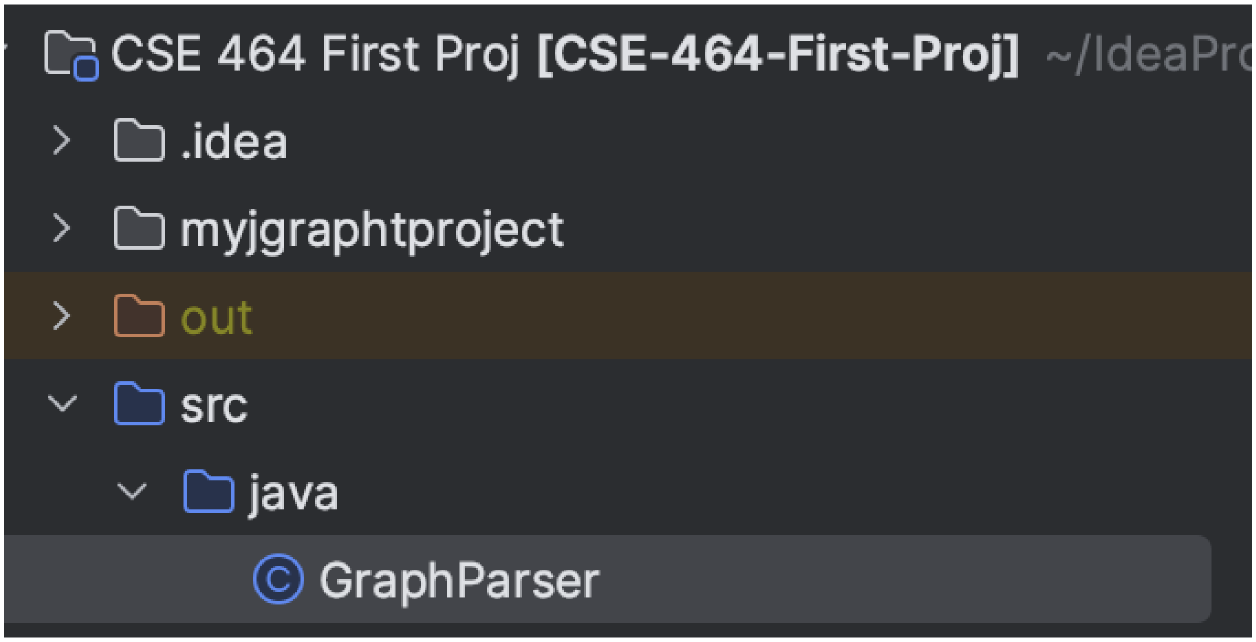
Course Project Part 1

Gabriel Valenzuela
Jgvalen2

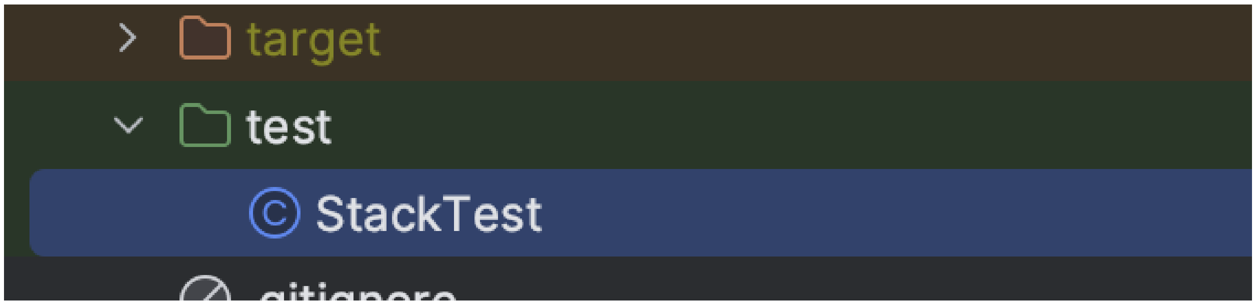
Building the Project

To compile the code using Maven, run the following command:

