

UML Diagram:

For this UML diagram I followed a sample one in an Introduction to Java Book:

MainGUI Class

```
+ public class MainGUI extends JFrame -> extends the JFrame Swing
+ public MainGUI () -> constructor for GUI
+ private class ButtonActionListener implements ActionListener -> registers button
+ public void actionPerformed (ActionEvent e) -> performs action
+ public static void main (String [] args) -> main method to execute program
```



DirectedGraph Class

```
+ public class DirectedGraph -> class name
+ Graph (String file) throws FileNotFoundException -> reads in file
+ private Vertex isVertexPresent (String in) -> sees if vertex is present
+ private void printHashMap () -> prints a Hash Map
+ private void depthFirstSearch (int v, boolean [] isVisited) throws CycleDetectedException
```



Neighbor Class

```
+ public class Neighbor -> Neighbor class
+ public Neighbor (int vertexNum, Neighbor neighbor) -> Neighbor class takes two parameters
+ public boolean isPresent (int vertex) -> isPresent method takes an integer
```



Vertex Class

```
+ public class Vertex -> Vertex Class
+ Vertex (String name) -> Vertex constructor takes name as a string
+ public boolean isPresent (int vertex) -> isPresent takes vertex as an integer
+ public void addNeighbor (int vertex) -> addNeighbor takes vertex as an integer
```



CycleDectedException Class

```
+ public class CycleDetectedException extends Exception    -> CycleDetectedException Class
+ public CycleDetectedException ()    -> CycleDetectedException method
```

Test Cases:

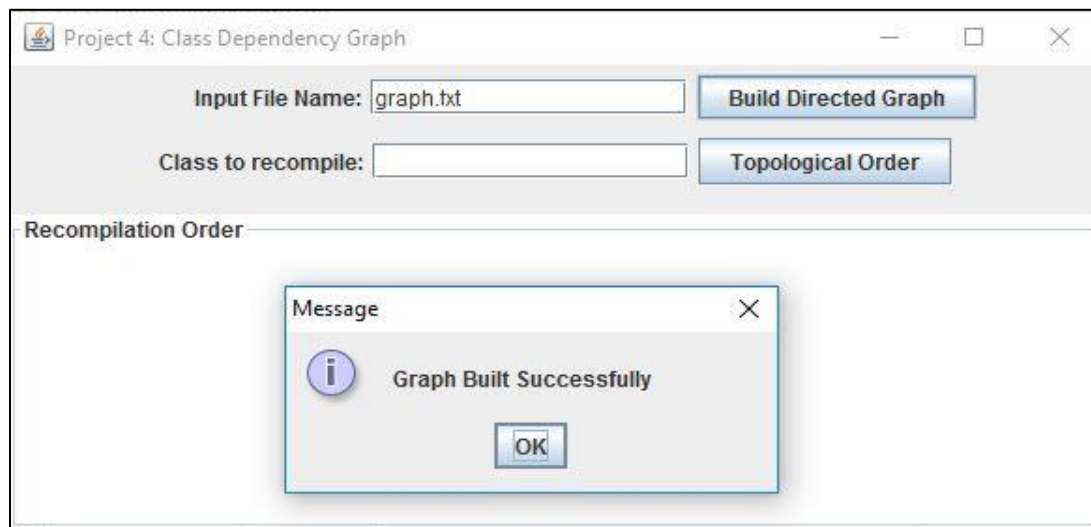
GUI Display:

