**Project 6 Write Up: Dolev Peleg**

**Approach, Design, and Algorithm**

This assignment was the most complicated for me from all the assignments this semester. As our instructor said, graphs are one of the most complicated topics of CMSC 204, and this was my first time coding graphs. Although I understand the concepts and algorithms regarding graphs, coding them was a much more complicated task

I started this project by first designing the most basic classes, Town and Road. These classes were easy to code, and I did not encounter any issues when working on them.

Next, I started to design the Graph class. When going through my notes, I decided that I will try to work with the adjacency list type of graph, so I can practice working more with linked lists rather than arrays. While it was challenging to design most of the class, I managed to code all methods besides shortestPath and dijkstraShortestPath. I could not find any full example for this algorithm within the book, and almost all the examples that I found online were using an adjacency matrix rather than an adjacency list. I decided to use an example algorithm that I found online, and alter it to fit my adjacency list instead (the link to the example that I used is cited under “work cited” below). This was still very challenging, and due to my need to focus on preparing for the upcoming final, I decided to not fully implement this method so I will not have to change too much of my original code and “break” my classes.

After finishing with the Graph class, I attended the TownGraphManager class. This class was not too complicated to implement after I already tested the Graph class and made sure that it was working as expected. I only encountered small issues that were easy to fix.

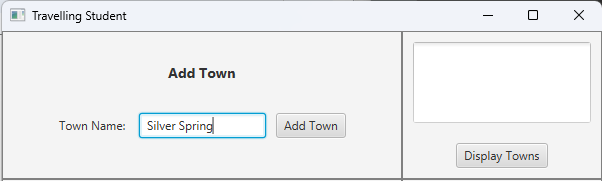
When testing my classes, the Graph\_GFA\_Test “failed” although the output was correct (screenshot will be cited below). I made sure that I am indeed correct by trying to call “assertTrue(graph.getEdge(town[1], town[0]).equals(new Road(town[1], town[0],2, "Road\_1")));”

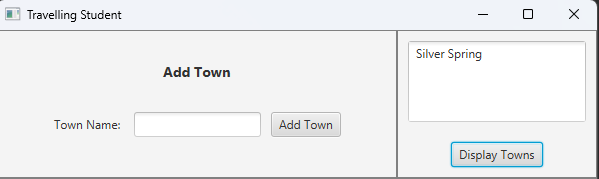
which passed, rather than “assertEquals(new Road(town[1], town[0],2, "Road\_1"), graph.getEdge(town[1], town[0]));” which failed.

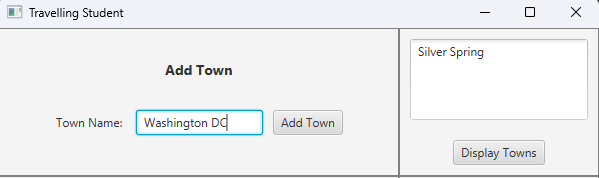
Besides shortestPath and the dijkstraShortestPath methods (and therefore the getPath method), my code ran as expected, and there were no other issues within my code and my tests.