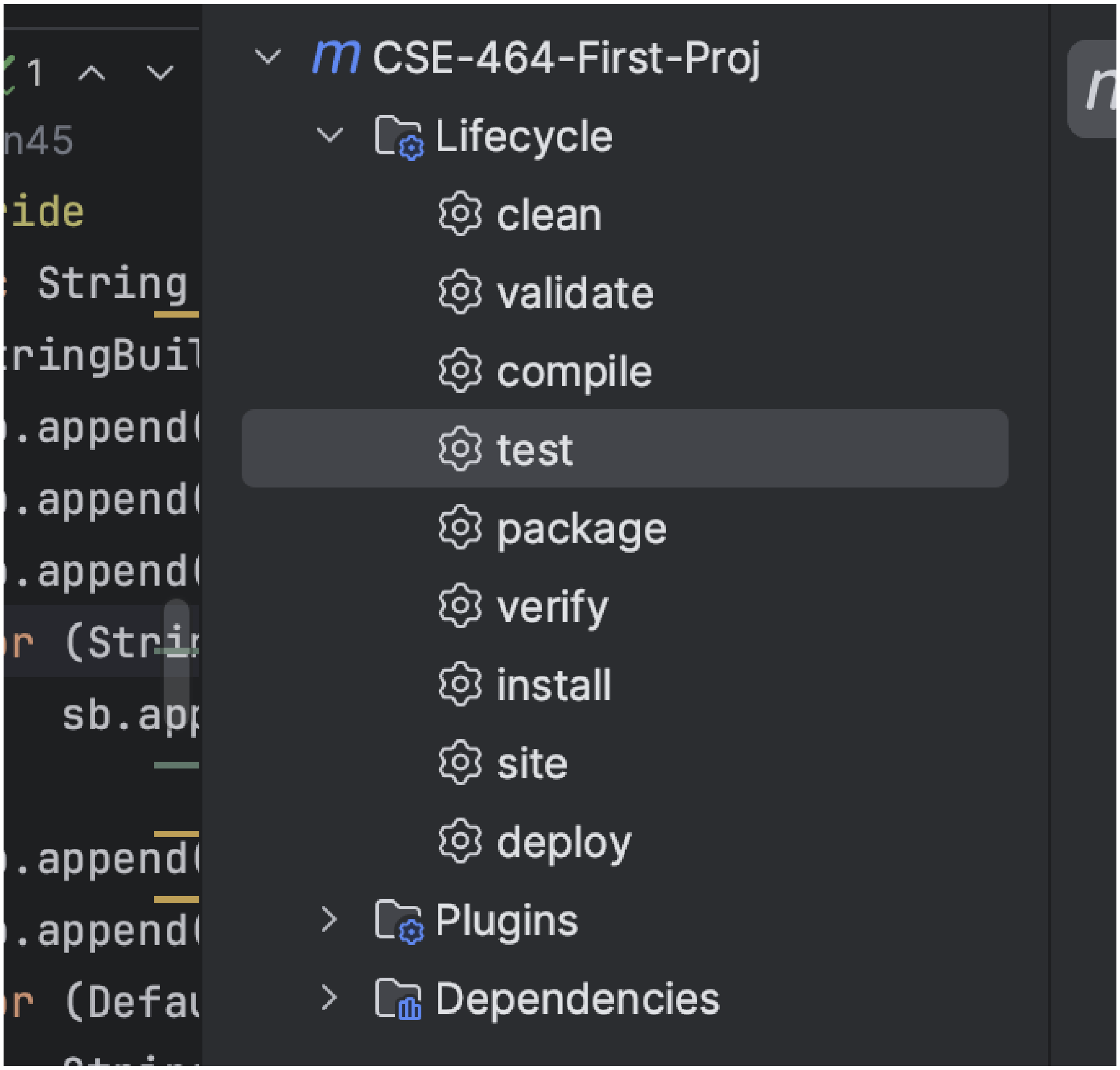
```
mvn package
```



`The class GraphParesr is my main code and runs to confirm the working of features 1-4.`



My test are in the StackTest.



Run Project with Maven, install -> clean -> run

Expected Output for Each Feature

Feature 1: Parse a DOT graph file to create a graph

This feature allows the user to parse a graph from a DOT graph file, generating an internal graph representation.

