```java
 * Implementation of Graph
import java.io.File;

public class Graph
{
    private Map<String, Vertex> vertices;
    private Map<String, List<Edge>> edges;

    public Graph()
    {
//      vertices = new HashMap<>();
//      edges = new HashMap<>();
    }

    /**
     * Reads a graph from the file with the given filename
     * @param filename - file to be read from
     */
    public Graph(String filename) throws FileNotFoundException
    {
        this();
        Scanner scanner = new Scanner(new File(filename));

        while (scanner.hasNext()) {
            String sourceLabel = scanner.next();
            String targetLabel = scanner.next();
            double weight = scanner.nextDouble();

            addEdge(sourceLabel, targetLabel, weight);
        }
        scanner.close();
    }

    /**
     * Adds an edge with the given weight from the vertex with the given source label to the given target
     * @param sourceLabel - the label of the source vertex
     * @param targetLabel - the label of the target vertex
     * @param weight      - weight of the source vertex to be
```

```java
added to
     */
    public void addEdge(String sourceLabel, String targetLabel, double weight)
    {
        Vertex source = getVertex(sourceLabel);
        Vertex target = getVertex(targetLabel);

        List<Edge> sourceEdges = edges.get(sourceLabel);

        Edge edge = new Edge(source, target, weight);
        sourceEdges.add( edge );
        edges.put(sourceLabel, sourceEdges);
    }

    /**
     * Returns a list of the edges that have the given vertex as their source
     * @param source - the source where the edges are from
     * @return a list of the edges that have the given vertex as their source
     */
    public List<Edge> getAdjacent(Vertex source)
    {
        return Collections.unmodifiableList(edges.get(source.label));
    }

    /**
     * Returns a set of the vertices in the graph
     * @return a set of the vertices in the graph
     */
    public Collection<Vertex> getVertices()
    {
        return Collections.unmodifiableCollection(vertices.values());
    }

    /**
```

```java
     * Returns string representation of the summary of the graph
     * @return string representation of the summary of the graph
     */
    public Graph printMST()
    {
        return null;
    }

    /**
     * Returns a list of all edges of the graph
     * @return a list of all edges of the graph
     */
    public List<Edge> getEdges()
    {
        return null;
    }

    /**
     * Returns a Vertex object for the given label and its index
     * @param label - the given label to get vertex
     * @return a vertex object for the given label and its index
     */
    public Vertex getVertex(String label)
    {
        if (!vertices.containsKey(label)) {
            Vertex vertex = new Vertex(label);
            vertices.put(label, vertex);
            List<Edge> sourceEdges = new LinkedList<Edge>();
            edges.put(label, sourceEdges);
        }
        return vertices.get(label);
    }

    /**
     * Returns an array of labels of the vertices in the graph
     * @return an array of labels of the vertices in the graph
     */
    public String[] getLabels()
    {
        return null;
    }

    /**
     * Returns the adjacency matrix of the graph
     * @return the adjacency matrix of the graph.
     */
    public double[][] getMatrix()
    {
        return null;
    }

}
```