



Design Analysis & Algorithms

Course # CSEE 5592 0001

Semester: FALL2017

PROJECT REPORT

Emergency Vehicle Dispatch System

Instructor: Mohammed Kuhail, PhD.

Email: kuhailm@umkc.edu

Department of Computer Science & Electrical Engineering

University of Missouri, Kansas City

TEAM MEMBERS:

SATYA SAI DEEPTHI KATTA (sk57f)

16231371

PRANATHI GOPIDI (pg5y7)

16233917

RAKESH REDDY PALLEPATI (rpd54)

16231311

SANDEEP PABOLU (sp4xp)

16230088

AKHILA KATTA (akz6d)

16232635

GitHub URL:

https://github.com/SSDeepthiKatta/DAA_Project_2017

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.

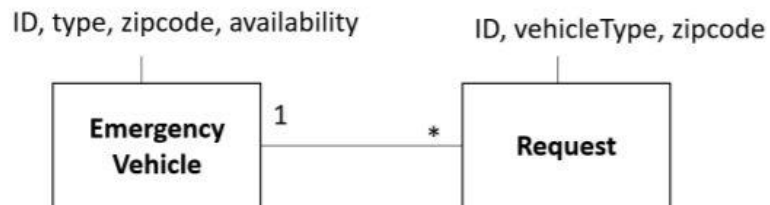


Figure 2

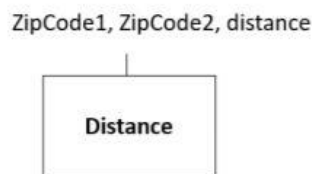


Figure 3

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

Emergency Vehicle Dispatch is the class project implemented under the guidance of Dr. Kuhail to submit for the grading of the course CS 5592(Design Analysis and Algorithms).

Language Used: JAVA

Software: ECLIPSE version JE-Oxygen2

Operating System: Windows, iOS

Project Briefing:

- The project mainly aims to dispatch a emergency vehicle to the desired location inserted.
- The vehicle types given are 1. Ambulance 2. Fire Truck and 3. Police Car.
- For the input like Vehicle ID, Vehicle Type and the Distance between the zip codes are manually inserted and are stored in JTables.
- By taking the current location, minimum distance between the zip codes in the locality are calculated.
- For calculating the minimum distance between the zip codes, famous Dijkstra's algorithm is used and the values are displayed in console.
- Availability of the desired vehicle is searched, and the minimum distance zip codes location dispatches the emergency vehicle.

WORK FLOW

The Following steps are implemented in the project.

Step 1. Creating Tables.

Step 2. Inserting the current zip code.

Step 3. Finding Shortest Paths.

Step 4. Availability Check

Step 5. Dispatch Vehicle Allocation.

Step 1. CREATING TABLES

As described in the project description, we have created two tables.

