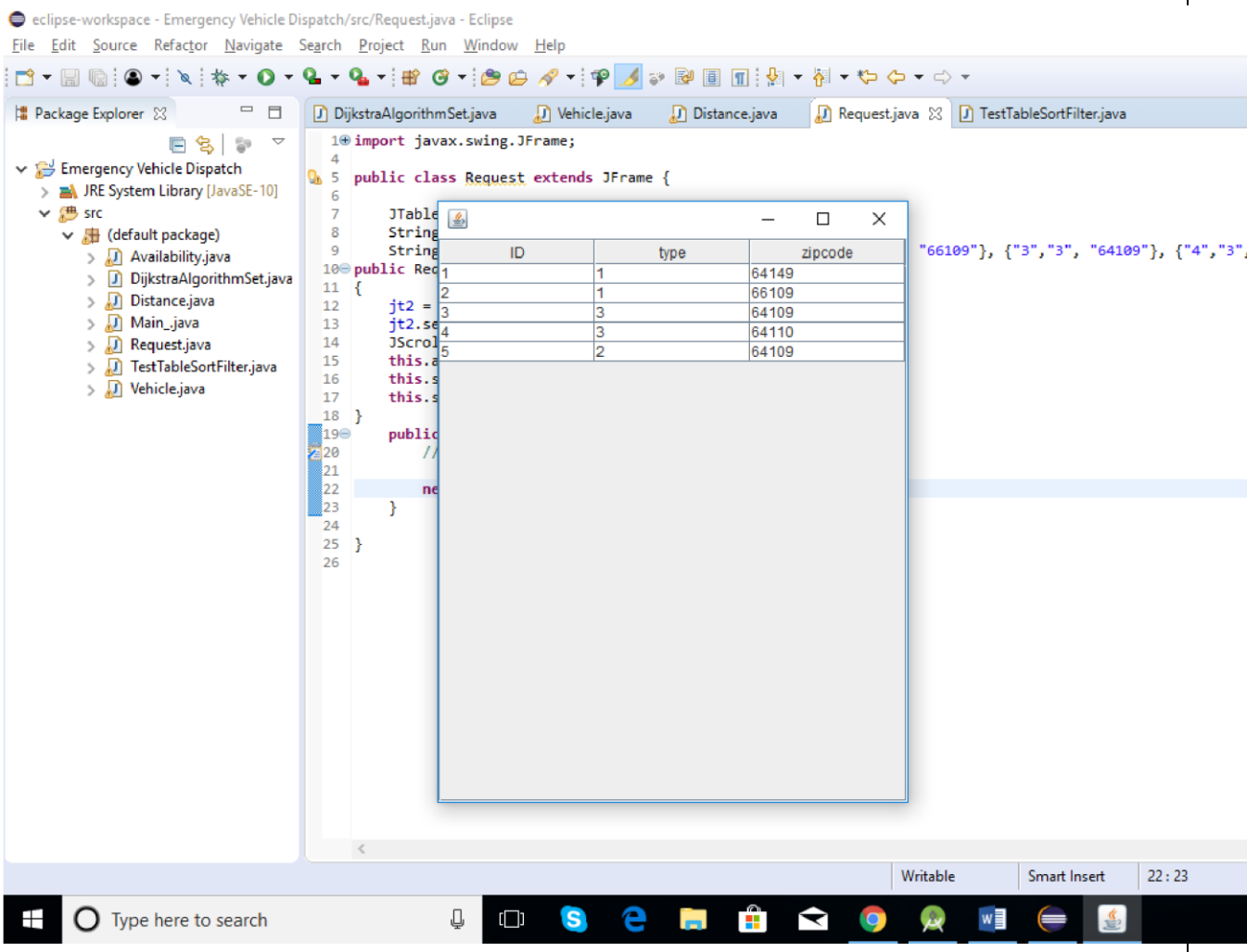
Distance between zipcode1 (64151) and zipcode2 (64149) is 10.



