Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	12-01
Receipt ID:	c889b78135c60f5bf8504a1bc2f90c91

Warning: Your turnin is 5 days late. Assignment 12-01 was due Thursday, June 1, 2017, 4:00 PM.

Turnin Successful!

The following file(s) were received:

TJGraphAdjMat.java (4478 bytes)

```
1. //name:   date:
2. // resource classes and interfaces
3. // for use with Graphs0: Intro
4. //              Graphs1: Wardhall
5. //              Graphs2: Floyd
6. import java.util.*;
7. import java.io.*;
8.
9. interface AdjacencyMatrix
10. {
11.    public void addEdge(int source, int target);
12.    public void removeEdge(int source, int target);
13.    public boolean isEdge(int from, int to);
14.    public void displayGrid();
15.    public int edgeCount();
16.    public List<Integer> getNeighbors(int source);
17.
18. }
19.
20. interface Warshall
21. {
22.    //User-friendly functionality
23.    public boolean isEdge(String from, String to);
24.    public Map<String, Integer> getVertices();
25.    public void readNames(String fileName) throws FileNotFoundException;
26.    public void readGrid(String fileName) throws FileNotFoundException;
27.    public void displayVertices();
28.    //Actual Warshall Algorithm
29.    public void allPairsReachability();
30. }
31.
32. interface Floyd
33. {
34.    public int getCost(int from, int to);
35.    public int getCost(String from, String to);
36.    public void allPairsWeighted();
37. }
38.
39. public class TJGraphAdjMat implements AdjacencyMatrix, Warshall //Floyd
40. {
41.    private int[][] grid = null;   //adjacency matrix representation
42.    private Map<String, Integer> vertices = null;
```

```
43.    private Map<Integer, String> vertices2 = null;
44.
45.    /*  enter your code here  */
46.    public TJGraphAdjMat(int size)
47.    {
48.       grid = new int[size][size];
49.       vertices = new TreeMap<String, Integer>();
50.       vertices2 = new HashMap<Integer, String>();
51.    }
52.
53.    public void addEdge(int source, int target)
54.    {
55.       grid[source][target]=1;
56.    }
57.
58.    public void removeEdge(int source, int target)
59.    {
60.       if(grid[source][target]==1)
61.          grid[source][target]=0;
62.       else
63.          System.out.println("Thats not an edge");
64.    }
65.
66.    public boolean isEdge(int from, int to)
67.    {
68.       return grid[from][to]==1;
69.    }
70.    public boolean isEdge(String from, String to)
71.    {
72.       return grid[vertices.get(from)][vertices.get(to)]==1;
73.    }
74.
75.    public void displayGrid()
76.    {
77.       for(int[] x: grid)
78.       {
79.          for(int y: x)
80.          {
81.             System.out.print(y+" ");
82.          }
83.          System.out.println("");
84.       }
85.    }
86.
87.    public int edgeCount()
88.    {
89.       int count = 0;
90.
91.       for(int[] x: grid)
92.       {
93.          for(int y: x)
94.          {
95.             if(y==1)
96.                count++;
97.          }
98.       }
99.       return count;
100.   }
101.   public List<Integer> getNeighbors(int source)
102.   {
103.      List<Integer> list = new ArrayList<Integer>();
104.      for(int i=0;i<grid[source].length;i++)
105.      {
106.         if(isEdge(source, i))
107.            list.add(i);
108.      }
109.      return list;
110.   }
111.
112.   public Map<String, Integer> getVertices()
113.   {
114.      return vertices;
115.   }
116.
117.   public void readNames(String fileName) throws FileNotFoundException
118.   {
119.      Scanner infile = new Scanner(new File(fileName));
120.      int number = infile.nextInt();
121.      for(int i=0;i<number;i++)
122.      {
123.         String str = infile.next();
```

```
124.          vertices.put(str, new Integer(i));
125.          vertices2.put(new Integer(i), str);
126.       }
127.    }
128.
129.    public void readGrid(String fileName) throws FileNotFoundException
130.    {
131.       Scanner infile = new Scanner(new File(fileName));
132.       int number = infile.nextInt();
133.       for(int i=0;i<number;i++)
134.       {
135.          for(int j=0;j<number;j++)
136.          {
137.             grid[i][j] = infile.nextInt();
138.          }
139.       }
140.    }
141.
142.    public void displayVertices()
143.    {
144.       Set<String> s = vertices.keySet();
145.       for(String str: s)
146.       {
147.          System.out.println(vertices.get(str)+"-"+str);
148.       }
149.       System.out.println();
150.    }
151.
152.    public void allPairsReachability()
153.    {
154.       for(int v=0;v<grid.length;v++)
155.       {
156.          for(int i=0;i<grid.length;i++)
157.          {
158.             for(int j=0;j<grid.length;j++)
159.             {
160.                if(grid[i][v]==1&&grid[v][j]==1)
161.                {
162.                   grid[i][j]=1;
163.                }
164.             }
165.          }
166.       }
167.    }
168.    public ArrayList<String> getReachables(String city)
169.    {
170.       ArrayList<String> reach = new ArrayList<String>();
171.       int i = vertices.get(city);
172.       for(int x=0;x<grid[0].length;x++)
173.       {
174.          if(grid[i][x]==1)
175.          {
176.             reach.add(vertices2.get(x));
177.          }
178.       }
179.       return reach;
180.    }
181. }
182.
```