The following File will be used for all test cases: graph.txt

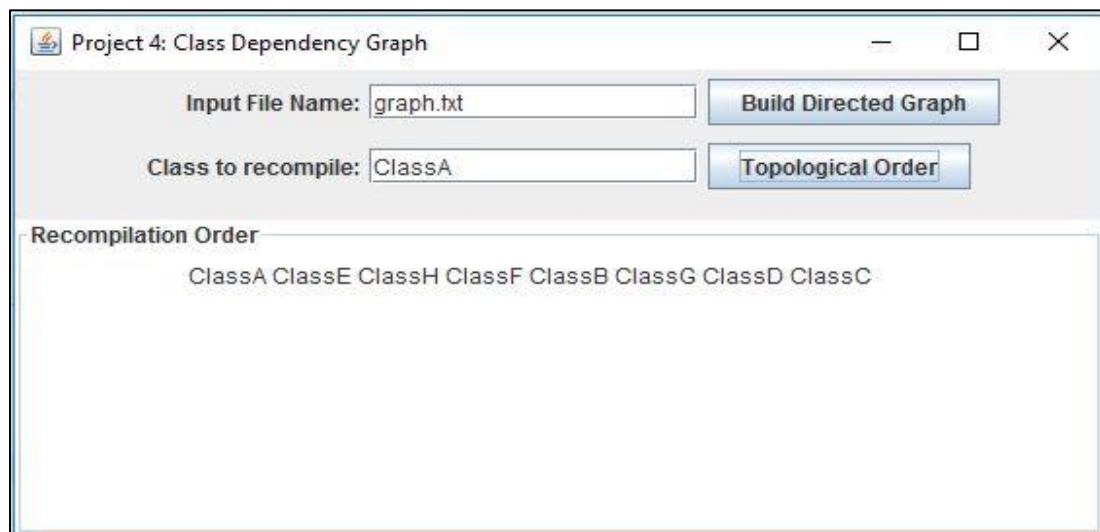
1	ClassA ClassC ClassE
2	ClassB ClassD ClassG
3	ClassE ClassB ClassF ClassH
4	ClassI ClassC

Test Case 1:

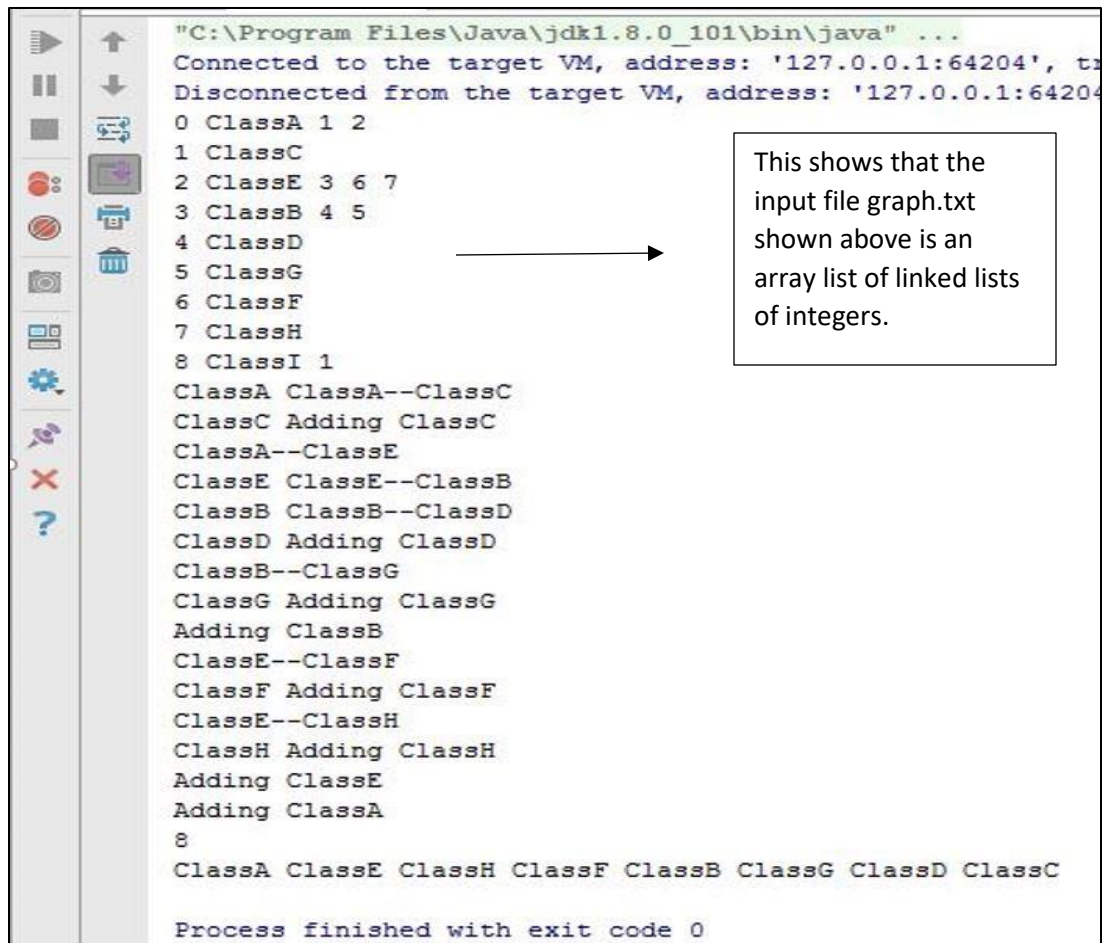
- Enter in graph.txt
- Press Build Direct Graph Button
- Message Graph Built Successfully



