```
import java.util.*;
class GraphShortestPath {
   private int numVertices;
   private int minDistance(int pathArray[], Boolean sptSet[]) {
      int min = Integer.MAX_VALUE, minIndex = -1;
      for (int v = 0; v < numVertices; v++) {
         if (!sptSet[v] && pathArray[v] <= min) {
            min = pathArray[v];
            minIndex = v;
         }
      }
      return minIndex;
   }
   private void printMinPath(int pathArray[], int parent[], int srcNode) {
      for (int i = 0; i < numVertices; i++) {
         if (i != srcNode) {
            System.out.print("To vertex "+i+" : " + "path ["+ getPath(parent, i, srcNode) +"],"+ "
" + "Distance : "+(pathArray[i] == Integer.MAX_VALUE ? "Infinity" : pathArray[i]));
         }
         else{
            System.out.print("To vertex "+i+" : "+ "path ["+i+"] , Distance : "+ 0 );
         }
         System.out.println();
      }
   }
   private String getPath(int parent[], int vertex, int srcNode) {
      StringBuilder path = new StringBuilder();
      for (int v = vertex; v != srcNode; v = parent[v]) {
         path.insert(0, ", " + v);
      }
```

```
        path.insert(0, srcNode);
        return path.toString();
    }
    public void algoDijkstra(int graph[][], int srcNode) {
        int pathArray[] = new int[numVertices];
        Boolean sptSet[] = new Boolean[numVertices];
        int parent[] = new int[numVertices];
        Arrays.fill(pathArray, Integer.MAX_VALUE);
        Arrays.fill(sptSet, false);
        Arrays.fill(parent, -1);
        pathArray[srcNode] = 0;
        for (int count = 0; count < numVertices - 1; count++) {
            int u = minDistance(pathArray, sptSet);
            sptSet[u] = true;
            for (int v = 0; v < numVertices; v++) {
                if (!sptSet[v] && graph[u][v] != 0 && pathArray[u] != Integer.MAX_VALUE &&
                    pathArray[u] + graph[u][v] < pathArray[v]) {
                    pathArray[v] = pathArray[u] + graph[u][v];
                    parent[v] = u;
                }
            }
        }
        printMinPath(pathArray, parent, srcNode);
    }
    public void setNumVertices(int numVertices) {
        this.numVertices = numVertices;
    }
}
class Main {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of vertices: ");
        int numVertices = scanner.nextInt();
        GraphShortestPath g = new GraphShortestPath();
        g.setNumVertices(numVertices);
        System.out.print("Enter the number of edges: ");
        int numEdges = scanner.nextInt();
        int graph[][] = new int[numVertices][numVertices];
        System.out.println("Enter each edge in the format : Source Destination Weight");
        for (int i = 0; i < numEdges; i++) {
            int src = scanner.nextInt();
            int dest = scanner.nextInt();
            int weight = scanner.nextInt();
            graph[src][dest] = weight;
            graph[dest][src] = weight;
        }
        System.out.print("Enter the starting vertex : ");
        int srcNode = scanner.nextInt();
        System.out.println("Shortest path from vertex " + srcNode + ":");
        g.algoDijkstra(graph, srcNode);
        scanner.close();
    }
}
```