**Project 6**

Every conversation in our area ends up in a discussion about traffic. In this project you will be creating an application to maintain a network of towns and roads connecting them.

The application will use Dijkstra’s Shortest Path algorithm to find the shortest distance between any two towns. Follow the interfaces, the below specifications, and the JUnit test files.

**Specifications**

**Data Element – Town (Vertex) –** Create a Town class that holds the name of the town and a list of adjacent towns, and other fields as desired, and the traditional methods (constructors, getters/setters, toString, etc.). It will implement the Comparable interface. This is the class header:

public class Town implements Comparable<Town>

Two towns will be considered the same if their name is the same.

**Data Element – Road (Edge)**

Create a class Road that can represent the edges of a Graph of Towns. The class must implement Comparable. The class stores references to the two vertices (Town endpoints), the distance between vertices, and a name, and the traditional methods (constructors, getters/setters, toString, etc.), and a compareTo, which compares two Road objects. Since this is a undirected graph, an edge from A to B is equal to an edge from B to A. This is the class header:

public class Road implements Comparable<Road>

**The Data Structure –Graph, implements GraphInterface**

* Create a Graph class that implements the GraphInterface given you. For Graph<V,E>, V is the vertex type (a Town), E is the edge type (a Road). You will need to decide how to store the graph, using an adjacent matrix, an adjacency list, or a Set<Town> and Set<Road>. This is the class header:

public class Graph implements GraphInterface<Town, Road>

Within the Graph interface is a method shortestPath, which finds the shortest path from a given Town to a destination Town. Since there is a unique shortest path from every vertex to the source, there is a back-pointer to the previous vertex. The method shortestPath calls dijkstraShortestPath which finds the shortest path from the source to every other vertex in the graph. You will be coding the Dijkstra’s Shortest Path algorithm. You will then be able to find the connections between two towns through the roads that connect them.

You may use the adjacency matrix or list approach found in the text book, or you may use a set of Towns and a set of Roads. The ShortestPath algorithm typically uses a weighted graph which means that the edges have a weight, and this is used to determine the shortest path. For this implementation, each weight will be the distance of the road in miles.

**The Data Manager – implements DataManagerInterface**

Your TownGraphManager will hold an object of your Graph. Implement the TownGraphManagerInterface. There are methods to populate the graph (reading from a text file), add a town (vertices), add a road (edge), list all towns and all roads, and list towns adjacent to a given town.

Your solution will find the shortest path from a start town to a destination town. It will account for the possibility of a disjoint graph (i.e., not all vertices can be reached from all other vertices.)

You may add any methods as needed for your design.

**Populating the Data Structure**

You will be reading from a data file. You are provided with two sample files: MD Towns.txt and US Towns.txt along with two PowerPoint slides showing these graphs.

The Towns.txt files hold the information for the individual Towns and Roads, and is in the following format:

road-name,miles;town-name; town-name

For example:

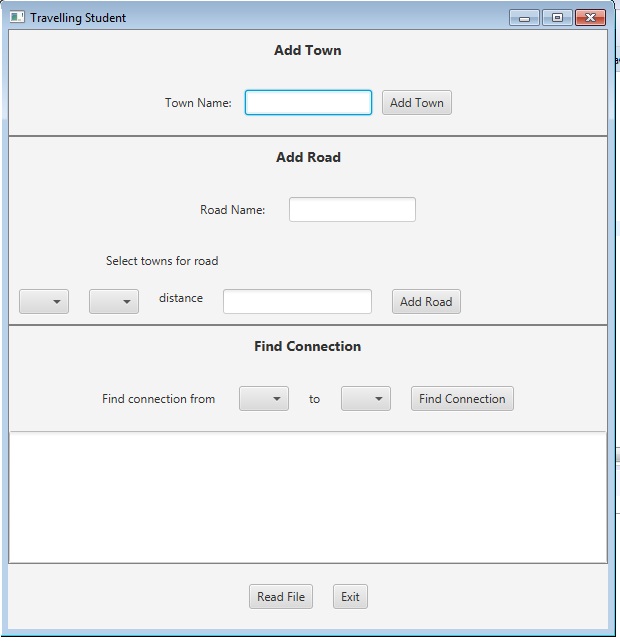
I-94,282;Chicago;Detroit

Notice that the road-name and miles are separated by a comma, while the road information and the two towns are separated by semi-colons.

