```java
package edu.wmich.cs3310.a6.DrivingApp;
/*
* CS3310 A5
* Asgn 6 (CS3310 F15)
* @author Jonah Groendal
* Last changed: 12/9/2015
*
* This class uses Djikstra's algorithm to find the shortest path
*/
import java.io.IOException;

public class Route {
    boolean[] included;
    short[] distance;
    int[] path;
    int[] traceOfTargets;
    int targetCounter;

    public void findMinCostPath(String[] cityNames, short startNum, short
destinationNum, short n, Map map, Log log) throws IOException {
        if (startNum > -1 && destinationNum > -1) {
            init3ScratchArrays(n, startNum, map);
            if (startNum != destinationNum) {
                searchForPath(startNum, destinationNum, n, map);
            }
            else {
                traceOfTargets[0] = startNum;
                targetCounter = 1;
            }
        }
        reportAnswer(cityNames, startNum, destinationNum, map, log);
    }
    private void init3ScratchArrays(short n, short startNum, Map map) throws
IOException {
        included = new boolean[n];
        distance = new short[n];
        path = new int[n];
        traceOfTargets = new int[n];

        distance[0] = 0;
        included[0] = true;
        path[0] = -1;
        for (int i=1; i<n; i++) {
            included[i] = false;
            distance[i] = map.getRoadDistance(startNum, i);
            if (distance[i] != 32767) {
                path[i] = startNum;
            }
```

```
            else {
                path[i] = -1;
            }
        }
    }
    private void searchForPath(short startNum, short destinationNum, short n,
Map map) throws IOException {
        targetCounter = 0;
        traceOfTargets[targetCounter] = startNum;
        targetCounter++;
        while (included[destinationNum] == false) {
            // 1) Out of the nodes not yet included, choose the target node
            int target = -1;
            for (int i=0; i<n; i++) {
                if (i == startNum || included[i] == true)
                    continue;
                if(target == -1)
                    target = i;
                else if (distance[i] < distance[target])
                    target = i;
            }

            // 2) Target now becomes included and counted
            included[target] = true;
            traceOfTargets[targetCounter] = target;
            targetCounter++;

            // 3) Check all distance values to see which ones can be lowered
            for (int i=0; i<n; i++) {
                if (included[i] == false) {
                    if (distance[i] != 0) {
                        if (distance[target] + map.getRoadDistance(target, i) <
distance [i]) {
                                distance[i] = (short) (distance[target] +
map.getRoadDistance(target, i));
                                path[i] = target;
                        }
                    }
                }
            }
        }
    }
    private void reportAnswer(String[] cityNames, short start, short end, Map
map, Log log) throws IOException {
        log.displayThisLine(String.format("%s (%d) TO %s (%d)", cityNames
[0].trim(), start, cityNames[1].trim(), end));
        if (start > -1 && end > -1) {
            if (distance[end] < 32767) {
```

```
                log.displayThisLine("DISTANCE:  "+distance[end]);
                log.displayThis("PATH:  ");
                displayPath(map, log, path, start, end);
                log.displayThisLine(map.getCityName(end).trim());
                log.displayThis("TRACE OF TARGETS: ");
                for (int i=0; i<targetCounter; i++) {
                    log.displayThis(" "+map.getCityCode((short)traceOfTargets
[i]));
                }
                log.displayThisLine("");
                log.displayThisLine("# TARGETS:  "+targetCounter);
            }
            else {
                log.displayThisLine("DISTANCE:  ?");
                log.displayThisLine("PATH:  SORRY – can NOT get to destination
city from start city");
                log.displayThisLine("TRACE OF TARGETS: . . .");
                log.displayThisLine("# TARGETS: . . .");
            }
        }
        else
            log.displayThisLine("ERROR – one city is NOT on this map");
        log.displaySeperator();
    }
    private void displayPath(Map map, Log log, int[] path, short start, short
end) throws IOException {
        if (start != path[end]) {
            displayPath(map, log, path, start, (short)path[end]);
        }
        log.displayThis(map.getCityName((short)path[end]).trim()+" > ");
    }
}
```