2. Request table:

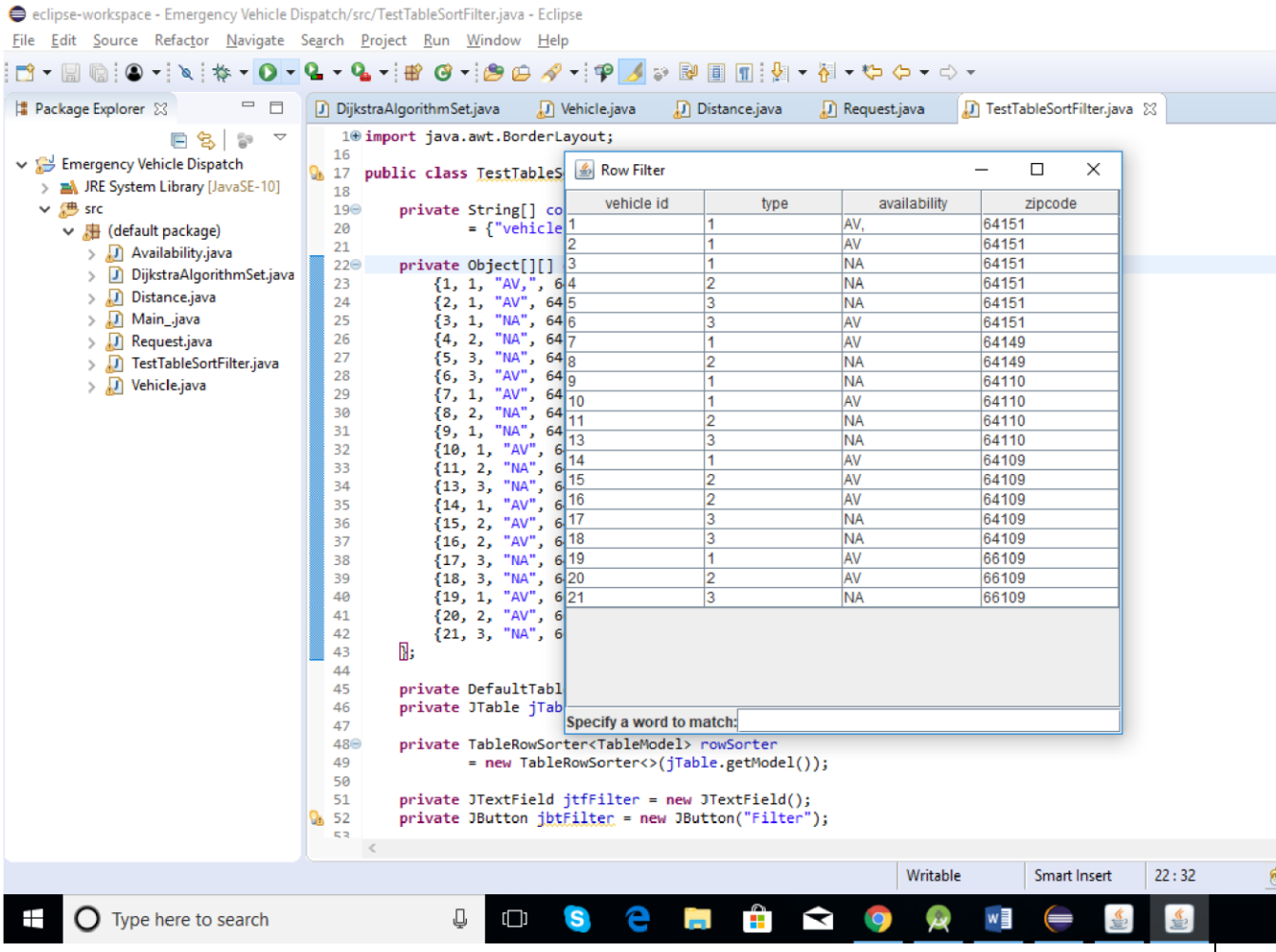
We have another table for the requests in which we can say that a vehicle is needed for a zip code.