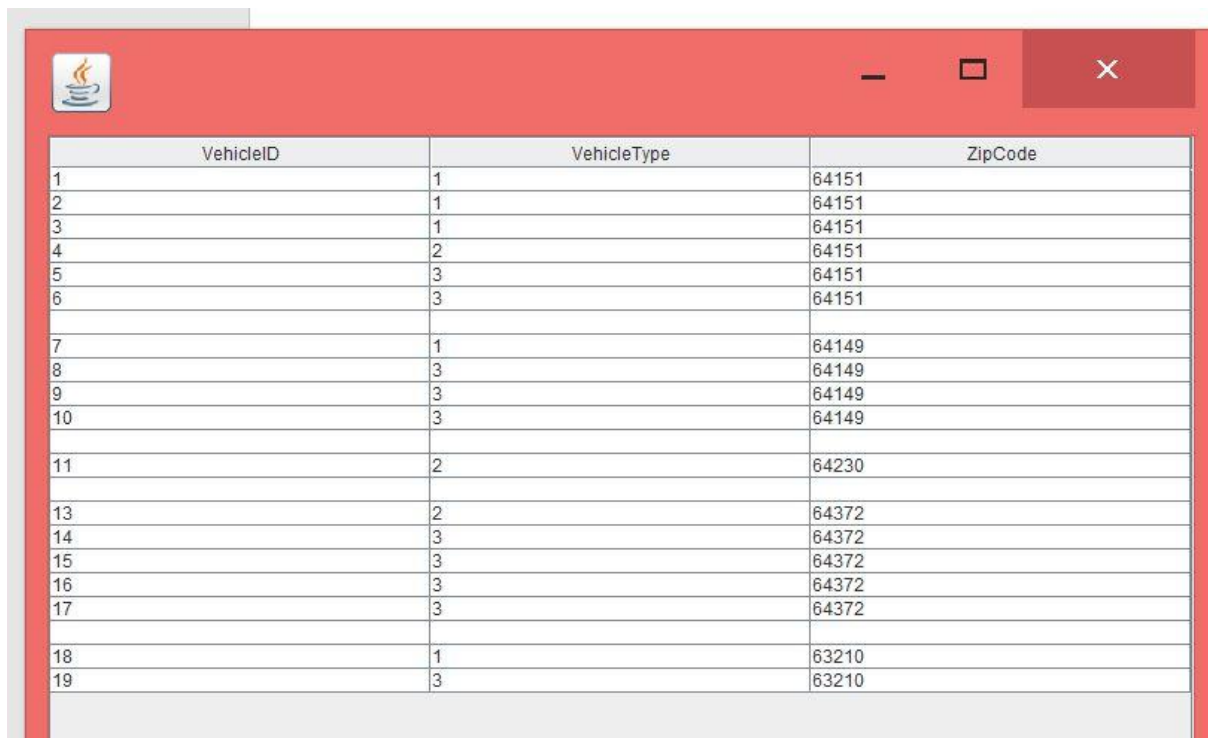
Table 1 gives the Vehicle details which is Vehicle ID, Vehicle Type and its ZIP Code. We have considered 5 zip codes of the locality.

Vehicle ID is given to each vehicle available in the zip code.

Vehicle Types are considered from the project description stating (1) for Ambulance, (2) for Fire Truck and (3) for Police Car.

Example:

In zip code 64179, one ambulance with vehicle ID- 7 and 3 Police Cars with vehicle Id's 8, 9 and 10 are available.



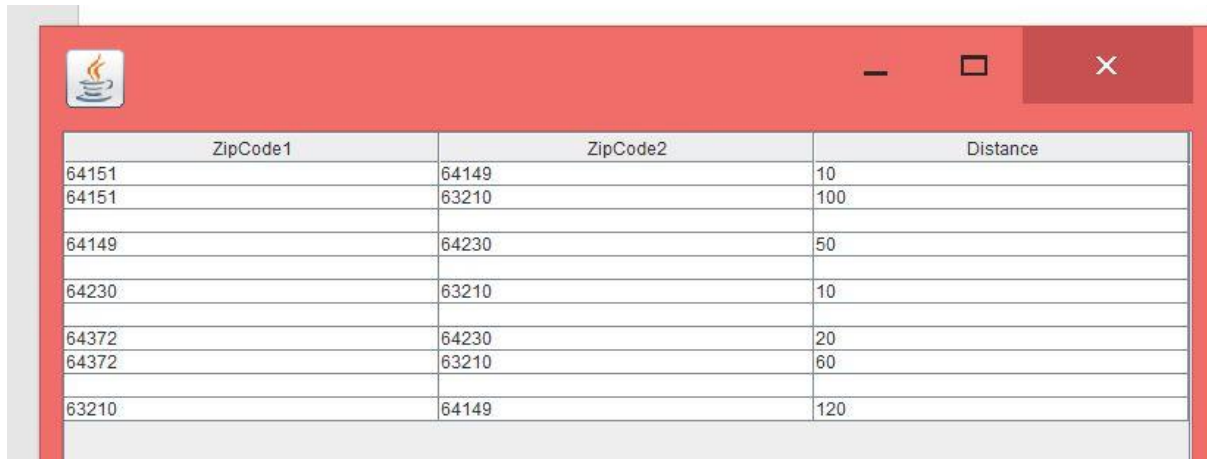
VehicleID	VehicleType	ZipCode
1	1	64151
2	1	64151
3	1	64151
4	2	64151
5	3	64151
6	3	64151
7	1	64149
8	3	64149
9	3	64149
10	3	64149
11	2	64230
13	2	64372
14	3	64372
15	3	64372
16	3	64372
17	3	64372
18	1	63210
19	3	63210