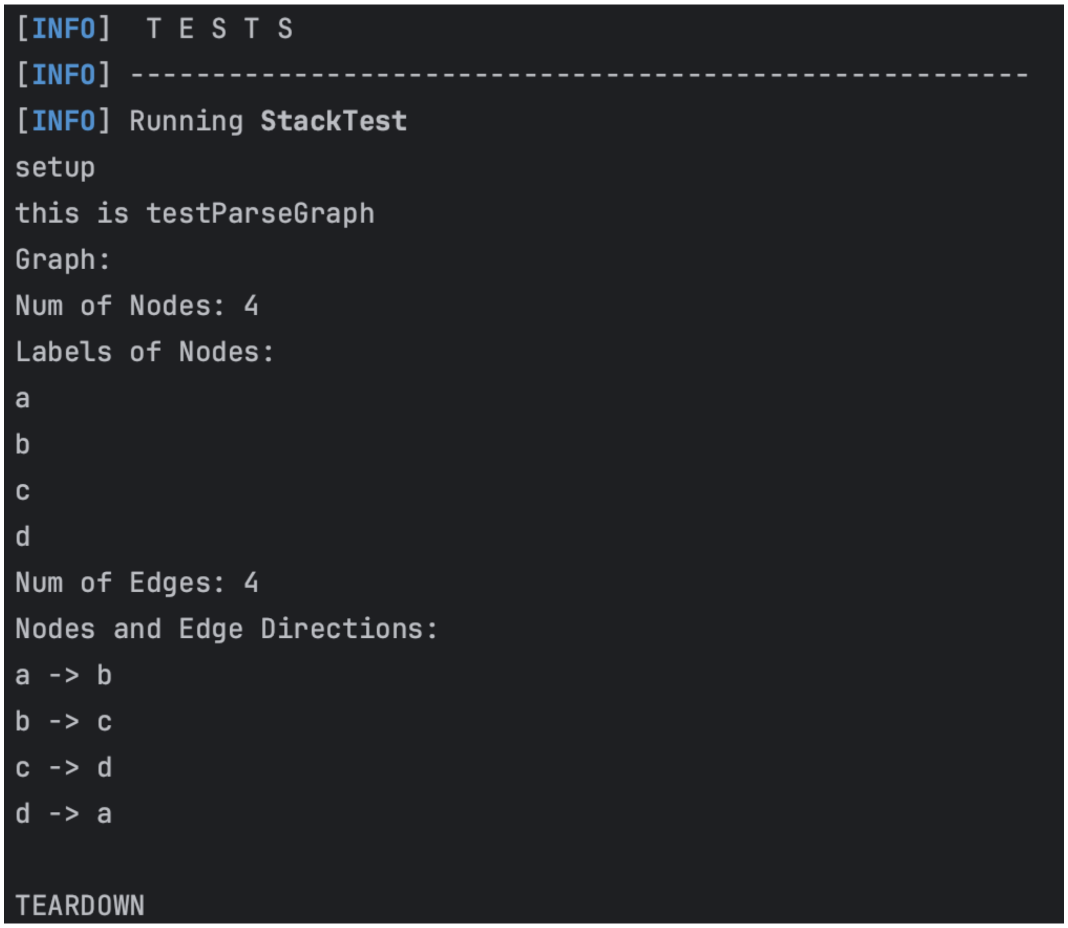
```
public boolean parseGraph(String filepath) {
    if (filepath == null || filepath.isEmpty()) {
        return false;
    }
    File fpath = new File(filepath);
    if (!fpath.exists()) {
        return false;
    }
    DOTImporter<String, DefaultEdge> myImporter = new DOTImporter<>();
    myImporter.setVertexFactory(id -> id);
    myImporter.importGraph(graph, fpath);
    return true;
}
```

Junit Test

```
@Test
public void testParseGraph() throws IOException {
   System.out.println("this is testParseGraph");

   s.parseGraph(filePath);

   // Assuming you've implemented the 'getNumNodes' and 'getNumEdges' methods
   assertEquals(4, s.getNumNodes());
   //assertEquals(4, s.getNumEdges());

   // To check if the graph has been parsed correctly, you can print it or
check specific nodes and edges
   System.out.println(s.toString());

}
```

Junit Expected Test

Testng Parse Graph method. This parses the graph and prints the graph to log to confirm Test.