For example, there is requirement of type-1 vehicle at 64149 areas.



3. Availability check:

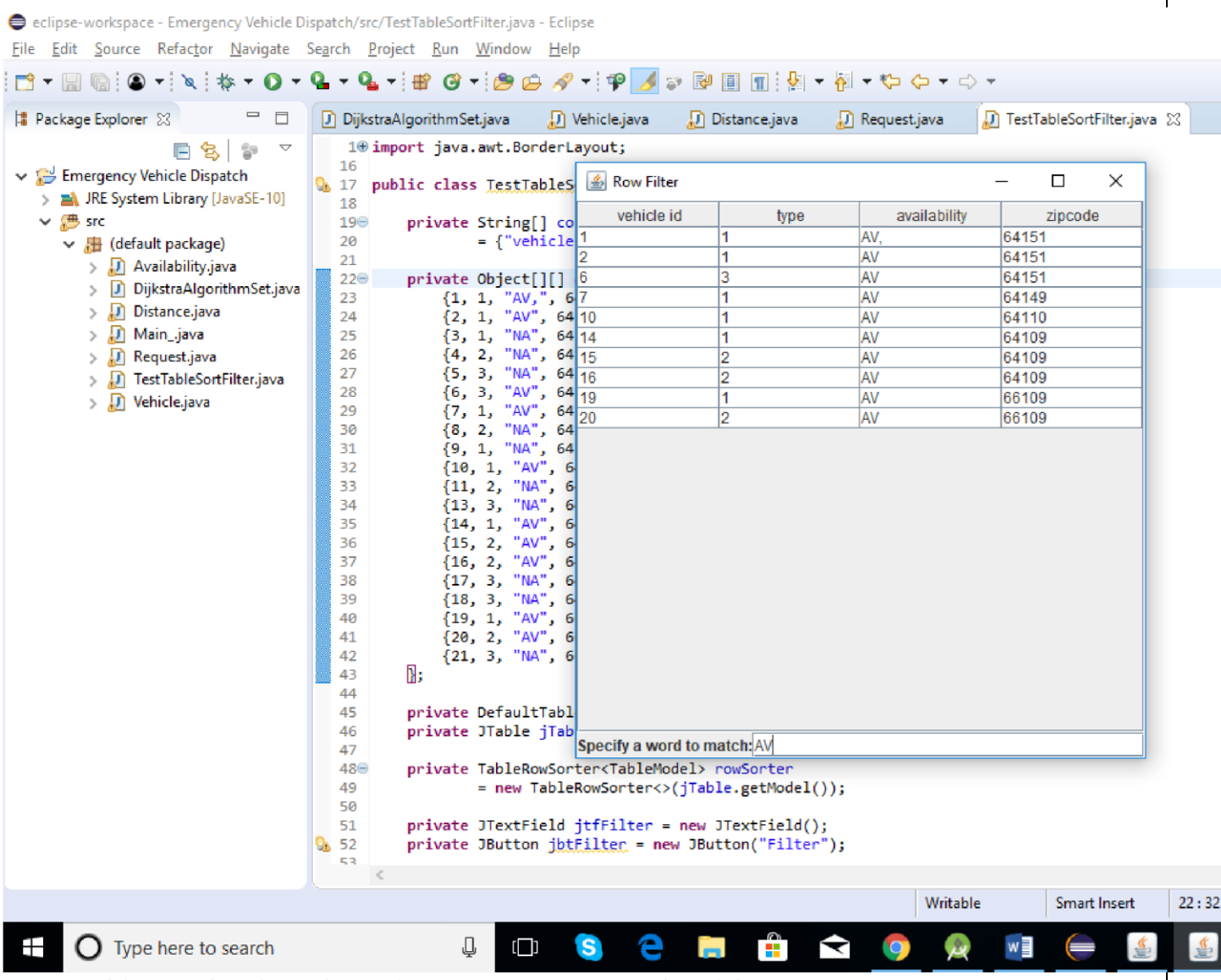
To dispatch an emergency vehicle to a location first one has to check for it's availability. So, I created a table in which we can check for the available vehicles using a search bar.
I assumed of AV for availability and NA for not available.