Table 2 is created with the same logic has table one implementing JTable importing JFrame Library.

In this table, we have mainly concentrated in the distance between two zip codes.

Example:

Distance between zipcode1 (64151) and zipcode2 (63210) is 100.

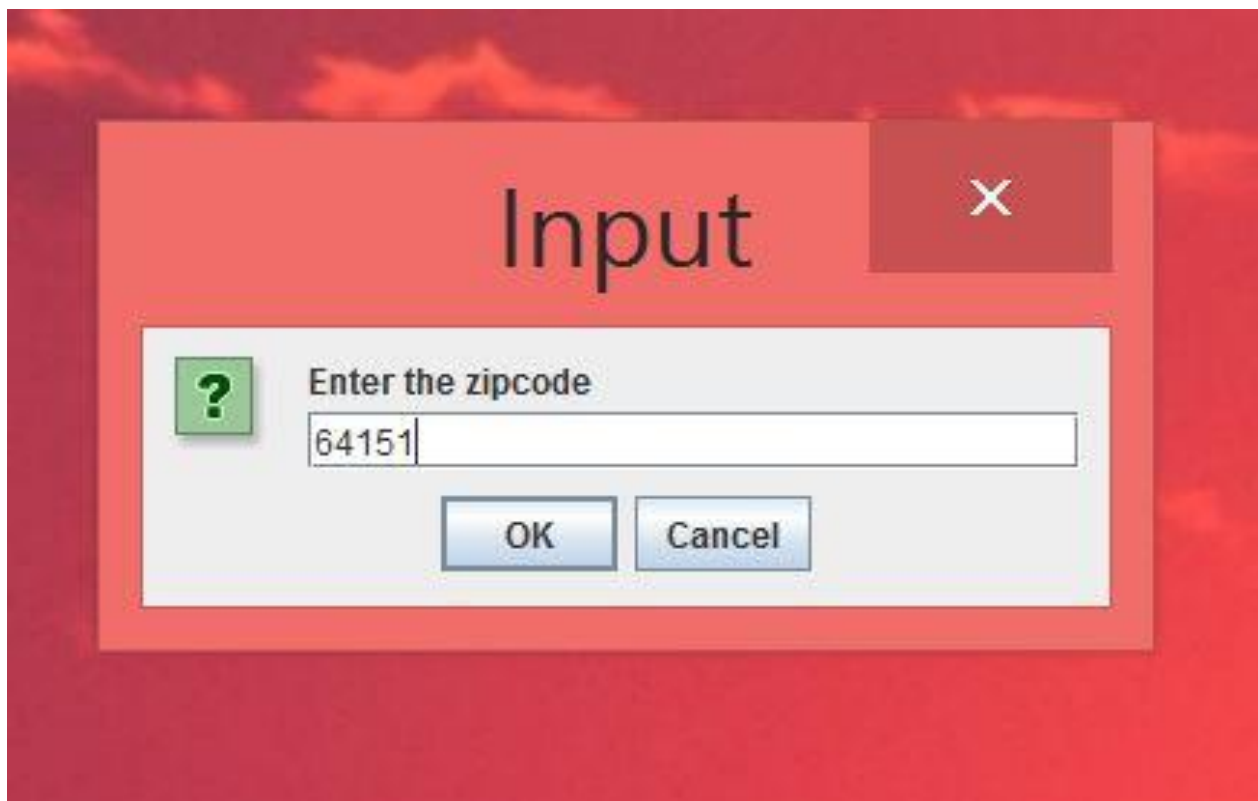


ZipCode1	ZipCode2	Distance
64151	64149	10
64151	63210	100
64149	64230	50
64230	63210	10
64372	64230	20
64372	63210	60
63210	64149	120

Step 2. INPUT ZIPCODE

To extract the input value from the user, we have created a dialogue which questions the user to enter the location (i.e zipcode) for which the emergency vehicle needs to be dispatched.

