**Programming Project 4**

**1. Discussion of Approach**

**Design Decisions**

Program control flow

- Following the instructions given to me, the Main class serves as the entry point and oversees creating the GUI components and handles user interactions. Each button click triggers specific actions, such as adding edges, checking graph properties and performing graph traversal algorithms (DFS and BFS).

Data structures and data types of variables

- I followed the learning material and tried to implement it in my own way. The Graph class utilizes ArrayLists to store vertices and edges. Vertices are represented by the Vertex class. Edges are represented as lists of vertices. Boolean arrays are created for tracking visited vertices during DFS and BFS, ensuring efficient traversal.

Implementing meaningful, friendly, and intuitive user interface

- I tried my best to stay true to the given GUI design. It focuses on simplicity, intuitiveness, and functionality. Input validation checks ensure that users are guided to input correct data, reducing the likelihood of errors. Visual representations of the graph using circles (vertices) and lines (edges) offer users a clear visualization of the graph's structure, aiding in understanding and interaction.

Additional methods, additional variables, and their role

- I created additional methods in the Graph class to perform various graph operations efficiently. Methods for adding vertices and edges, checking graph properties, and DFS and BFS are used to support different functionalities of the program. Additionally, including methods like formatArrayList() helped with formatting the output of DFS and BFS results.

**2. UML Class Diagram**

A screenshot of a computer

Description automatically generated

**3. Test Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **What aspect is tested** | **Input** | **Expected Output** | **Actual Output** | **Pass / Fail** |
| 1. Testing DFS and BFS button with vertices without edges | 3 vertices added to GUI. | DFS displays [A, B, C].  BFS displays [A, B, C]. | Depth First Search Result: [A, B, C]  Breadth First Search Result: [A, B, C] | Pass |
| 2. Testing “Add Edge” button | 3 vertices added to GUI. Added a to d.  Added “ ” to “ ”.  Added a to C. | Error in status text field.  Prompt to enter both vertices.  Line connecting A and C together. | One or both vertices do not exist.  Please enter both vertices.  Line drawn from A to C | Pass |
| 3. Testing “Is Connected?” button for connected and not connected graph | 3 vertices added to GUI and clicked “Is Connected?” Button.  Added edges A to B and B to C and clicked “Is Connected?” button | Graph is not connected.  Graph is connected. | Graph is not connected.  Graph is connected. | Pass |
| 4. Testing DFS and BFS button with binary tree looking graph | 10 vertices added to GUI. Added edges: A to B, C, and D  B to E  E to F  F to G  C to H  D to I and J.  Clicked DFS and BFS button. | DFS displays [A, B, E, F, G, C, H, D, J, I].  BFS displays [A, B, C, D, E, H, J, I, F, G]. | Depth First Search Result: [A, B, E, F, G, C, H, D, J, I]  Depth First Search Result: [A, B, C, D, E, H, J, I, F, G] | Pass |
| 5. Testing “Has Cycles?” button | 3 vertices added to GUI. Added edges: A to B and B to C. Clicked “Has Cycles?” Button.  Added edge: A to C. Clicked “Has cycles?” Button. | Graph doesn’t have cycles.  Graph has cycles. | Graph does not have cycles.  Graph has cycles. | Pass |

**4. Screenshots**

**Test Case 1**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Test Case 2**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Test Case 3**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Test Case 4**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Test Case 5**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**5. Lessons Learned**

This project helped increase my proficiency with JavaFX for GUI development. Being able to work with events gave me the chance to write shorthanded code with event handlers. During this project I was able to debug in a more efficient way by adding printing statements to help me locate bugs. I realized that it was difficult to debug a GUI, so I had to figure out a way to adapt. Additionally, I was able to apply lessons on graphs. Being able to see and create a program with a better grasp of algorithms and data structures really helped me try to make my code more efficient. In conclusion, I learned how to work more efficiently with JavaFX and how to debug in a more efficient way.