- Enter in ClassA
- Press the Topological Order Button
- Displays Recompile Order:
 - ClassA ClassE ClassH ClassF ClassB ClassG ClassD ClassC



Console Output:



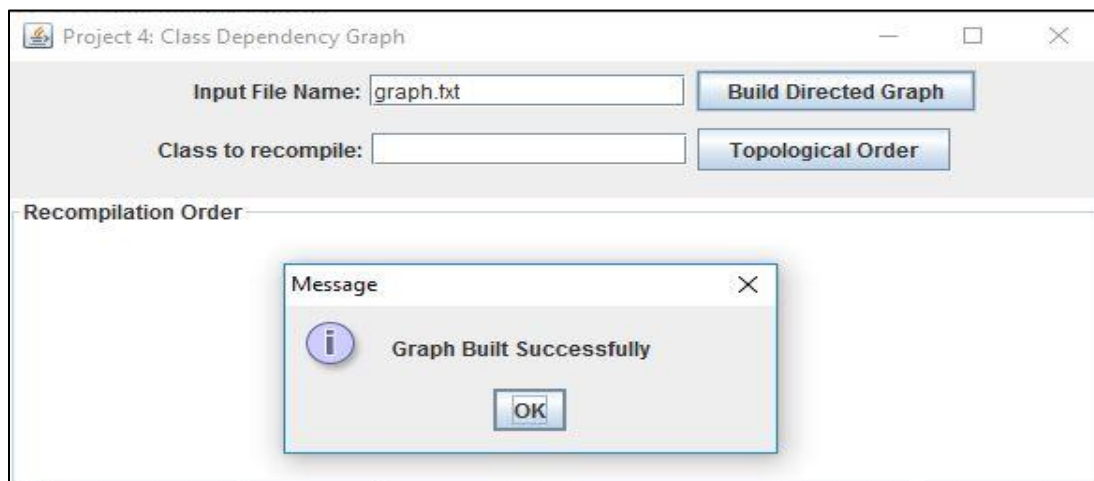
```
"C:\Program Files\Java\jdk1.8.0_101\bin\java" ...
Connected to the target VM, address: '127.0.0.1:64204', t
Disconnected from the target VM, address: '127.0.0.1:64204'
0 ClassA 1 2
1 ClassC
2 ClassE 3 6 7
3 ClassB 4 5
4 ClassD
5 ClassG
6 ClassF
7 ClassH
8 ClassI 1
ClassA ClassA--ClassC
ClassC Adding ClassC
ClassA--ClassE
ClassE ClassE--ClassB
ClassB ClassB--ClassD
ClassD Adding ClassD
ClassB--ClassG
ClassG Adding ClassG
Adding ClassB
ClassE--ClassF
ClassF Adding ClassF
ClassE--ClassH
ClassH Adding ClassH
Adding ClassE
Adding ClassA
8
ClassA ClassE ClassH ClassF ClassB ClassG ClassD ClassC

Process finished with exit code 0
```

