/\*\*

\* This is the main class that will implement a GUI application showing the

\* results of recompiling a directed graph in topological order.

\*

\* Name: Nabeel Hussain

\* Class: CMSC 350

\* Professor: Didier Vergamini

\* Project 4

\* Date: 12/16/2016

\*

\* **@author** Nabeel Hussain

\*/

This program for project 4 involves implementing a directed graph, and then recompiling the vertices of the graph in topological order, using a depth-first search. The graph will be built using a file that is read in containing the dependency information of the vertices. The sorted result is shown using a GUI. The program contains a button to build the graph, and a button to perform the topological sort.

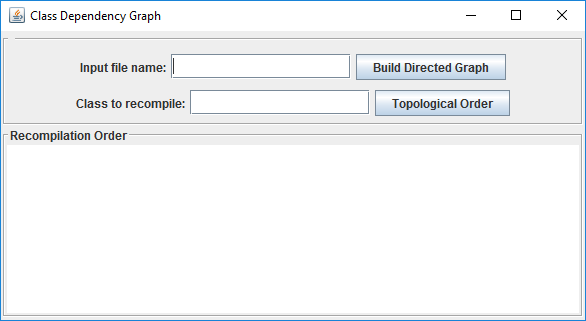
My program has **Three classes:**

The first class is the Stack data structure, which is a generic class. It will be used in the depthFirstSearch method to store the vertex indices associated with their hash map location, when performing the topological sort through a dfs.

The second class is named DirectedGraph.java, which is a generic class that builds the directed graph of the vertices, using the file of dependency information read in. It also holds the methods that will be used to add edges into the adjacency list of a specific vertex, and another to perform a dfs topological sort. To recompile a vertex in the graph in topological order, there must be a starting vertex specified, and then it will use the pseudo code algorithm provided in the instructions to store its edge vertices in reverse topological order using a stack. The stack will then be popped to display the recompilation of the vertex in the correct order.

The third class is used for the GUI, which will contain the buttons, text fields, and a text area output to meet the functional requirements for this assignment. It will properly extract the information from the file name entered by a user, which will then be used by the DirectedGraph class to build the graph and perform a topological sort.

**My GUI:**



**Test Plan:**

**graph.txt**:

ClassA ClassC ClassE

ClassB ClassD ClassG

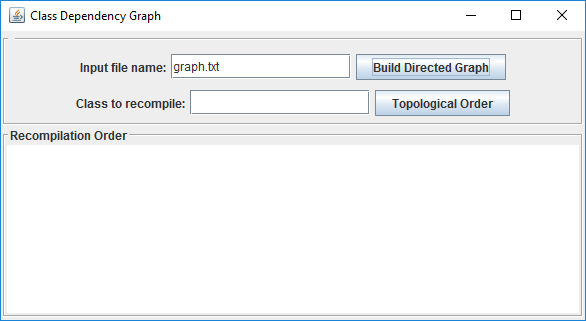
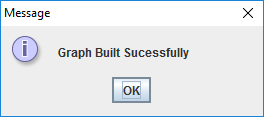
ClassE ClassB ClassF ClassH

ClassI ClassC

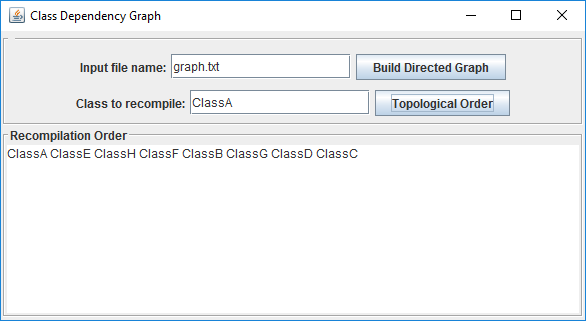
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output**  Sorted List: | **Did Test Pass?** |
| 1 | **Input file name:** graph.txt  **\***Check if graph gets built successfully\* | JOption Pane Show Message Dialog:  "Graph Built Successfully " | Y |
| 2 | **Input file name:** graph.txt  **Class to recompile:** ClassA | Recompilation Order:  ClassA ClassE ClassH ClassF ClassB ClassG ClassD ClassC | Y |
| 3 | **Input file name:** graph.txt  **Class to recompile:** ClassB | Recompilation Order:  ClassB ClassG ClassD | Y |
| 4 | **Input file name:** graph.txt  **Class to recompile:** ClassE | Recompilation Order:  ClassE ClassH ClassF ClassB ClassG ClassD | Y |
| 5 | **Input file name:** graph.txt  **Class to recompile:** ClassI | Recompilation Order:  ClassI ClassC | Y |
| 6 | **Input file name:** nofile.txt  **\***Check message notifying the user that a file could not be found\* | JOption Pane Show Message Dialog:  "File Did Not Open " | Y |
| 7 | **Input file name:** cycle.txt  **Class to recompile:** ClassE  **\***Check if a cycle is detected in the graph, and display message\* | JOption Pane Show Message Dialog:  "Cycle Detected " | Y |
| 8 | **Input file name:** graph.txt  **Class to recompile:** ClassZ  **\***Displays error message if an invalid class name is entered to be recompiled. \* | JOption Pane Show Message Dialog:  "Invalid Class Name " | Y |

