```java
package graph;

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.ListIterator;

class AdjacencyTable {

    class Node {
        String name;
        Integer weight;

        Node(String name, Integer weight) {
            this.name = name;
            this.weight = weight;
        }
    }

    private HashMap<String, Integer> vertexNameAndIndexTable;
    private LinkedList<LinkedList<Node>> innerAdjacencyTable;
    private Integer index = 0;

    public AdjacencyTable(String[] vertex) {

        innerAdjacencyTable = new LinkedList<LinkedList<Node>>();
        vertexNameAndIndexTable = new HashMap<>();

        for (int i = 0; i < vertex.length; i++) {
            LinkedList<Node> currentList = new LinkedList<Node>();
            currentList.add(new Node(vertex[i], null));
            innerAdjacencyTable.add(currentList);
            vertexNameAndIndexTable.put(vertex[i], index);
            index++;
        }
    }

    public void updateEdge(String srcEdge, String dstEdge, Integer weight) {

        for (LinkedList<Node> currentList : innerAdjacencyTable) {

            if (currentList.get(0).name.equals(srcEdge)) {

                for (Node currentNode : currentList) {
                    if (currentNode.name.equals(dstEdge)) {
                        currentNode.weight = weight;
                        return;
                    }
                }

                currentList.add(new Node(dstEdge, weight));
                return;
            }
        }

        LinkedList<Node> newRow = new LinkedList<>();
        newRow.add(new Node(srcEdge, null));
        newRow.add(new Node(dstEdge, weight));
        innerAdjacencyTable.add(newRow);
        vertexNameAndIndexTable.put(srcEdge, index);
        index++;
    }

    public void deleteEdge(String srcEdge, String dstEdge) {

        for (LinkedList<Node> currentList : innerAdjacencyTable) {

            if (currentList.get(0).name.equals(srcEdge)) {

                for (Node currentNode : currentList) {
                    if (currentNode.name.equals(dstEdge)) {
                        currentList.remove(currentNode);
                        return;
```

```
                    }
                }
            }
        }
    }

    public Integer getWeight(String srcEdge, String dstEdge) {
        for (LinkedList<Node> currentList : innerAdjacencyTable) {
            if (currentList.get(0).name.equals(srcEdge)) {

                for (Node currentNode : currentList) {
                    if (currentNode.name.equals(dstEdge)) {
                        return currentNode.weight;
                    }
                }
            }
        }

        return null;
    }

    public LinkedList<LinkedList<Node>> getInnerAdjacencyTable() {

        return innerAdjacencyTable;
    }

    public Integer getIndex(String name) {
        return vertexNameAndIndexTable.get(name);
    }

    public static void main(String[] args) {

        String[] vertex = {"1", "2", "3", "4", "5", "6", "7"};
        AdjacencyTable adjacencyTable = new AdjacencyTable(vertex);

        adjacencyTable.updateEdge("1", "2", 1);
        adjacencyTable.updateEdge("3", "1", 6);
        adjacencyTable.updateEdge("4", "1", 3);
        adjacencyTable.updateEdge("2", "3", 4);
        adjacencyTable.updateEdge("2", "4", 4);
        adjacencyTable.updateEdge("4", "3", 9);
        adjacencyTable.updateEdge("6", "2", 7);
        adjacencyTable.updateEdge("7", "6", 7);
        adjacencyTable.updateEdge("7", "4", 7);
        adjacencyTable.updateEdge("5", "7", 7);
        adjacencyTable.updateEdge("5", "6", 7);


        System.out.println(adjacencyTable.getWeight("1", "2"));
        System.out.println(adjacencyTable.getWeight("1", "3"));

        System.out.println("haha");
    }
}
```