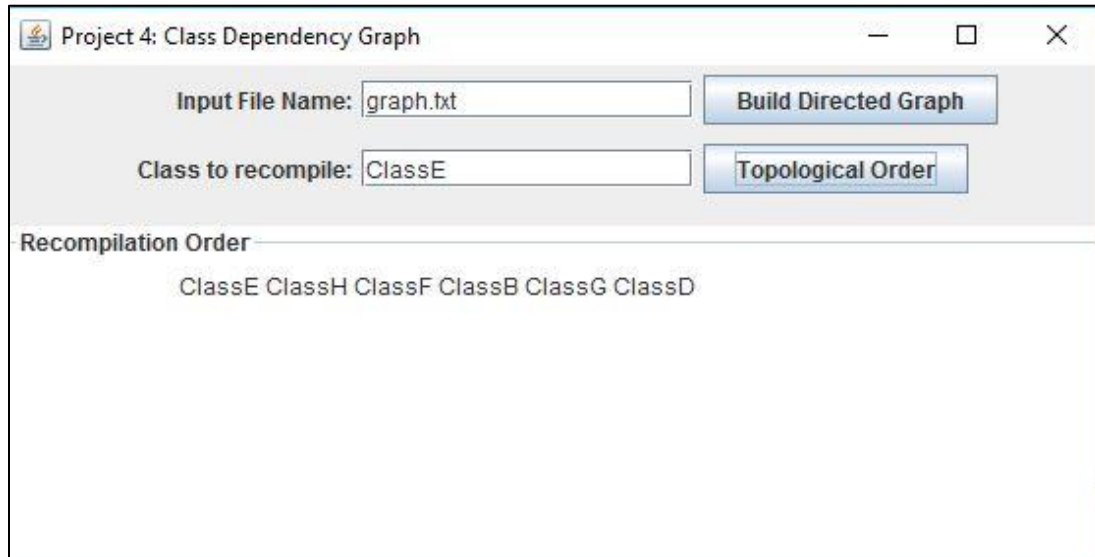
This shows that the input file graph.txt shown above is an array list of linked lists of integers.

Test Case 2:

- Enter in graph.txt
- Press Build Direct Graph Button
- Message Graph Built Successfully

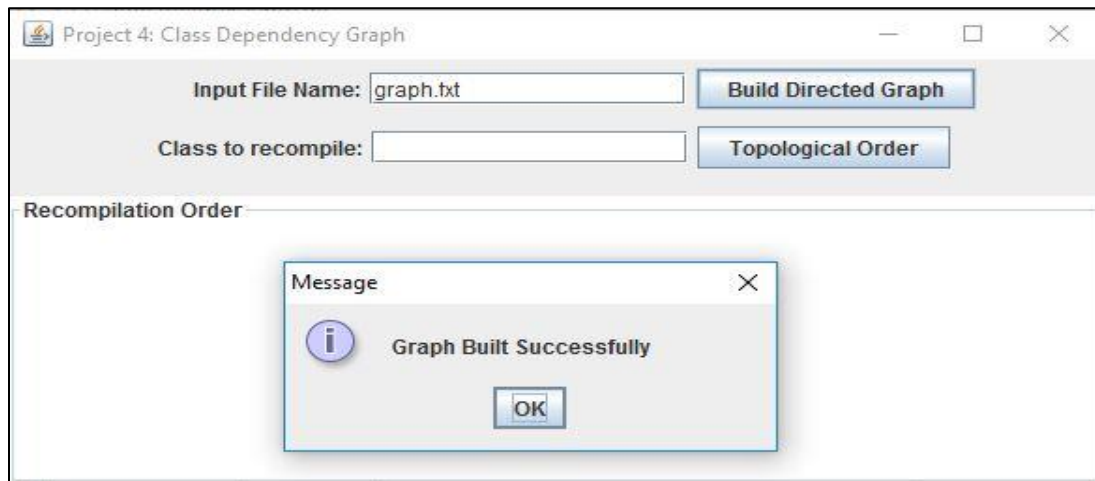


- Enter in ClassE
- Press the Topological Order Button
- Displays Recompile Order:
 - ClassE ClassH ClassF ClassB ClassG ClassD