The dialogue box is created by importing `javax.swing.JOptionPane`. The input is given is stored in the `String`.



Step 3. FINDING SHORTEST PATHS

Now the task to find the shortest paths from remaining zip codes available in the locality to the current location.

In order to implement a efficient algorithm, we have used DIJKSTRA's algorithm which gives the minimum shortest path without leaving any vertex behind. The complexity of Dijkstra's is very less and the accuracy of the algorithm is highly favorable.

We displayed the shortest paths calculated from the given location the all other zip codes available in the locality using Dijkstra's algorithm in the CONSOLE of Java Eclipse Software.

Screenshot is provided below.



```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Dec 4, 2017, 2:19:57 PM)
Given zipcode is : 64151
```

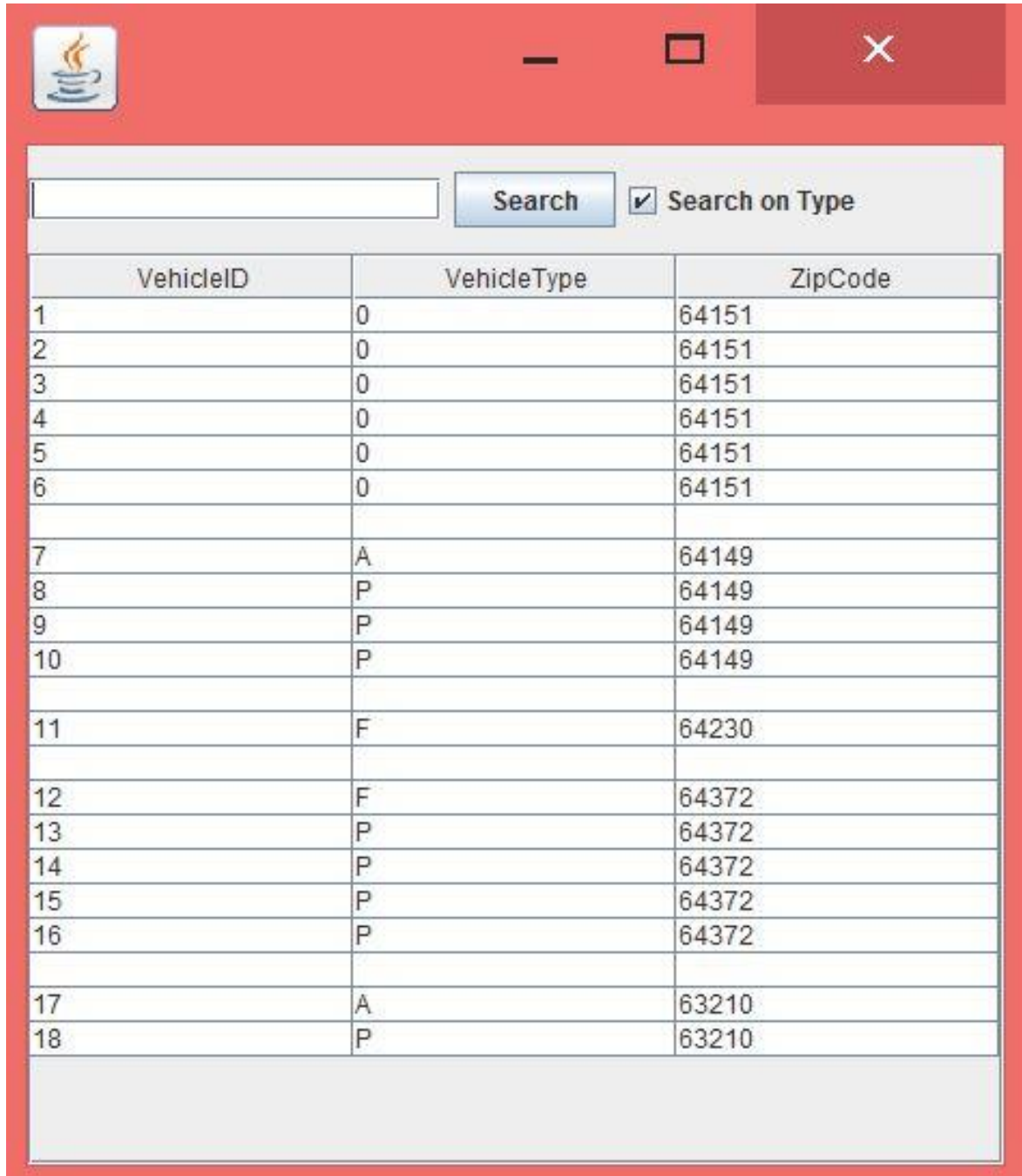
Using Dijstra's Algorithm,
shortest paths from given zipcode to other zipcodes are displayed below