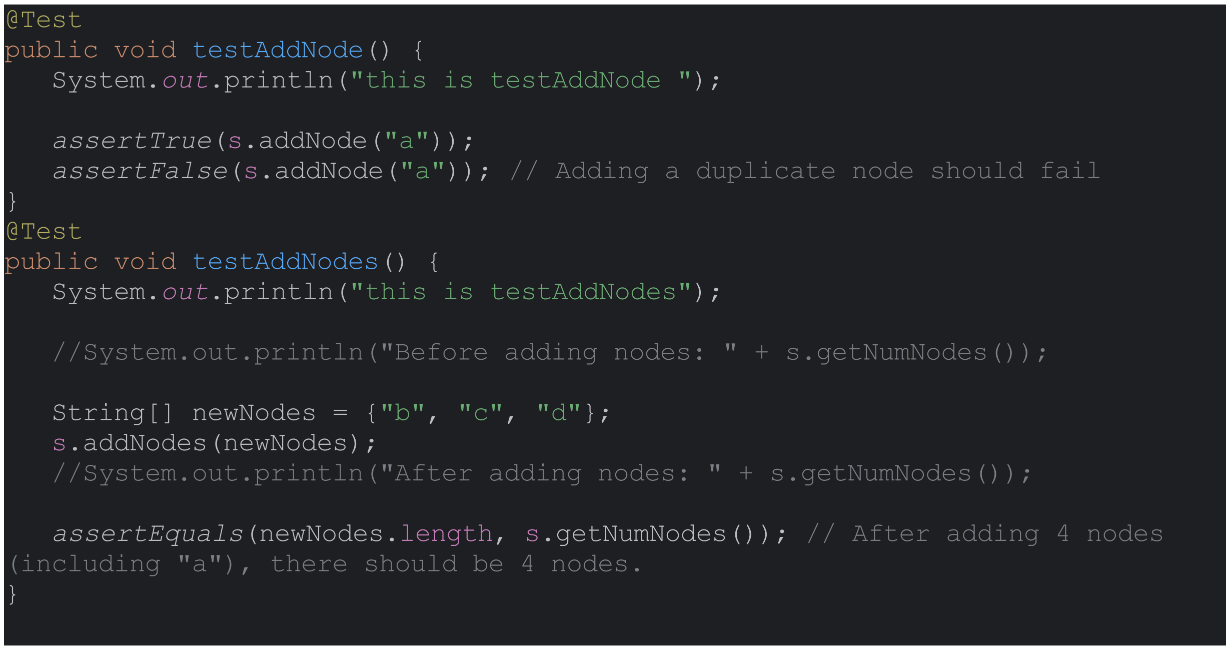
```
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running StackTest
setup
this is testParseGraph
Graph:
Num of Nodes: 4
Labels of Nodes:
a
b
c
d
Num of Edges: 4
Nodes and Edge Directions:
a -> b
b -> c
c -> d
d -> a

TEARDOWN
```

Feature 2: Adding nodes to the imported graph

This feature provides the capability to add individual nodes as well as multiple nodes to the existing graph. It ensures that there are no duplicate node labels

```
public boolean addNode(String label) {
    if (graph.containsVertex(label)) {
        System.out.println("Node " + label + " is a duplicate. Cannot add node");
        return false;
    }
    return graph.addVertex(label);
}

public void addNodes(String[] labels) {
    for (String label : labels) {
        addNode(label);
    }
}
public void addNodes(String[] labels) {
   for (String label : labels) {
       addNode(label);
   }
}
```