After reading these files, you will have an initial set of vertices and edges in your Graph.

**The GUI**

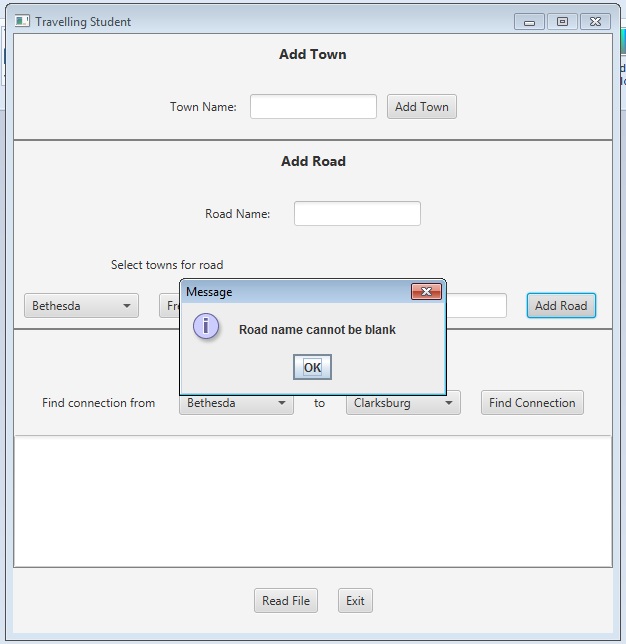
The GUI will have four sections: an Add Town section, an Add Road section, a Find Connection section, and an administration section. There will be four ComboBoxes each containing the same list of Towns. On startup the graph will be empty.



**Add a Town Button**

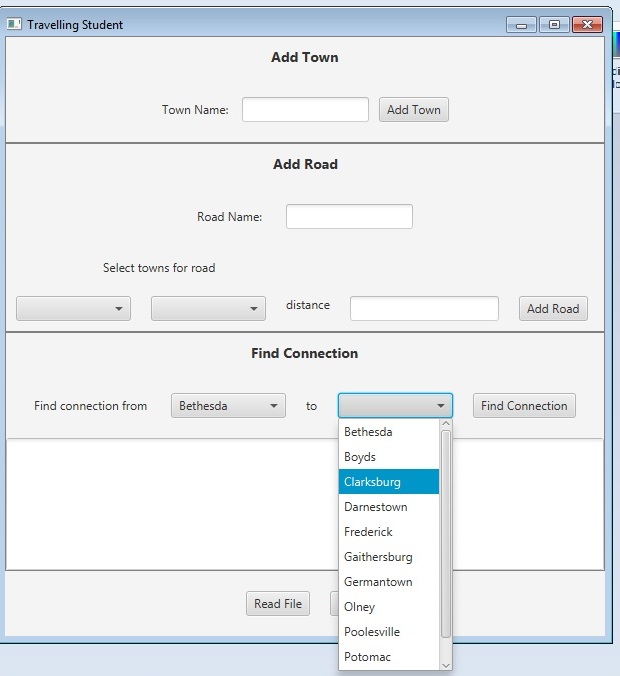
The user may add a new Town by typing its name in the textfield. If the textfield is blank when the Add Town button is selected, the GUI should show an error message. When a new Town is added, the TownGraphManager will add it to the graph, and the Town’s name will be added to the four ComboBoxes.