If we search for AV in search bar its will show the list of availability vehicles.

4. **FINDING SHORTEST PATHS**

If there is no available vehicle for the request in respective location, we need to find a location where vehicle is available which is nearer to the request location.



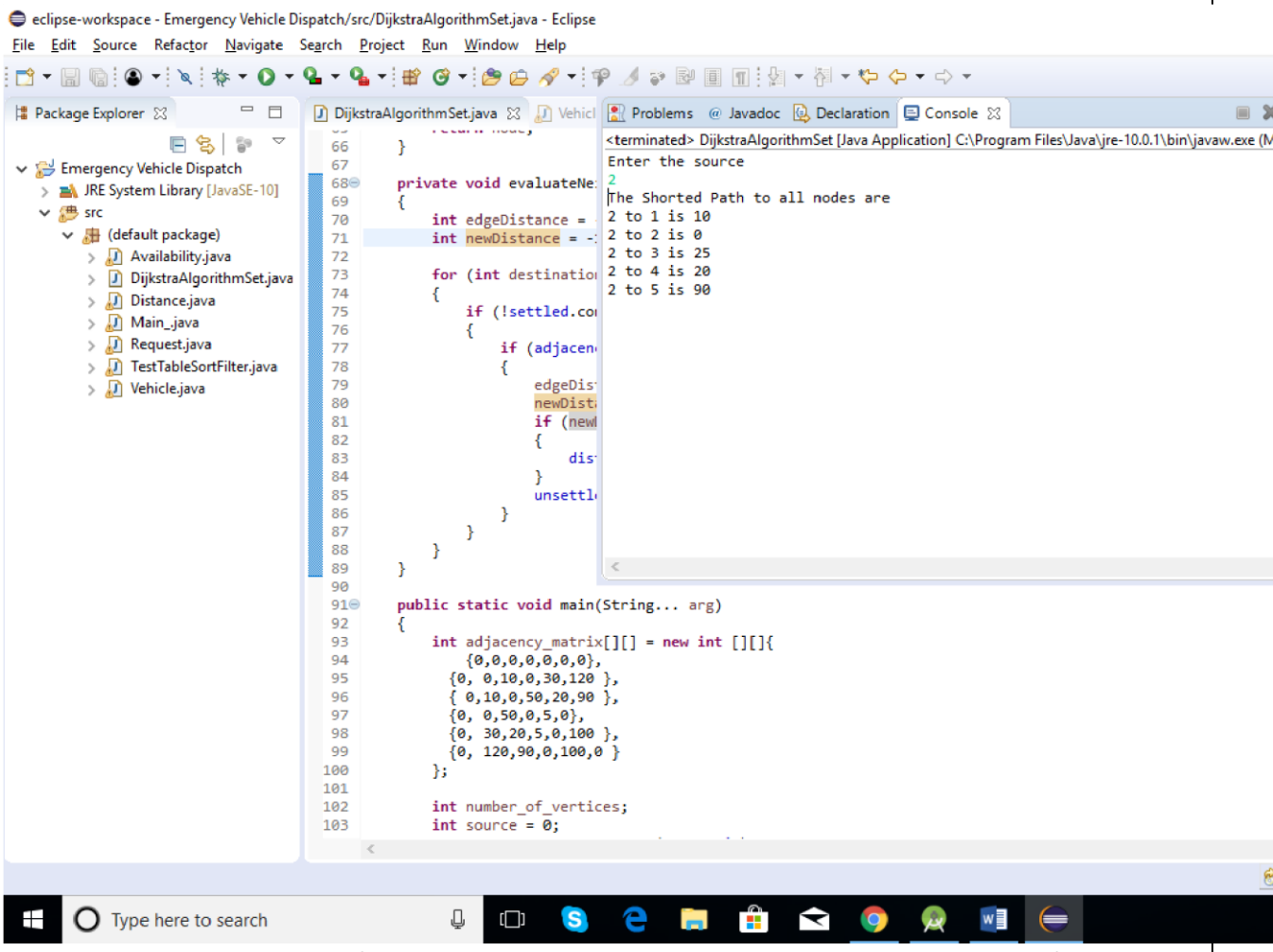
This can be done by using DIJKSTRA'S algorithm.