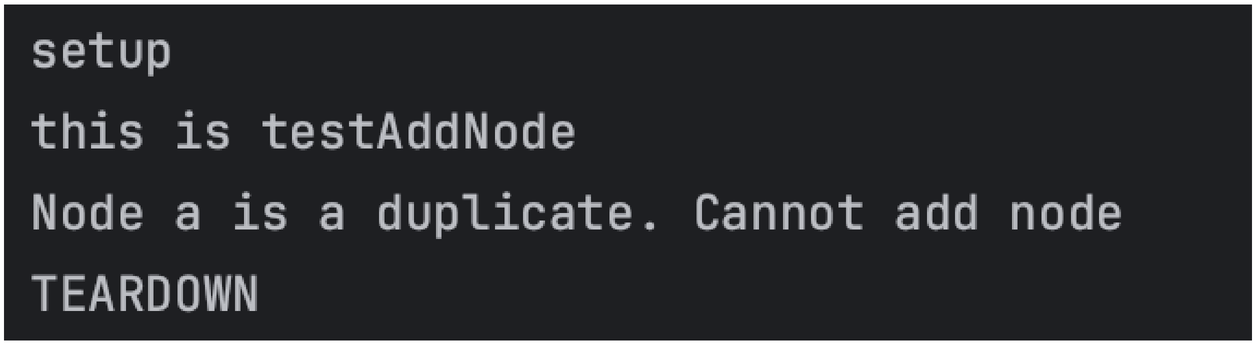
Junit Test



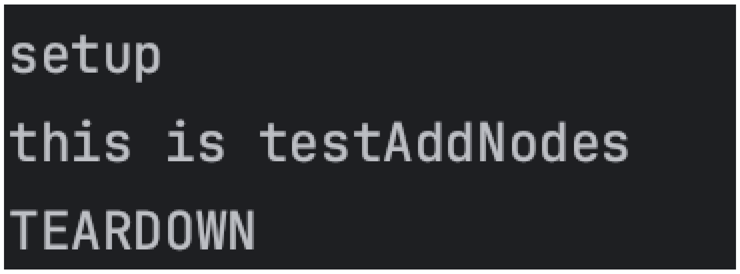
```
@Test
public void testAddNode() {
   System.out.println("this is testAddNode ");

   assertTrue(s.addNode("a"));
   assertFalse(s.addNode("a")); // Adding a duplicate node should fail
}
@Test
public void testAddNodes() {
   System.out.println("this is testAddNodes");

   //System.out.println("Before adding nodes: " + s.getNumNodes());

   String[] newNodes = {"b", "c", "d"};
   s.addNodes(newNodes);
   //System.out.println("After adding nodes: " + s.getNumNodes());

   assertEquals(newNodes.length, s.getNumNodes()); // After adding 4 nodes
(including "a"), there should be 4 nodes.
}
```

Junit Expected Test

Adding a duplicate edge should fail



```
setup
this is testAddNode
Node a is a duplicate. Cannot add node
TEARDOWN
```



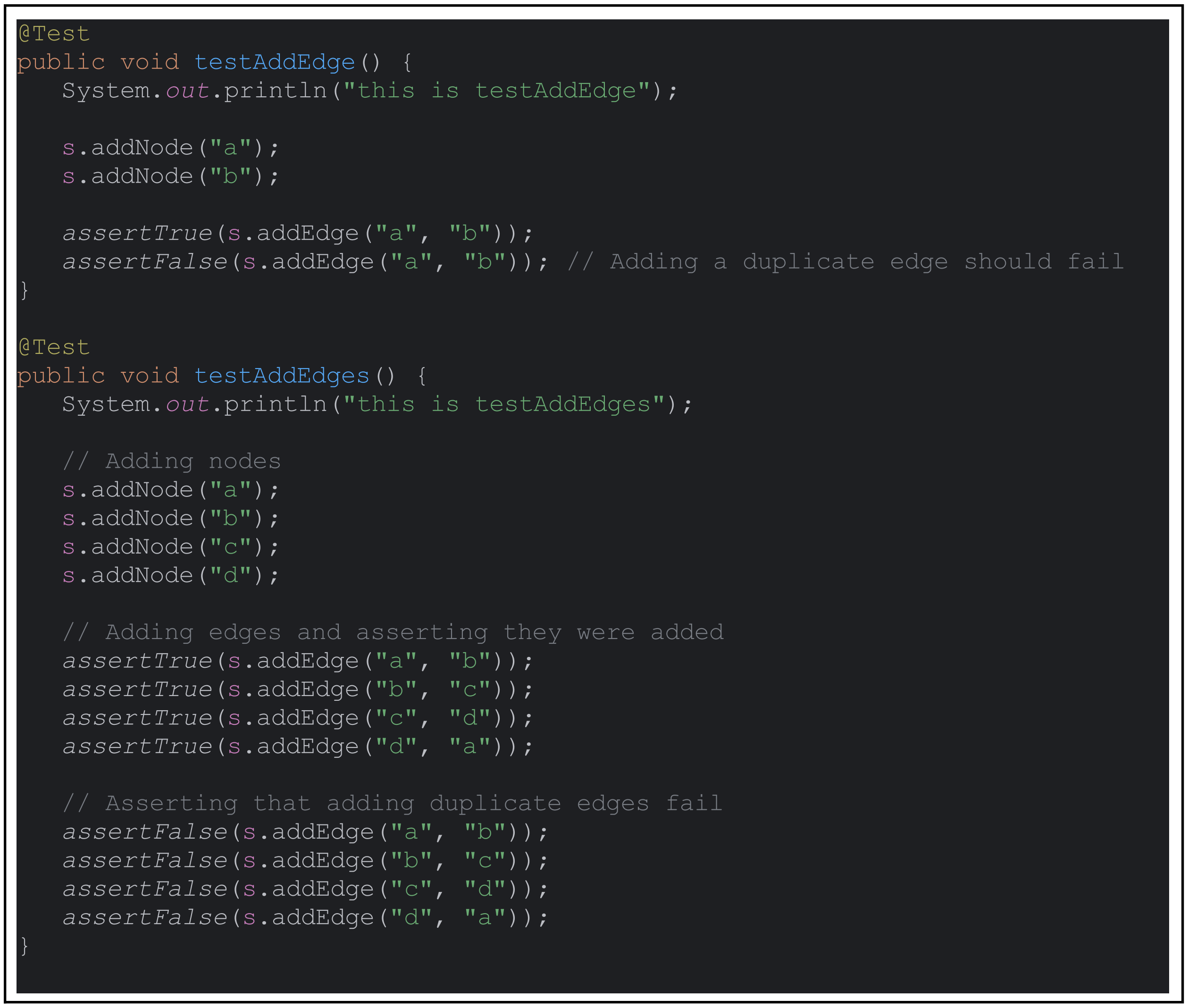
```
setup
this is testAddNodes
TEARDOWN
```