**Screen shots of successful compilation and running for all test cases**

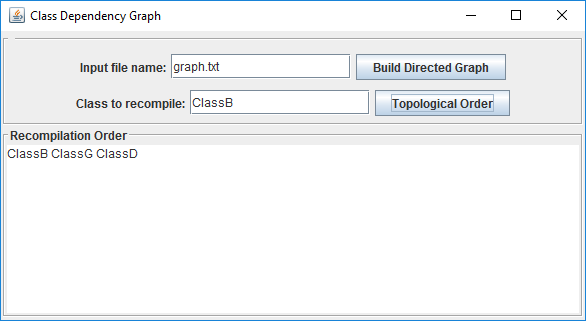
**Test Case 1:** Graph Built Successfully



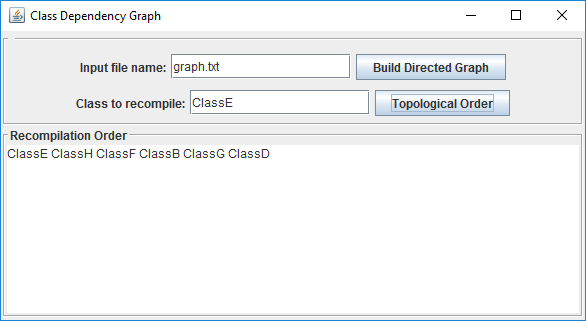
**Test Case 2:** Topological sort with source vertex of ClassA



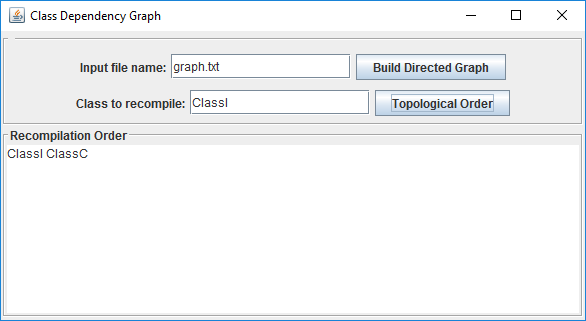
**Test Case 3:** Topological sort with source vertex of ClassB



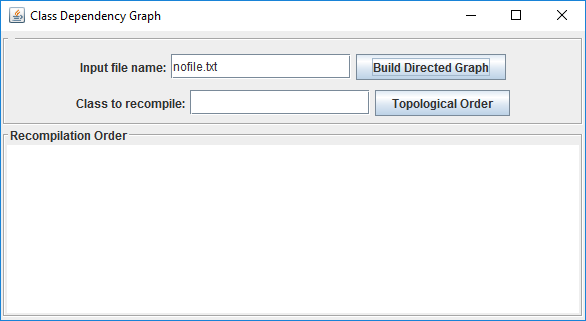
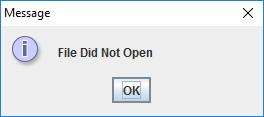
**Test Case 4:** Topological sort with source vertex of ClassE



**Test Case 5:** Topological sort with source vertex of ClassI



**Test Case 6:** File could not be found



**Test Case 7:** If a cycle is detected in the graph

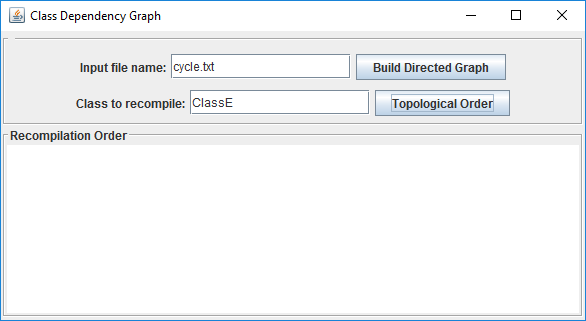
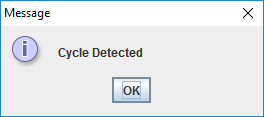
**Cycle.txt:**

ClassA ClassC ClassE

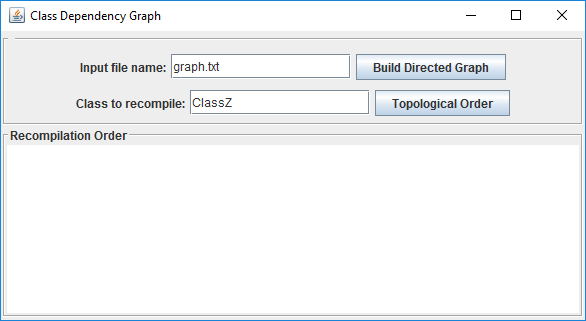
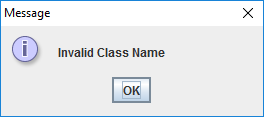
ClassB ClassD ClassG