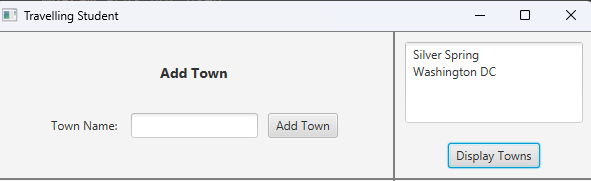
**Test Runs and Cases**

**Adding Towns:**

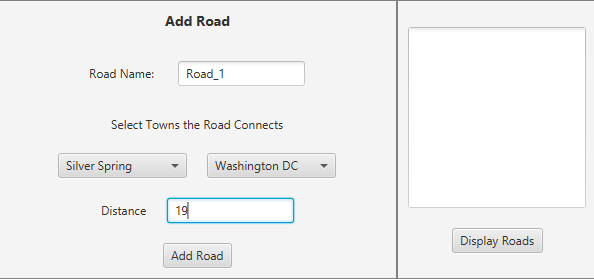
****

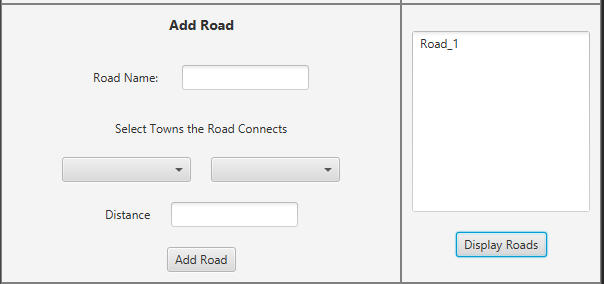
****

****

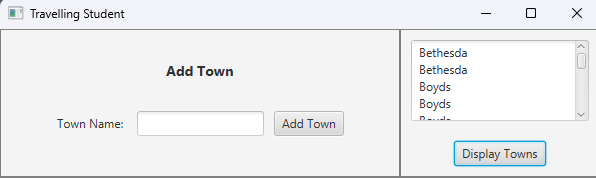
****

**Adding a Road:**

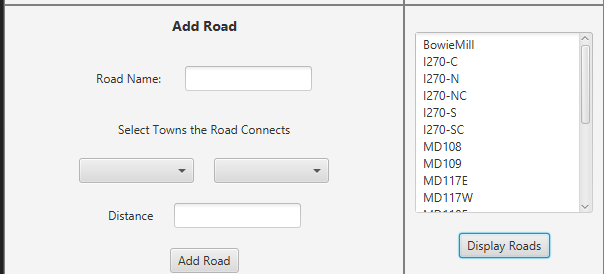
****

****

**Towns read from file:**

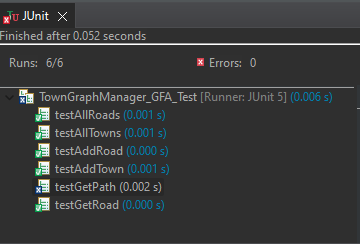
****

**Roads read from a file:**

****

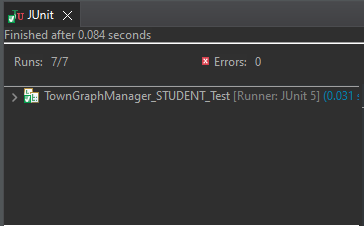
**JUnit Testing**

**TownGraphManager\_GFA\_Test**

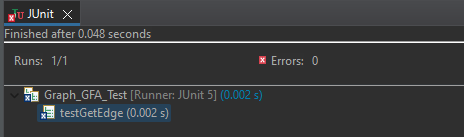
****

* GetPath was not implemented, see explanation above.

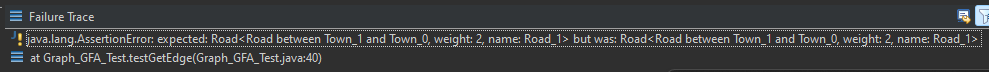
**TownGraphManager\_STUDENT\_Test**

****

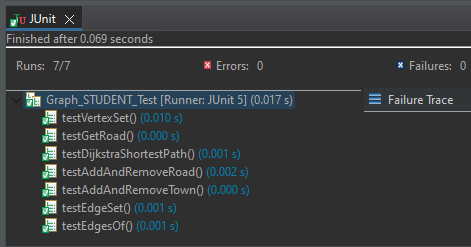
**Graph\_GFA\_Test**

****

* **Test did not pass although the input is correct:**

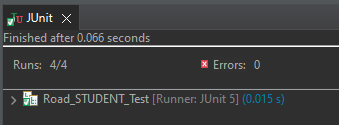
****

**Graph\_STUDENT\_Test**

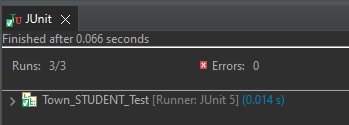
****

* **The dijkstraShortestPath method was not tested due to not being implemented**

**Road\_STUDENT\_Test**

****

**Town\_STUDENT\_Test**

****

**Learning Experiences**

This project was very challenging and I have learned a lot from working on it. Before this project, I thought I already knew how to code graphs, because I understood their concept from our lecture and textbook. I now understand that there is a big difference between understanding a concept, and implementing it in a code. Next project I will make more time to understand that I can code as well as I understand the concepts that I learn.

On the more practical level, I learned how to code a graph using the adjacency list implementation, and how to implement its vertices and edges.

**Assumptions**

1. The user will be using JUnit 5, Java, and JavaFX
2. Files used for this program will be in the same format as the files given to me: roadName,distance;town;town.

**Enhancements**

Added a class called DijkstraShortestPath that I took from the link cited below, the class was also altered by me.

**Citations**

GeeksforGeeks. (2022, October 13). Dijkstra's shortest path algorithm in Java using priorityqueue. GeeksforGeeks. Retrieved May 7, 2023, from https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-in-java-using-priorityqueue/#