Feature 3: Adding edges to the imported graph

This feature enables the user to add directed edges between nodes in the graph. The feature ensures that there are no duplicate edges.

```
public boolean addEdge(String srcLabel, String dstLabel) {
    if (graph.containsEdge(srcLabel, dstLabel)) {
        System.out.println("Duplicate edge from: " + srcLabel + "->" + dstLabel);
        return false;
    }
    try {
        graph.addEdge(srcLabel, dstLabel);
    } catch (Exception ex) {
        System.out.println("Exception while adding edge " + ex.getMessage());
        return false;
    }
    return true;
}
```

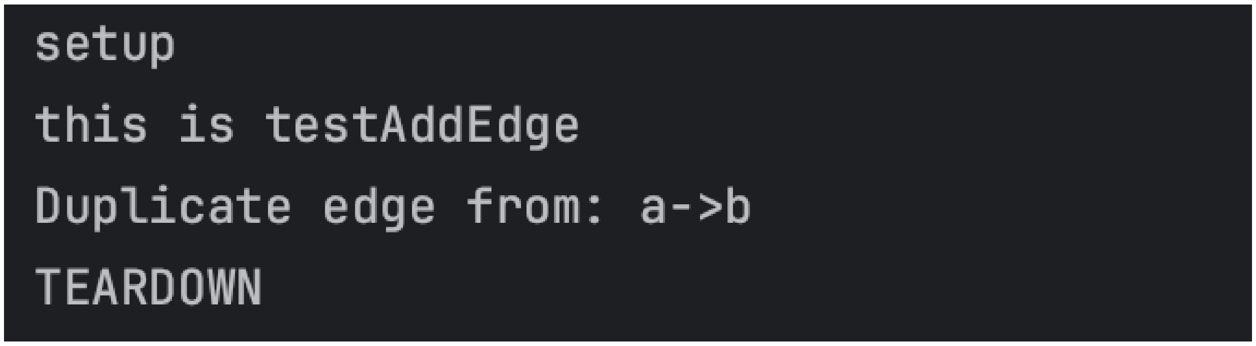
Junit Test



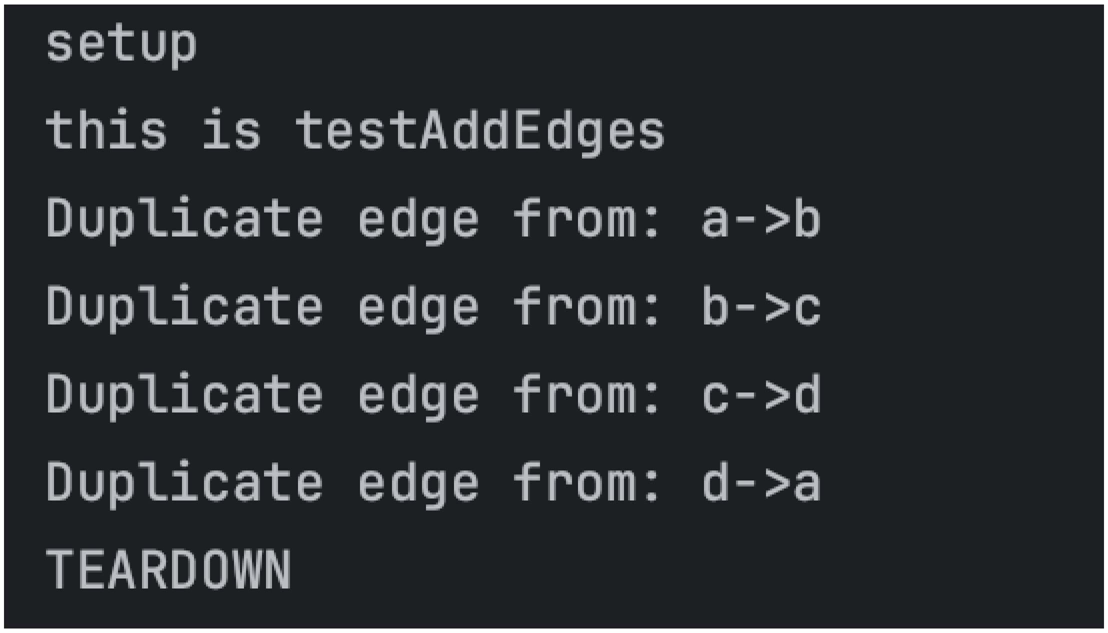
```
@Test
public void testAddEdge() {
   System.out.println("this is testAddEdge");

   s.addNode("a");
   s.addNode("b");

   assertTrue(s.addEdge("a", "b"));
   assertFalse(s.addEdge("a", "b")); // Adding a duplicate edge should fail
}

@Test