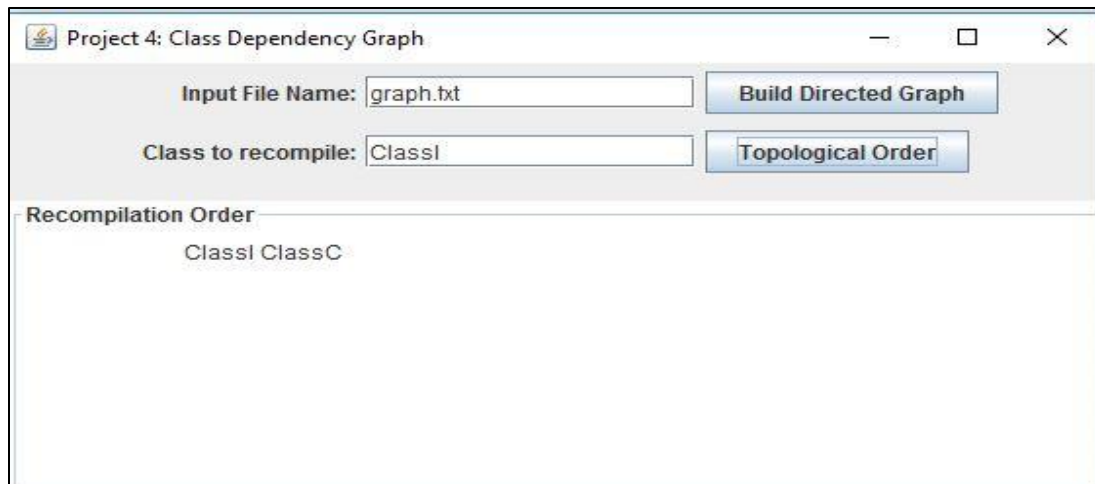


Test Case 3:

- Enter in graph.txt
- Press Build Direct Graph Button
- Message Graph Built Successfully



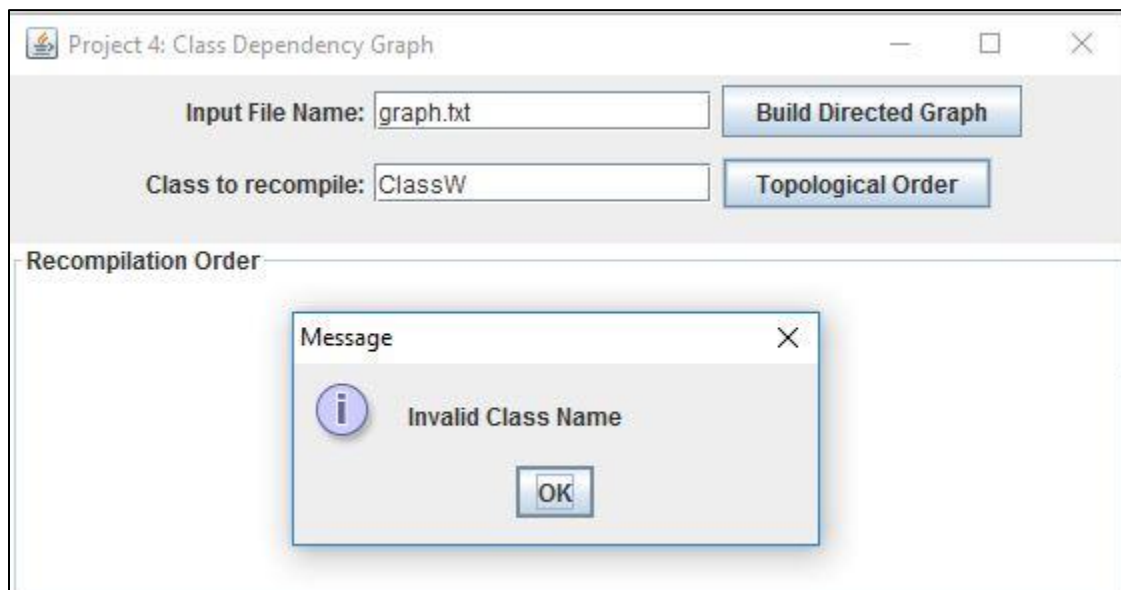
- Enter in ClassI
- Press the Topological Order Button
- Displays Recompile Order:
 - ClassI ClassC



Exceptions:

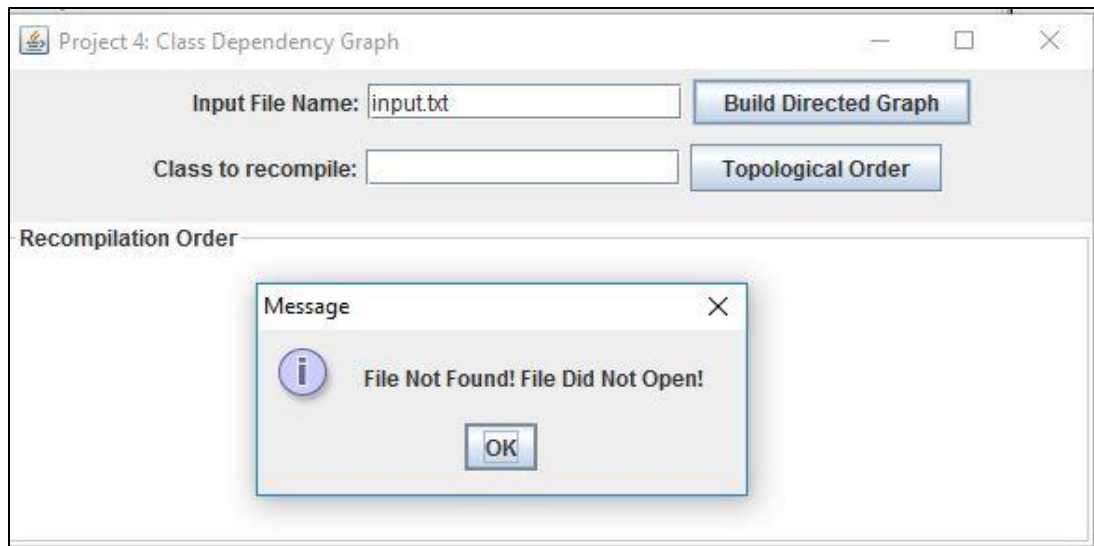
Invalid Class Name:

- Enter in ClassW
- Press the Topological Order Button
- Invalid Class Name Message



File Not Found, and File Did Not Open:

- Type in input.txt
- Press Build Directed Graph Button
- Displays error message: File Not Found! File Did Not Open!



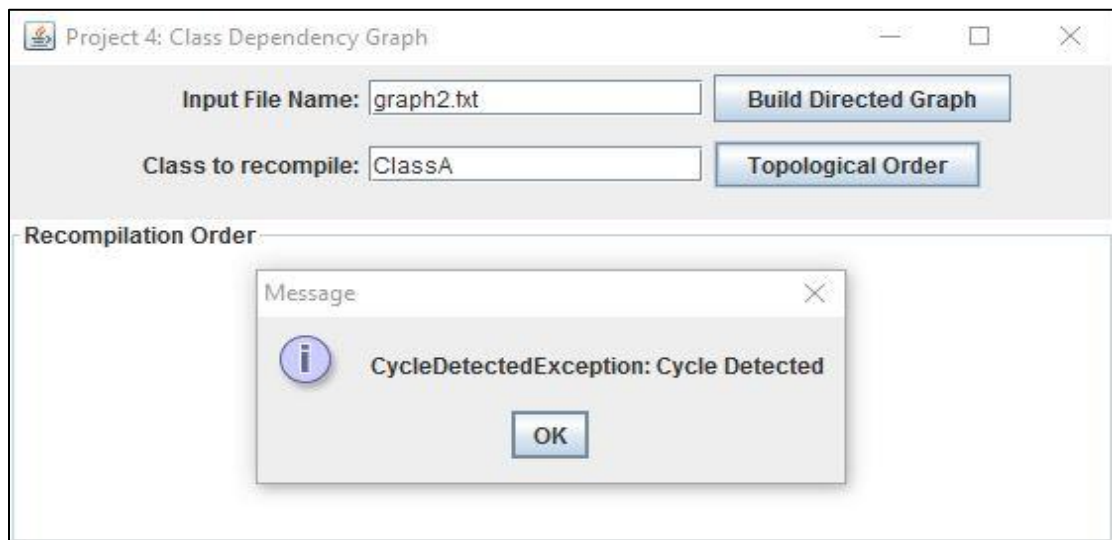
Cycle Detection Exception: reading in graph2.txt which is used to test exception

```
1 ClassA ClassC ClassE ClassA
2 ClassB ClassD ClassG
3 ClassE ClassB ClassF ClassH
4 ClassI ClassC
```

→

Notice how **ClassA** is listed twice in the file graph2.txt

- I read a file in named graph2.txt, that has ClassA listed twice in the file
- This detects that there was a cycle, which means duplicate value



Lessons Learned:

After completing this project there are many things that were learned. I realize how important Graphs, Directed Graphs, Hash Maps, Depth First Search, and Topology Sort algorithms are to data structures. What is most important, like in previous projects, understanding what the problem is asking, and how to implement a solution to solve the problem only comes through trial and error. The hardest part of this project was getting the topology sort to work. Also, implementing the Hash Map was difficult since you had to construct your methods to fit around it. However, I did realize that with this project implementing generics was unnecessary since the output is always going to be a string; therefore, it did not make sense to do it. The easiest part once everything else was working properly, was developing the GUI and then calling the classes. All in all, it took me the full two weeks we had to complete this project, but I am satisfied with my work.