

```

1  package graph;
2
3  import java.util.HashMap;
4  import java.util.Iterator;
5  import java.util.LinkedList;
6  import java.util.ListIterator;
7
8  class AdjacencyTable {
9
10     class Node {
11         String name;
12         Integer weight;
13
14         Node(String name, Integer weight) {
15             this.name = name;
16             this.weight = weight;
17         }
18     }
19
20     private HashMap<String, Integer> vertexNameAndIndexTable;
21     private LinkedList<LinkedList<Node>> innerAdjacencyTable;
22     private Integer index = 0;
23
24     public AdjacencyTable(String[] vertex) {
25
26         innerAdjacencyTable = new LinkedList<LinkedList<Node>>();
27         vertexNameAndIndexTable = new HashMap<>();
28
29         for (int i = 0; i < vertex.length; i++) {
30             LinkedList<Node> currentList = new LinkedList<Node>();
31             currentList.add(new Node(vertex[i], null));
32             innerAdjacencyTable.add(currentList);
33             vertexNameAndIndexTable.put(vertex[i], index);
34             index++;
35         }
36     }
37
38     public void updateEdge(String srcEdge, String dstEdge, Integer weight) {
39
40         for (LinkedList<Node> currentList : innerAdjacencyTable) {
41
42             if (currentList.get(0).name.equals(srcEdge)) {
43
44                 for (Node currentNode : currentList) {
45                     if (currentNode.name.equals(dstEdge)) {
46                         currentNode.weight = weight;
47                         return;
48                     }
49                 }
50
51                 currentList.add(new Node(dstEdge, weight));
52                 return;
53             }
54         }
55
56         LinkedList<Node> newRow = new LinkedList<>();
57         newRow.add(new Node(srcEdge, null));
58         newRow.add(new Node(dstEdge, weight));
59         innerAdjacencyTable.add(newRow);
60         vertexNameAndIndexTable.put(srcEdge, index);
61         index++;
62     }
63
64     public void deleteEdge(String srcEdge, String dstEdge) {
65
66         for (LinkedList<Node> currentList : innerAdjacencyTable) {
67
68             if (currentList.get(0).name.equals(srcEdge)) {
69
70                 for (Node currentNode : currentList) {
71                     if (currentNode.name.equals(dstEdge)) {
72                         currentList.remove(currentNode);
73                         return;

```

```

74 ..... }
75 ..... }
76 ..... }
77 ..... }
78 ..... }
79
80 .... public Integer getWeight(String srcEdge, String dstEdge) {
81 ..... for (LinkedList<Node> currentList : innerAdjacencyTable) {
82 .....     if (currentList.get(0).name.equals(srcEdge)) {
83
84 .....         for (Node currentNode : currentList) {
85 .....             if (currentNode.name.equals(dstEdge)) {
86 .....                 return currentNode.weight;
87 .....             }
88 .....         }
89 .....     }
90 ..... }
91
92 ..... return null;
93 ..... }
94
95 .... public LinkedList<LinkedList<Node>> getInnerAdjacencyTable() {
96
97 ..... return innerAdjacencyTable;
98 ..... }
99
100 .... public Integer getIndex(String name) {
101 ..... return vertexNameAndIndexTable.get(name);
102 ..... }
103
104 .... public static void main(String[] args) {
105
106 ..... String[] vertex = {"1", "2", "3", "4", "5", "6", "7"};
107 ..... AdjacencyTable adjacencyTable = new AdjacencyTable(vertex);
108
109 ..... adjacencyTable.updateEdge("1", "2", 1);
110 ..... adjacencyTable.updateEdge("3", "1", 6);
111 ..... adjacencyTable.updateEdge("4", "1", 3);
112 ..... adjacencyTable.updateEdge("2", "3", 4);
113 ..... adjacencyTable.updateEdge("2", "4", 4);
114 ..... adjacencyTable.updateEdge("4", "3", 9);
115 ..... adjacencyTable.updateEdge("6", "2", 7);
116 ..... adjacencyTable.updateEdge("7", "6", 7);
117 ..... adjacencyTable.updateEdge("7", "4", 7);
118 ..... adjacencyTable.updateEdge("5", "7", 7);
119 ..... adjacencyTable.updateEdge("5", "6", 7);
120
121
122 ..... System.out.println(adjacencyTable.getWeight("1", "2"));
123 ..... System.out.println(adjacencyTable.getWeight("1", "3"));
124
125 ..... System.out.println("haha");
126 ..... }
127 }

```