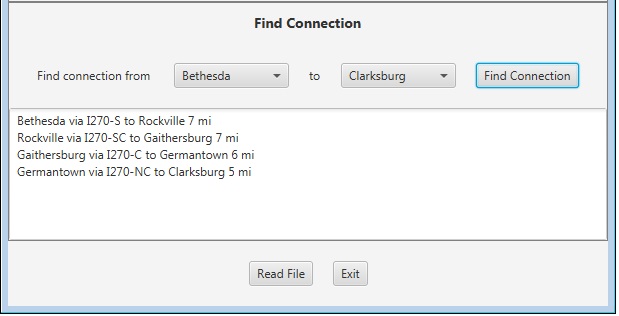
**Add a Road Button**

To add a road, a town must be selected from each of the two ComboBoxes in the Add Road section, an integer distance entered, and a road name entered. When the Add Road button is selected, the edge is created and entered in the graph.

**Find Connection Button**

Display all the available towns in the ComboBoxes (in alpha order by name). When the user selects the towns, display the name in the ComboBoxes. When the user selects the “Find Connection” button, the TownGraphManager’s shortestPath method is called. The resulting list of roads connecting towns, and the distance along each road, is displayed in the text area.





If the “source” town and “destination” town are the same, or if there is no route between the two, state that in the text area.

**Read File Button**

The Towns.txt files hold information for individual Towns and Roads, and is in the following format:

road-name**,**miles**;**town-name**;**town-name

For example:

I-94,282;Chicago;Detroit

Notice that the road-name and miles are separated by a comma, while the road information and the two towns are separated by semi-colons.

**Exit Button**

The program will terminate.

**Concepts Utilized in Project**

* Implement Graph Interface
* Use Graph to maintain a network of Vertices
* Implement Shortest Path Algorithm

**Write-up & Submission Requirements**

Review the provided rubric (xls file) to understand project expectations, including the documentation, CMSC204, and programming requirements.

In addition to completing the Java project, a write-up is required. At a minimum, the write-up should address -

* Approach, Design & Algorithm
  + DO NOT start coding your project immediately! Come up with a high level design of the project
    - What’s your game plan to tackle the project?
    - Break the project into smallest modules
  + Each student is welcome to expand on the design, if it makes sense. Students will not be penalized for going “above and beyond” the specifications of the project
  + **Complete this step first, then write your code**
* Test Plan & Test Cases
  + Which test cases did you test your program against? (Did you run your program for different type of employees?)
  + What did you do with your program, as far as ensuring that it’s working?
  + I want to see your “thinking,” as to how you are testing your program
  + **Ensure that your project can successfully pass the provided “public test cases.” Your instructor will test your project using a set of “private test cases.”**
  + In theory, each submitted project should be a “rock solid” working program with “near zero” bugs
  + Capture screenshots of most of your test runs in your write-up
* Any assumptions that you are making for this project
* Highlight your learning experience and lessons learned
  + **I am very interested to learn about what you have done, how you did, etc.**
* Anything else that you want to share with me
* If the project consists of several applications, include several paragraphs to discuss each application in details in a single write-up. Submit one write-up (Word document or PDF file) for the (entire) project

Each student must submit one compressed (.zip) file back to the Assignment (link) with the following deliverables:

* Source code – one or more **project folder(s)** of all of the code that you have written while working on this project
* Write-up (in Word or PDF – one write-up per project)

Name the compressed file (zip format) as <lastname>\_project\_x

* + where x is the project number and your last name (e.g. Thai\_Project\_1.zip)
* Review provided instructions on how to submit the project carefully (don’t assume anything)
* Double check your submission, as I can only grade what’s being submitted
* I MUST BE able to compile, run and test every submitted project on my computer
* A set of public and private test cases will be test against your project

**Not clear? That’s okay, but do ask your questions. “I did not know” or “I did not understand” is not good enough.**

Starting working on each project immediately so that we can discuss any concerns or questions you have!

**Academic Honesty Policy Reminder** – Each submitted project will be compared against other submissions from current and previous semesters.

**Sample Test Runs & Outputs**

None.