Distance to 64151:	0.0
Distance to 64149:	10.0
Distance to 64230:	30.0
Distance to 64372:	50.0
Distance to 63210:	60.0

Step 4. AVAILABILITY CHECK

The significant part of the project is to check for the emergency vehicles available at the zip codes.

As we have taken the given location as 64151 i.e, no vehicle is available at the location, so it needs to get the vehicle from the shortest distance location available.

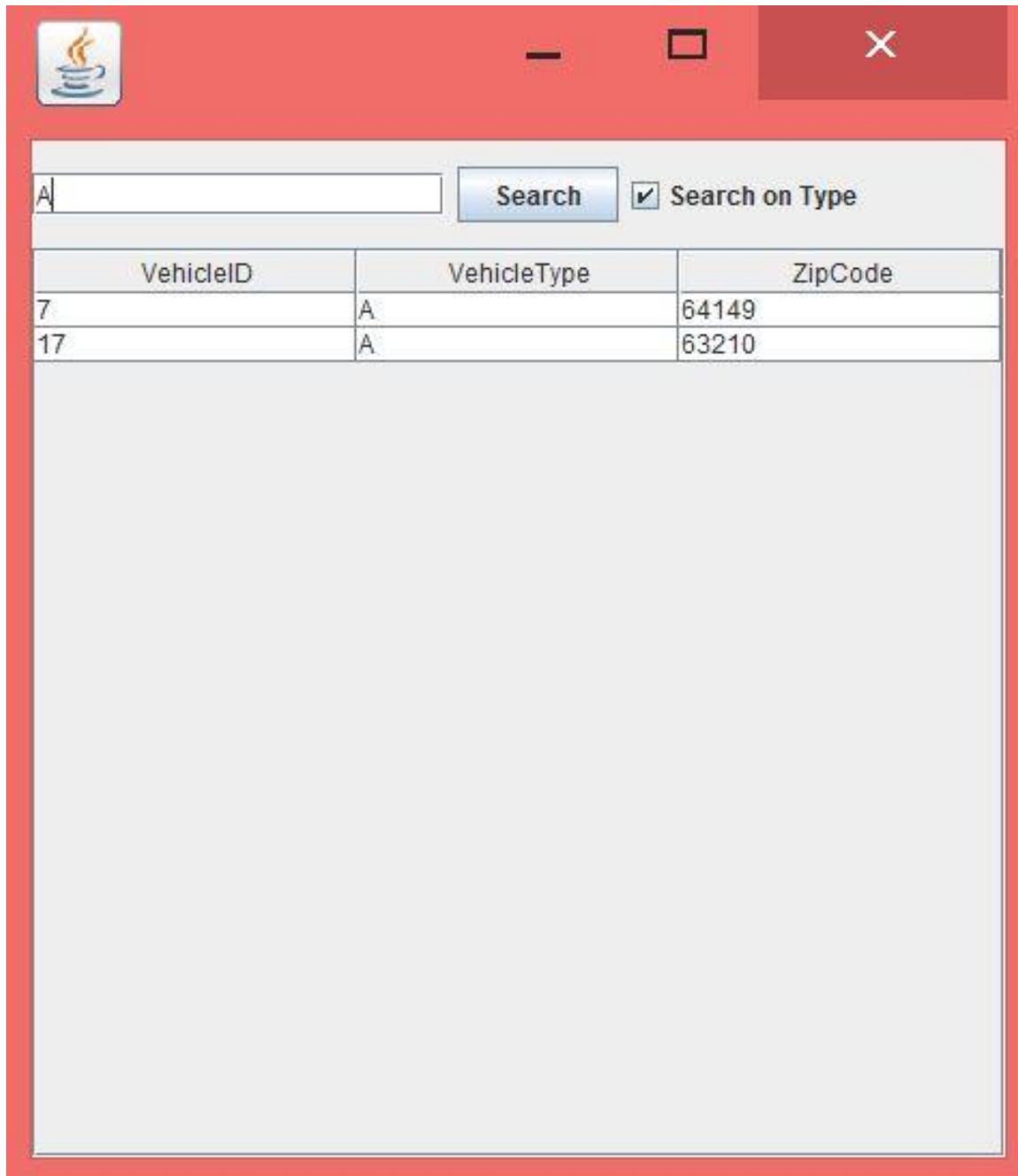


The screenshot shows a Java Swing window with a red title bar. The window contains a search interface with a text input field, a 'Search' button, and a checked checkbox labeled 'Search on Type'. Below the search interface is a table with three columns: 'VehicleID', 'VehicleType', and 'ZipCode'. The table lists 18 vehicles, grouped by zip code: 64151 (vehicles 1-6, type 0), 64149 (vehicles 7-10, types A, P, P, P), 64230 (vehicle 11, type F), 64372 (vehicles 12-16, types F, P, P, P, P), and 63210 (vehicles 17-18, types A, P).

VehicleID	VehicleType	ZipCode
1	0	64151
2	0	64151
3	0	64151
4	0	64151
5	0	64151
6	0	64151
7	A	64149
8	P	64149
9	P	64149
10	P	64149
11	F	64230
12	F	64372
13	P	64372
14	P	64372
15	P	64372
16	P	64372
17	A	63210
18	P	63210

Now, in order to get the availability, we have implemented a search function where the emergency vehicle type (A, F or P) is entered in the Search Dialogue Box.