The output of the DIJKSTRA'S algorithm is it will show you shortest paths from a location to all other locations.
As soon as you run the code it will ask you to enter the source, it means from which location you want to find the shortest path.

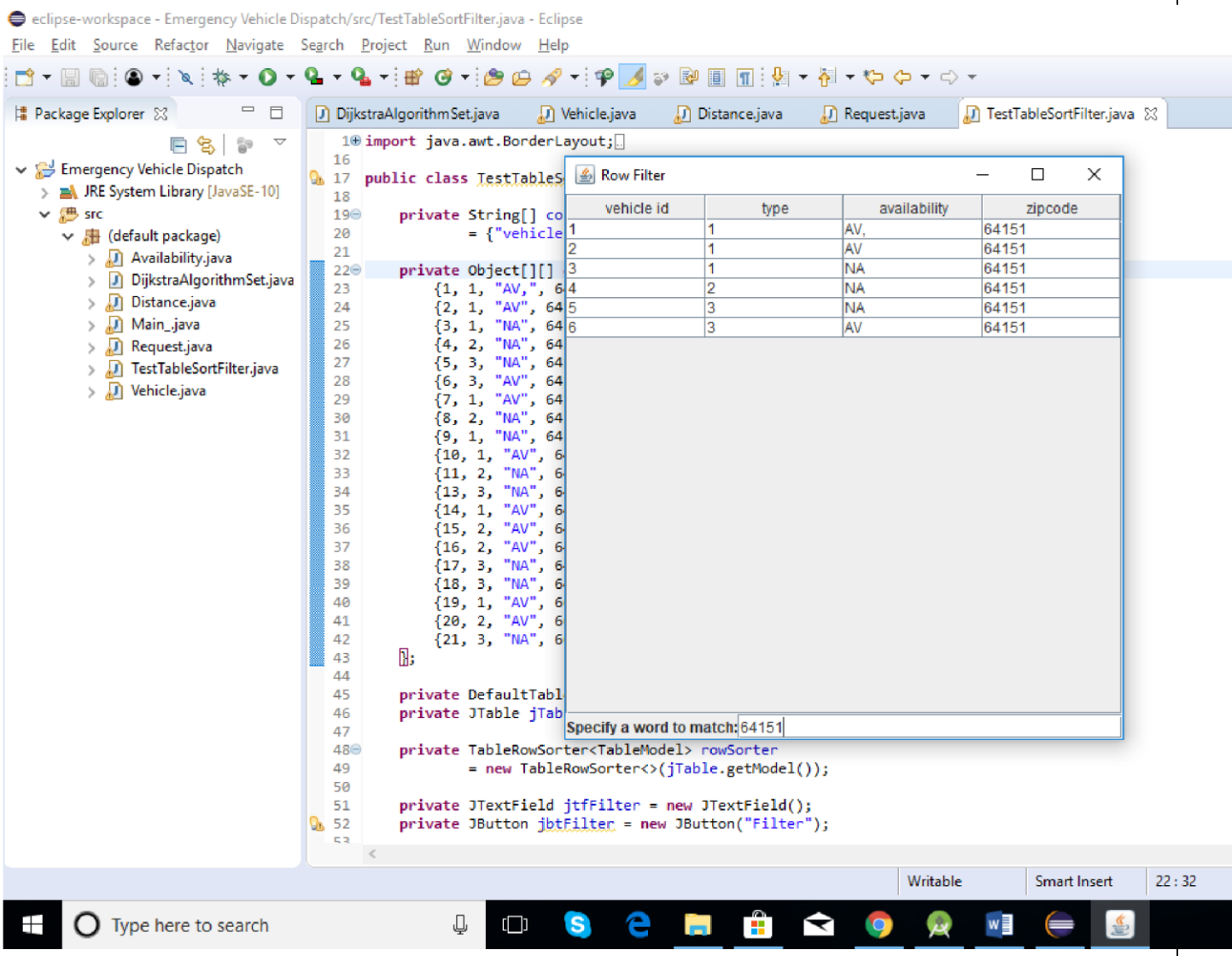
I have assigned some numbers to the zip codes so we deal with numbers instead of zip codes.
64151 as 1,