public void testAddEdges() {
   System.out.println("this is testAddEdges");

   // Adding nodes
   s.addNode("a");
   s.addNode("b");
   s.addNode("c");
   s.addNode("d");

   // Adding edges and asserting they were added
   assertTrue(s.addEdge("a", "b"));
   assertTrue(s.addEdge("b", "c"));
   assertTrue(s.addEdge("c", "d"));
   assertTrue(s.addEdge("d", "a"));

   // Asserting that adding duplicate edges fail
   assertFalse(s.addEdge("a", "b"));
   assertFalse(s.addEdge("b", "c"));
   assertFalse(s.addEdge("c", "d"));
   assertFalse(s.addEdge("d", "a"));
}
```

Junit Expected Test

This Junit test test the add Edge method. It adds a edge and then performs a duplicate edges add test.



```
setup
this is testAddEdge
Duplicate edge from: a->b
TEARDOWN
```



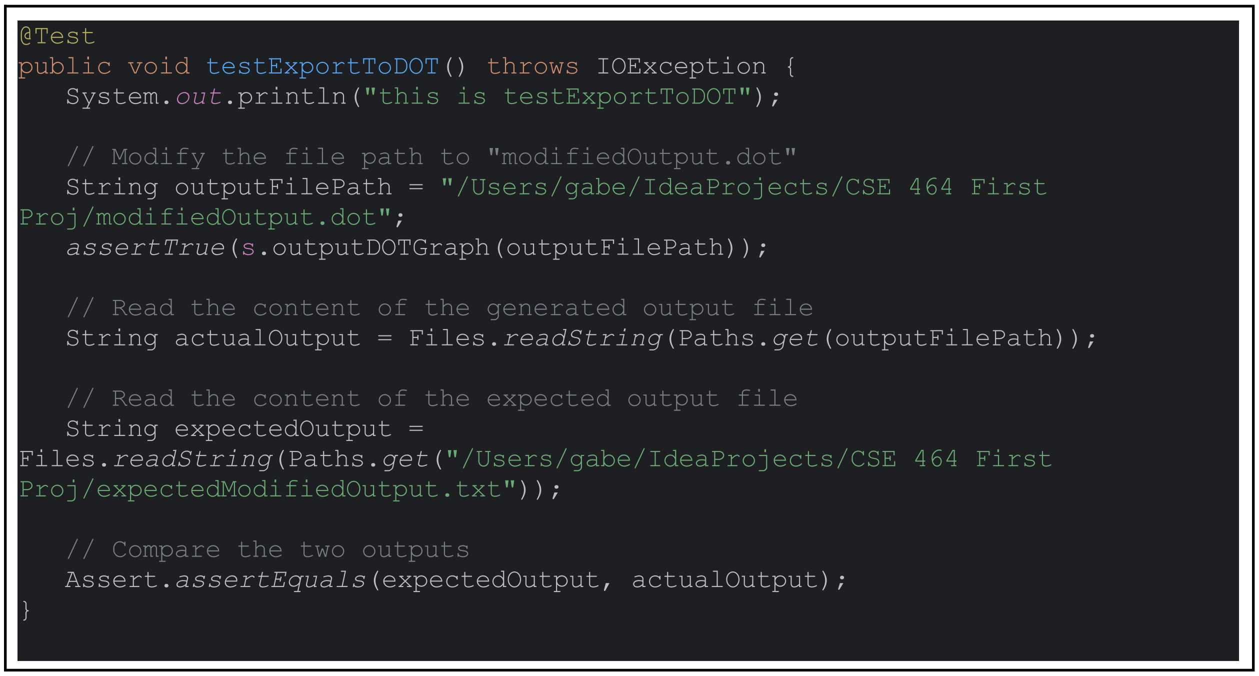
```
setup
this is testAddEdges
Duplicate edge from: a->b
Duplicate edge from: b->c
Duplicate edge from: c->d
Duplicate edge from: d->a
TEARDOWN
```