ClassE ClassB ClassF ClassH ClassA

ClassI ClassC



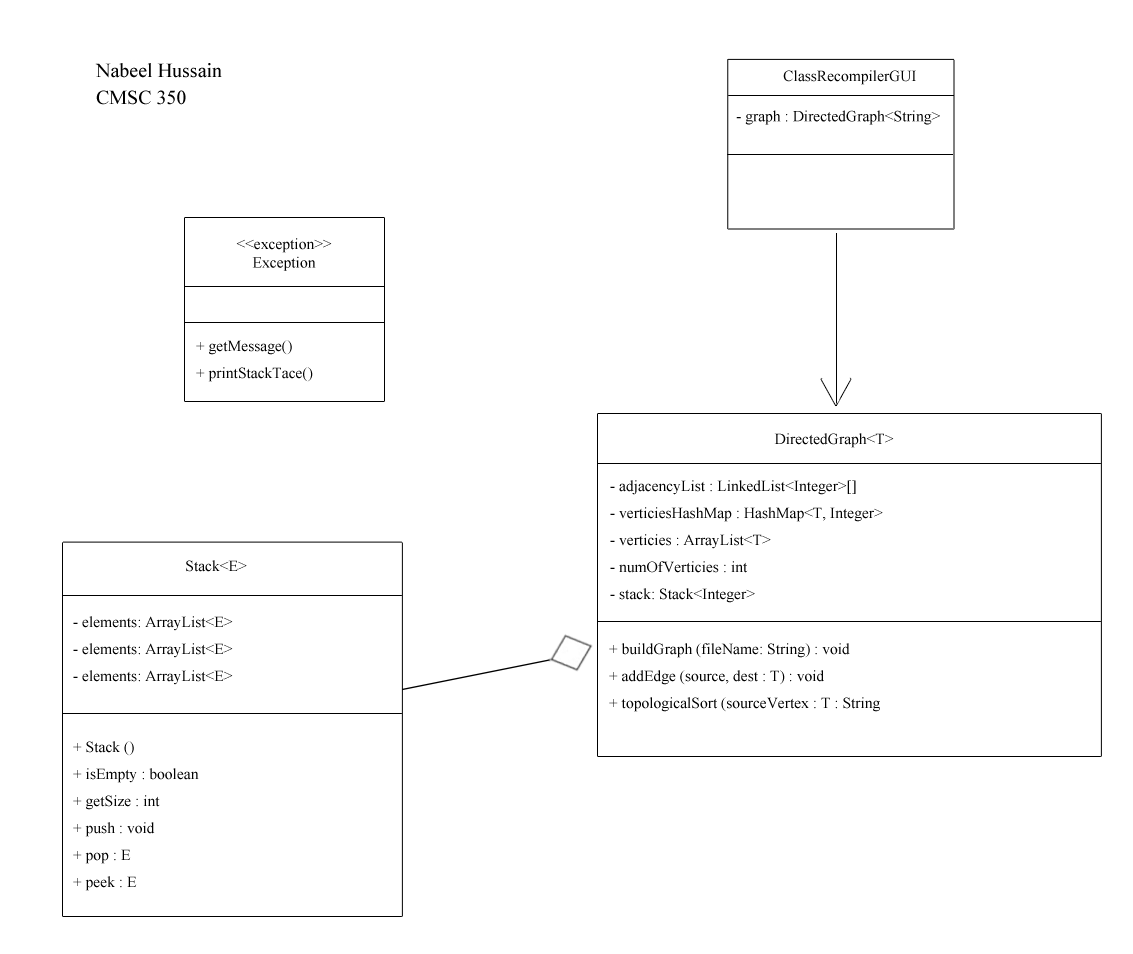
**Test Case 8:** If an invalid class name is entered to be recompiled.



**Lessons Learned:**

While working on Project 4, there were many new concepts and techniques that I learned about building graphs, such as how to go about adding new edges to them, and performing depth-first traversals on them. I also learned about topological ordering, and how to sort vertices in a graph this way, using the dfs traversal algorithm. Graphs are a superset of the tree data structure, so there were many similarities to it, just with new terminology.

This assignment was challenging, but it definitely helped me understand and reinforce all the material I learned from this week’s module much better. Overall, I found this class to be very fast paced and intellectually stimulating, as I learned a lot of new things during the duration of this course. I definitely feel as though I am a better programmer now than before I started this class, which is what I hoped for. Thank you, and happy holidays!

**UML Diagram:**