Antonio Scalfaro

Project 4 – Undirected Graph GUI

CMSC 315

03/05/2024

This is the Undirected Graph GUI project which is designed to display a GUI that allows vertices to be added to the GUI and then the user can input edges between vertices that are selected. The program consists of 4 classes: Main, Vertex, Graph, and GraphPane. The GUI is composed of a 600 x 400 BorderPane and includes 5 buttons and 3 text fields. The buttons are as follows: Add Edge, Is Connect, Has Cycles, Depth First Search, and Breadth First Search. The two text fields located at the top are used to add the edges between two vertices. The text field at the bottom is used to display output from the button actions. The buttons are connected to their associated functions (i.e Add Edge calls addEdge from the Graph class and creates the edge between the two input vertices). The buttons will display an Error message in the bottom text field if the operation cannot be completed. There is a limitless amount of vertices and edges that can be added, but in reality the number tops out around 52 (alphabet x 2) and edges would be Vertex – 1. The real limit to that is the naming convention, I start by naming A-Z and then move to naming them a-z. Once z is reached, it will continue to increment the char so other characters will display for the next few which is why I state it tops out *around 52* because the number is more but the naming becomes non-alphabetical.

**Test Plan:**

As far as testing goes for this program, it was mostly conducted through the GUI. I tested all of the functionality through the GUI. First, I ensure that all the buttons error out if they cannot complete their intended function and that an error message is displayed in the bottom text field. After that check, I add in vertices (using Primary Mouse Button) and edges to test the button functions. Next, I test that each function being called through the buttons operates as designed and returns the correct output to the bottom text field. I then check to see that the number of vertices that can be added are “limitless” with the added quantifier from the paragraph above. The same goes for the number of edges. I check that an error message is displayed if the user tries to add edges to vertices that are not present.

**Class Diagram:**

****

**Initial GUI:**

A computer screen with a white box

Description automatically generated

**Error Messages from button presses:**

A screenshot of a computer

Description automatically generated

A computer screen with a white screen

Description automatically generated

A computer screen with a white square

Description automatically generated

A computer screen with a white screen

Description automatically generated

A computer screen with a white screen

Description automatically generated

**Add 1 Vertex, Test Buttons:**

A computer screen with a white square

Description automatically generated

A computer screen with a white screen

Description automatically generated

A computer screen with a white square on it

Description automatically generated

A computer screen with a white square

Description automatically generated

**Two Vertex, One Edge Test Connected and has Cycle:**

A computer screen with a white square on it

Description automatically generated

A computer screen with a white square on it

Description automatically generated

**Multiple Vertices and Edges All functions:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Test isConnected (No) & hasCycles (Yes):**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Many Vertices and Edges:**

A screenshot of a computer

Description automatically generated

**What was Learned?**

This program was a lot of fun to create, and I am enjoying growing my JavaFX skill base. I am not a wizard with it just yet but the more I use it the more fluent I become with it. I did not struggle with this program too much up until it came to testing the breadth first search and depth first search. I had mostly coded the algorithms correctly with one fatal mistake, I had used a HashSet for my Set<Vertex> visited variable and when I tried to print out the Set the order would be off. I had come to realize that I needed to use a LinkedHashSet so that I could go through the Set in order. This took a few hours to figure out but eventually everything worked as designed.