Feature 4: Output the imported graph into a DOT file or graphics

This feature facilitates the exportation of the internal graph representation back into a DOT file.

```
public boolean outputDOTGraph(String filepath) {
    if (graph == null) {
        System.out.println("Graph is empty");
        return false;
    }
    File outputFile = new File(filepath);
    try {
        DOTExporter<String, DefaultEdge> exporter = new DOTExporter<>(v -> v.toString());
        exporter.exportGraph(graph, outputFile);
        System.out.println("Graph has been successfully exported to: " + filepath);
    } catch (Exception ex) {
        System.out.println("Exception while exporting DOT file " + ex);
        return false;
    }
    return true;
}
```

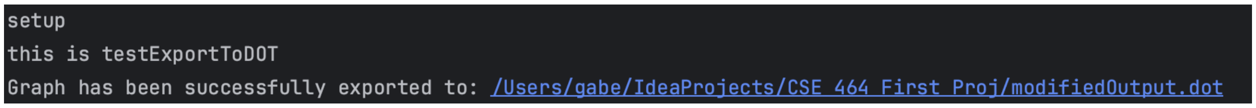
Junit Test



```
@Test
public void testExportToDOT() throws IOException {
    System.out.println("this is testExportToDOT");

    // Modify the file path to "modifiedOutput.dot"
    String outputFilePath = "/Users/gabe/IdeaProjects/CSE 464 First 
Proj/modifiedOutput.dot";
    assertTrue(s.outputDOTGraph(outputFilePath));

    // Read the content of the generated output file
    String actualOutput = Files.readString(Paths.get(outputFilePath));

    // Read the content of the expected output file
    String expectedOutput = 
Files.readString(Paths.get("/Users/gabe/IdeaProjects/CSE 464 First 
Proj/expectedModifiedOutput.txt"));

    // Compare the two outputs
    Assert.assertEquals(expectedOutput, actualOutput);
}
```

Junit Expected Test

This test exports the modfiedOutput.dot and compared it with the expected results to incur the created .dot is correct and working.



```
setup
this is testExportToDOT
Graph has been successfully exported to: /Users/gabe/IdeaProjects/CSE 464 First Proj/modifiedOutput.dot
```

Contribution History

Github/Continuous Integration Setup

- https://github.com/gvalen45/CSE-464-First-Proj

Feature Commits

Feature 1:
https://github.com/gvalen45/CSE-464-First-Proj/commit/d7f7ea4c5ba21c18d51ebf0fea3be309fe45399f

Feature 2:
https://github.com/gvalen45/CSE-464-First-Proj/commit/18c1169f199a2a6694c78c164e0a204ad4dfef32
Feature 3:
https://github.com/gvalen45/CSE-464-First-Proj/commit/cd3dab3746709d61e4eabefb6aab4d166fe1a2c7
Feature 4:
https://github.com/gvalen45/CSE-464-First-Proj/commit/889d14b5a2b938ebbd2a7d448d3b4bf8643ad59d

Branches and Successful Merges

- Branch Name: main
- Successful Merge Commit:
https://github.com/gvalen45/CSE-464-First-Proj/commit/7b9dd7da6e7d509c9f4718ef9a9f2461689989f1