```java
package DS.DSLab;

import java.util.Scanner;

public class cityproblem {

    static final int nCity = 9;

    public void cityList(){
        String[] cit = new String[nCity];
        cit[0] = "City A";
        cit[1] = "City B";
        cit[2] = "City C";
        cit[3] = "City D";
        cit[4] = "City E";
        cit[5] = "City F";
        cit[6] = "City G";
        cit[7] = "City H";
        cit[8] = "City I";

        System.out.println("City List :");

        for (int i = 0; i < cit.length; i++) {
            System.out.println("\t "+ i + ". " + cit[i]);
        }

    }

    public int fare(int citD){
        int i = citD;
        if(i>0 && i<50)
            return 1000;

        if(i>=50 && i<110)
            return 2000;

        if(i>=110 && i<160)
            return 3000;

        if(i>=160 && i<210)
            return 4000;

        if(i>=210)
            return 5000;

            return 0;
    }
    int minDistance(int dist[], Boolean sptSet[])
    {
        int min = Integer.MAX_VALUE, min_index = -1;

        for (int v = 0; v < nCity; v++)
            if (sptSet[v] == false && dist[v] <= min) {
                min = dist[v];
                min_index = v;
            }

        return min_index;
    }

    public void distPrint(int dist[], int cita, int citb){
        int i = citb;

        int citD = dist[i];
        int f = fare(citD);
        System.out.println("\n\tDistance between Cities is " + dist[i] + " and
 fare is " +  f + "INR \n\n");
    }

    void minDistM(int graph[][], int src, int des)
    {
        int dist[] = new int[nCity];

        Boolean sptSet[] = new Boolean[nCity];

        for (int i = 0; i < nCity; i++) {
            dist[i] = Integer.MAX_VALUE;
            sptSet[i] = false;
        }

        dist[src] = 0;


        for (int count = 0; count < nCity - 1; count++) {
            int u = minDistance(dist, sptSet);
            sptSet[u] = true;

            for (int i = 0; i < nCity; i++)
                if (!sptSet[i] && graph[u][i] != 0 && dist[u] != Integer.MAX_V
ALUE && dist[u] + graph[u][i] < dist[i])
                    dist[i] = dist[u] + graph[u][i];
        }

        distPrint(dist, src, des);
    }
    public static void main(String[] args)
    {
        int graph[][] = new int[][] { { 0, 40, 0, 0, 0, 0, 0, 80, 0 },
                                      { 40, 0, 80, 0, 0, 0, 0, 110, 0 },
                                      { 0, 80, 0, 70, 0, 40, 0, 0, 20 },
                                      { 0, 0, 70, 0, 90, 140, 0, 0, 0 },
                                      { 0, 0, 0, 90, 0, 100, 0, 0, 0 },
                                      { 0, 0, 40, 140, 100, 0, 20, 0, 0 },
                                      { 0, 0, 0, 0, 0, 20, 0, 10, 60 },
                                      { 80, 110, 0, 0, 0, 0, 10, 0, 70 },
                                      { 0, 0, 20, 0, 0, 0, 60, 70, 0 } };

        cityproblem t = new cityproblem();

        t.cityList();

        Scanner sc = new Scanner(System.in);
        System.out.print("\n\tEnter Source City Number=> ");
        int cita = sc.nextInt();
        System.out.print("\tEnter Desination City Number=> ");
        int citb = sc.nextInt();

        t.minDistM(graph, cita, citb);
    }

}
```