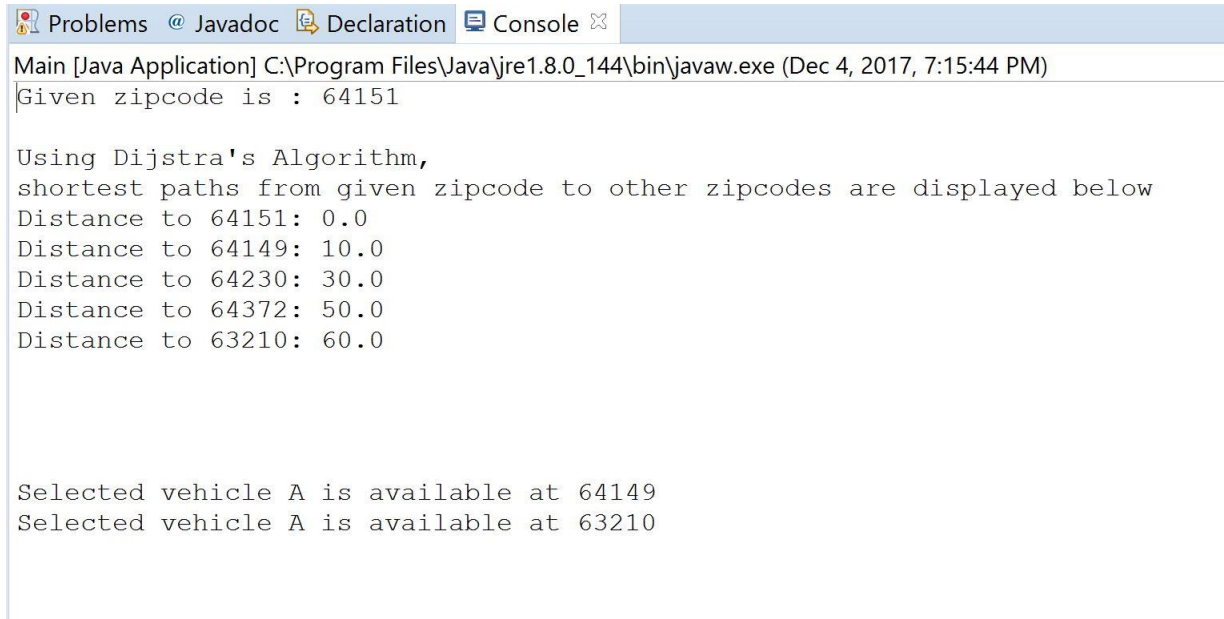
The zip codes are displayed along with the vehicle ID where the entered emergency vehicle is available.



The image shows a software window with a red title bar. Inside, there is a search interface. At the top left is an icon of a coffee cup with a flame. To its right are standard window controls: a minus sign, a maximize button (represented by a square), and a close button (represented by an 'X'). Below the title bar is a search area containing a text input field with the letter 'A' entered, a blue 'Search' button, and a checked checkbox labeled 'Search on Type'. Below the search area is a table with three columns: 'VehicleID', 'VehicleType', and 'ZipCode'. The table contains two rows of data. Below the table is a large, empty light gray rectangular area.

VehicleID	VehicleType	ZipCode
7	A	64149
17	A	63210

We have displayed the results in the Console and also stored the values in the output text file.



```
Problems @ Javadoc Declaration Console
Main [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Dec 4, 2017, 7:15:44 PM)
Given zipcode is : 64151

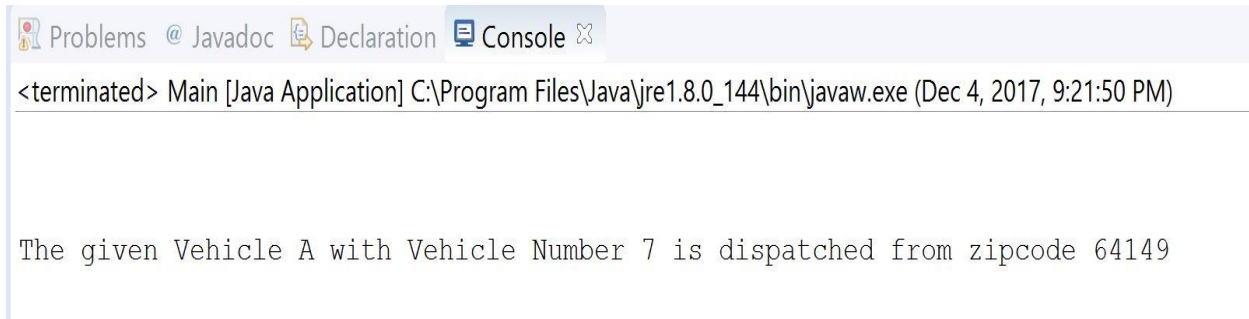
Using Dijkstra's Algorithm,
shortest paths from given zipcode to other zipcodes are displayed below
Distance to 64151: 0.0
Distance to 64149: 10.0
Distance to 64230: 30.0
Distance to 64372: 50.0
Distance to 63210: 60.0

Selected vehicle A is available at 64149
Selected vehicle A is available at 63210
```

Step 5. DISPATCH VEHICLE ALLOCATION

To dispatch the vehicle from the nearest location to the given location, minimum of the shortest paths is taken from the available vehicle locations.

