Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	12-02
Receipt ID:	587297e1fdd66f69c8e30c40972e43a6

Warning: Your turnin is 1 day late. Assignment 12-02 was due Monday, June 5, 2017, 4:00 PM.

Turnin Successful!

The following file(s) were received:

TJGraphAdjMat.java (5241 bytes)

```java
//name:   date:
// resource classes and interfaces
// for use with Graphs0: Intro
//                Graphs1: Wardhall
//                Graphs2: Floyd
import java.util.*;
import java.io.*;

interface AdjacencyMatrix
{
   public void addEdge(int source, int target);
   public void removeEdge(int source, int target);
   public boolean isEdge(int from, int to);
   public void displayGrid();
   public int edgeCount();
   public List<Integer> getNeighbors(int source);

}

interface Warshall
{
   //User-friendly functionality
   public boolean isEdge(String from, String to);
   public Map<String, Integer> getVertices();
   public void readNames(String fileName) throws FileNotFoundException;
   public void readGrid(String fileName) throws FileNotFoundException;
   public void displayVertices();
   //Actual Warshall Algorithm
   public void allPairsReachability();
}

interface Floyd
{
   public int getCost(int from, int to);
   public int getCost(String from, String to);
   public void allPairsWeighted();
}

public class TJGraphAdjMat implements AdjacencyMatrix, Warshall //Floyd
{
   private int[][] grid = null;   //adjacency matrix representation
   private Map<String, Integer> vertices = null;
   private Map<Integer, String> vertices2 = null;

   /*  enter your code here  */
   public TJGraphAdjMat(int size)
   {
      grid = new int[size][size];
      vertices = new TreeMap<String, Integer>();
      vertices2 = new HashMap<Integer, String>();
   }

   public void addEdge(int source, int target)
   {
      grid[source][target]=1;
   }

   public void removeEdge(int source, int target)
   {
      if(grid[source][target]==1)
         grid[source][target]=0;
      else
         System.out.println("Thats not an edge");
   }

   public boolean isEdge(int from, int to)
   {
      return grid[from][to]<9999&&grid[from][to]!=0;
   }
   public boolean isEdge(String from, String to)
   {
      return grid[vertices.get(from)][vertices.get(to)]<9999&&grid[vertices.get(from)][vertices.get(to)]!=0;
   }

   public void displayGrid()
   {
      for(int[] x: grid)
      {
         for(int y: x)
         {
            System.out.print(y+" ");
         }
         System.out.println("");
      }
   }

   public int edgeCount()
   {
      int count = 0;

      for(int x=0; x<grid.length; x++)
      {
         for(int y=0; y<grid[0].length; y++)
         {
            if(isEdge(x, y))
               count++;
         }
      }
      return count;
   }
   public List<Integer> getNeighbors(int source)
   {
      List<Integer> list = new ArrayList<Integer>();
      for(int i=0;i<grid[source].length;i++)
      {
         if(isEdge(source, i))
            list.add(i);
      }
      return list;
   }

   public int getCost(int from, int to)
   {
      return grid[from][to];
   }

   public int getCost(String from, String to)
   {
      return grid[vertices.get(from)][vertices.get(to)];
   }

   public Map<String, Integer> getVertices()
   {
      return vertices;
   }

   public void readNames(String fileName) throws FileNotFoundException
   {
      Scanner infile = new Scanner(new File(fileName));
      int number = infile.nextInt();
      for(int i=0;i<number;i++)
      {
         String str = infile.next();
         vertices.put(str, new Integer(i));
         vertices2.put(new Integer(i), str);
      }
   }

   public void readGrid(String fileName) throws FileNotFoundException
   {
      Scanner infile = new Scanner(new File(fileName));
      int number = infile.nextInt();
      for(int i=0;i<number;i++)
      {
         for(int j=0;j<number;j++)
         {
            grid[i][j] = infile.nextInt();
         }
      }
   }

   public void displayVertices()
   {
      Set<String> s = vertices.keySet();
      for(String str: s)
      {
         System.out.println(vertices.get(str)+"-"+str);
      }
      System.out.println();
   }

   public void allPairsReachability()
   {
      for(int v=0;v<grid.length;v++)
      {
         for(int i=0;i<grid.length;i++)
         {
            for(int j=0;j<grid.length;j++)
            {
               if(grid[i][v]==1&&grid[v][j]==1)
               {
                  grid[i][j]=1;
               }
            }
         }
      }
   }

   public void allPairsWeighted()
   {
      for(int v=0;v<grid.length;v++)
      {
         for(int i=0;i<grid.length;i++)
         {
            for(int j=0;j<grid.length;j++)
            {
               if(i==j);
               else if(isEdge(i, v) && isEdge(v, j))
               {
                  grid[i][j]=getCost(i, v) + getCost(v, j);
               }
            }
         }
      }
   }

   public ArrayList<String> getReachables(String city)
   {
      ArrayList<String> reach = new ArrayList<String>();
      int i = vertices.get(city);
      for(int x=0;x<grid[0].length;x++)
      {
         if(grid[i][x]==1)
         {
                reach.add(vertices2.get(x));
            }
        }
        return reach;
    }


}

```