```
 2 * Implementation of Graph
 5import java.io.File;
 8
 9public class Graph
10{
11    private Map<String, Vertex> vertices;
12    private Map<String, List<Edge>> edges;
13
14    public Graph() {
15        vertices = new HashMap<>();
16        edges = new HashMap<>();
17    }
18
19    /**
20     * Reads a graph from the file with the given filename
21     * @param filename - file to be read from
22     */
23    public Graph(String filename) throws FileNotFoundException
24    {
25        vertices = new HashMap<>();
26        edges = new HashMap<>();
27
28        File file = new File(filename);
29        Scanner scanner = new Scanner(file);
30
31        while (scanner.hasNextLine()) {
32            String line = scanner.nextLine().trim();
33            String[] data = line.split(" ");
34
35            String sourceLabel = data[0];
36            String targetLabel = data[1];
37            Double weight = Double.parseDouble(data[2]);
38            addEdge(sourceLabel, targetLabel, weight);
39        }
40
41        scanner.close();
42    }
43
44    /**
```

```
45      * Adds an edge with the given weight from the vertex with
  the given source label to the given target
46      * @param sourceLabel - the label of the source vertex
47      * @param targetLabel - the label of the target vertex
48      * @param weight      - weight of the source vertex to be
  added to
49      */
50     public void addEdge(String sourceLabel, String targetLabel,
  double weight)
51     {
52         Vertex source = getVertex(sourceLabel);
53         Vertex target = getVertex(targetLabel);
54 
55         List<Edge> sourceEdges = edges.get(sourceLabel);
56 
57         Edge edge = new Edge(source, target, weight);
58         sourceEdges.add( edge );
59         edges.put(sourceLabel, sourceEdges);
60     }
61 
62     /**
63      * Returns a list of the edges that have the given vertex
  as their source
64      * @param source - the source where the edges are from
65      * @return a list of the edges that have the given vertex
  as their source
66      */
67     public List<Edge> getAdjacent(Vertex source)
68     {
69         return
  Collections.unmodifiableList(edges.get(source.label));
70     }
71 
72     /**
73      * Returns a set of the vertices in the graph
74      * @return a set of the vertices in the graph
75      */
76     public Collection<Vertex> getVertices()
77     {
```

```
 78         return
   Collections.unmodifiableCollection( vertices.values() );
 79     }
 80
 81     /**
 82      * Returns string representation of the summary of the graph
 83      * @return string representation of the summary of the graph
 84      */
 85     public void printMST()
 86     {
 87         int totalWeight = 0;
 88         List<Edge> edgeList = new LinkedList<>();
 89
 90         for (String label : edges.keySet() ) {
 91             List<Edge> adjEdges = edges.get(label);
 92             for (Edge edge : adjEdges) {
 93                 edgeList.add(edge);
 94                 totalWeight += edge.weight;
 95             }
 96         }
 97
 98         edgeList.sort( new EdgeComparator() );
 99         for (Edge edge : edgeList) {
100             System.out.println(edge);
101         }
102         System.out.println("Total Weight: " + totalWeight);
103     }
104
105     /**
106      * Returns a list of all edges of the graph
107      * @return a list of all edges of the graph
108      */
109     public List<Edge> getEdges()
110     {
111         List<Edge> list = new LinkedList<>();
112         for (String label : edges.keySet()) {
113             List<Edge> adjEdges = edges.get(label);
114             list.addAll(adjEdges);
115         }
```

```
116         return list;
117     }
118 
119     /**
120      * Returns a Vertex object for the given label and its index
121      * @param label - the given label to get vertex
122      * @return a vertex object for the given label and its index
123      */
124     public Vertex getVertex(String label)
125     {
126         if (!vertices.containsKey(label)) {
127             int idx = vertices.size();
128 
129             Vertex vertex = new Vertex(label, idx);
130             vertices.put(label, vertex);
131 
132             List<Edge> sourceEdges = new LinkedList<Edge>();
133             edges.put(label, sourceEdges);
134         }
135 
136         return vertices.get(label);
137     }
138 
139     /**
140      * Returns an array of labels of the vertices in the graph
141      * @return an array of labels of the vertices in the graph
142      */
143     public String[] getLabels()
144     {
145         String[] labels = new String[vertices.size()];
146         for (Vertex v : this.getVertices()) {
147             int idx = v.index;
148             labels[idx] = v.label;
149         }
150         return labels;
151     }
152 
153     /**
154      * Returns the adjacency matrix of the graph
```

```
155     * @return the adjacency matrix of the graph
156     */
157    public double[][] getMatrix()
158    {
159        int n = vertices.size();
160        double[][] matrix = new double[n][n];
161        for (int i = 0; i < n; i++) {
162            Arrays.fill(matrix[i], Double.POSITIVE_INFINITY);
163        }
164        for (int i = 0; i < n; i++) {
165            matrix[i][i] = 0;
166        }
167
168        for (List<Edge> edgeList : edges.values()) {
169            for (Edge edge : edgeList) {
170                int sourceIndex = edge.source.index;
171                int targetIndex = edge.target.index;
172                matrix[sourceIndex][targetIndex] = edge.weight;
173            }
174        }
175        return matrix;
176    }
177
178}
```