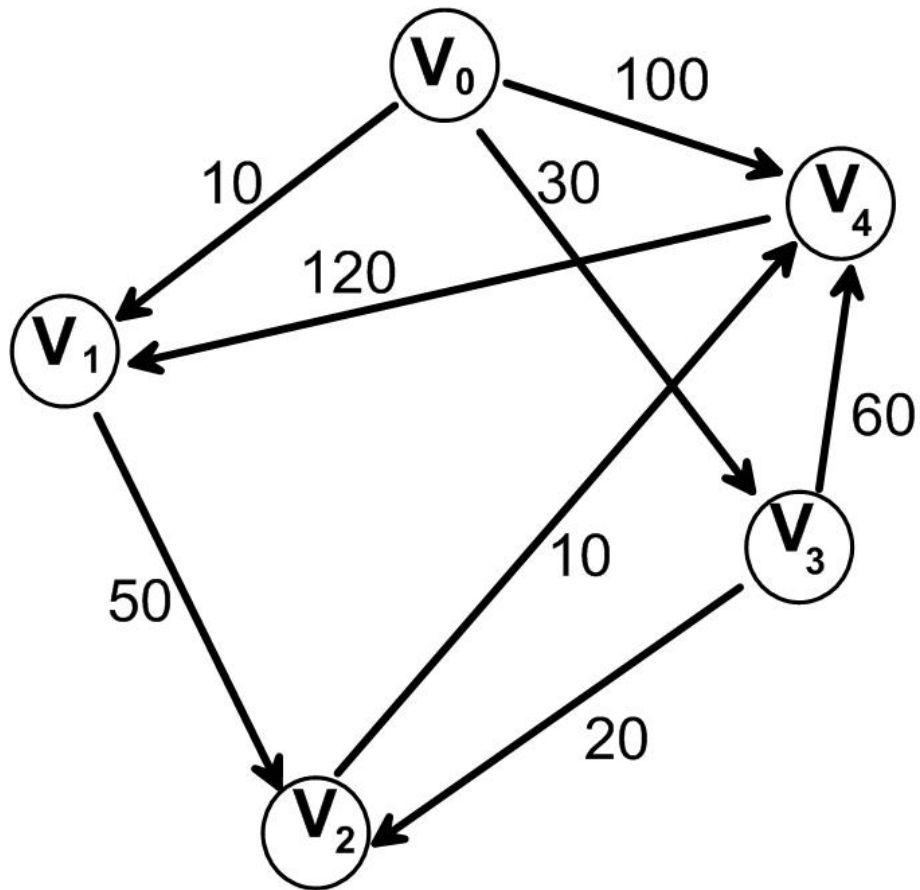
After getting the minimum distance location with the available emergency vehicle, corresponding vehicle ID emergency vehicle is dispatched to the given location.



```
Problems @ Javadoc Declaration Console
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Dec 4, 2017, 9:21:50 PM)

The given Vehicle A with Vehicle Number 7 is dispatched from zipcode 64149
```

ILLUSTRATING DIAGRAM



Shortest Distance from vertex V0 to vertex V1: 10

Shortest Distance from vertex V0 to vertex V2: 50

Shortest Distance from vertex V0 to vertex V3: 30

Shortest Distance from vertex V0 to vertex V4: 60

COMPLEXITY CALCULATION

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://pdfs.semanticscholar.org/d69a/69142f67573c9584eb6e220ca749b2bf30bb.pdf>

<https://stackoverflow.com/questions/8265307/file-input-for-dijkstras-algorithm>

<https://stackoverflow.com/questions/4615814/dijkstra-and-fileinput-java>

<https://stackoverflow.com/questions/1994255/how-to-write-console-output-to-a-txt-file>

<https://stackoverflow.com/questions/22066387/how-to-search-an-element-in-a-jtable-java>