64149 as 2,
64110 as 3,
64109 as 4, 66109
as 5.

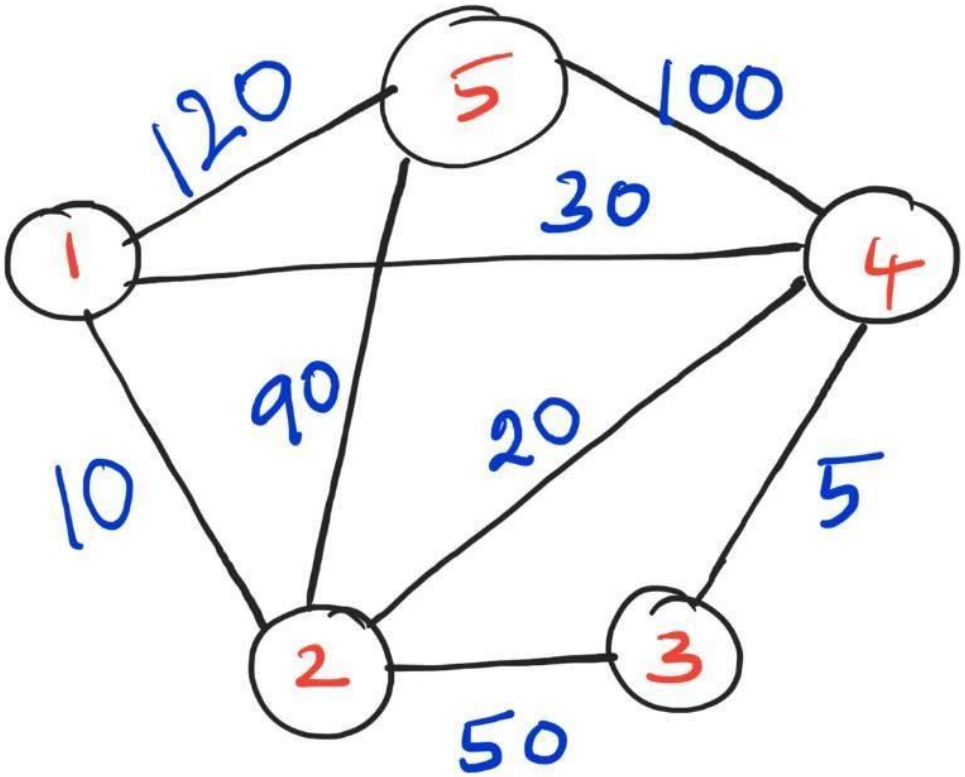
So, after we find the shortest path we take the nearest location and then check for availability from availability table, if we have a vehicle then we can



dispatch that vehicle else we find another nearest location and again check for availability and will dispatch the vehicle.



Model Diagram:



Last modified: 10:02 PM

We have used DIJKSTRA's Algorithm which gives the efficient time complexity of O**(E log V)** where E – No. of Edges

V – No. of Vertices

REFERENCES

https://www.youtube.com/watch?v=EbL1pj3tOgQ https://www.sanfoundry.com/java-program-implement-dijkstras-algorithm-using-queue/
https://stackoverflow.com/questions/22066387/how-to-search-an-element-in-a-jtable-java
https://stackoverflow.com/questions/1107911/how-can-i-filter-rows-in-a-jtable
https://stackoverflow.com/questions/3179136/jtable-how-to-refresh-table-model-afterinsert-delete-or-update-the-data