GraphDot Project - CSE 464

Bradyn Flahart

Repo: https://github.com/btflahar/CSE-464-btflahar-graph.git

mvn package command works and will properly build

There should be a total of 7 Tests that run when package is run

I used the Maven Sideboard and did Lifecycle -> Package but mvn package is the same so it should work

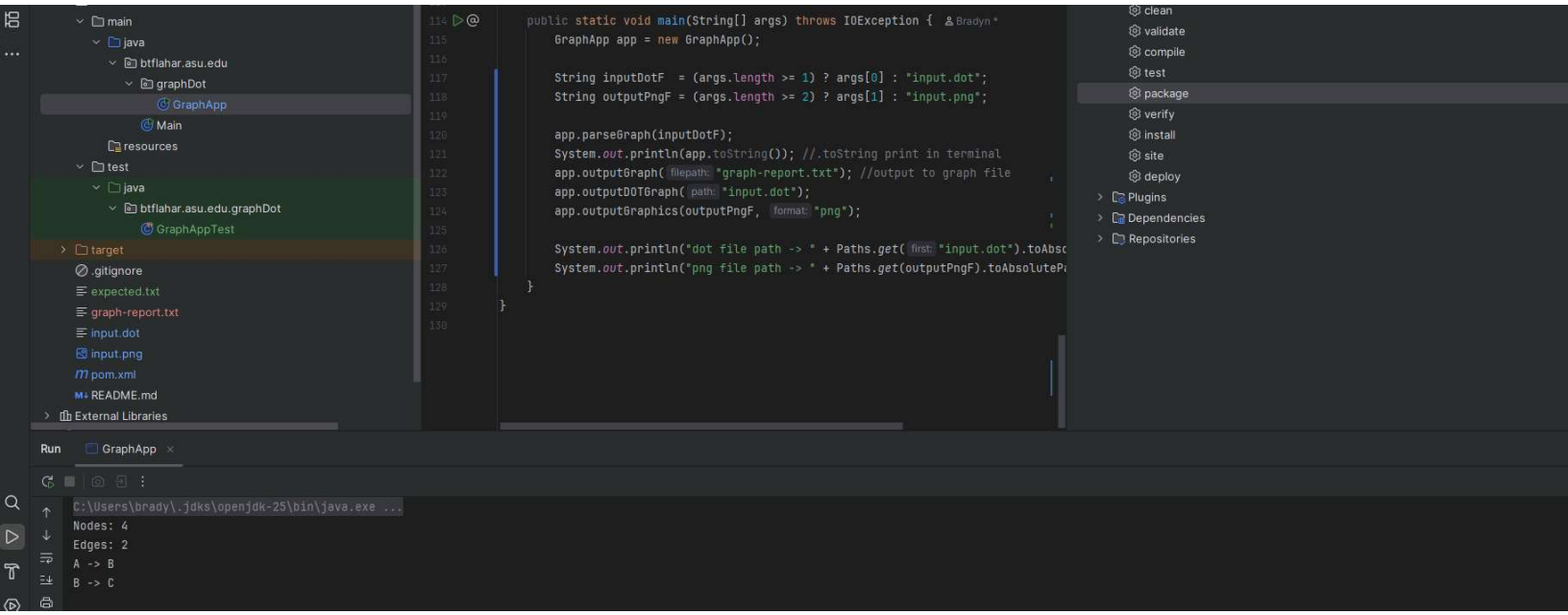
From here, you can either click the StartArrow to the left of GraphApp.main() located in CSE464-btflahar-graph/src/main/java/btflahar/asu/edu/graphDot/GraphApp
Or
You can run the command
java -cp target/classes btflahar.asu.edu.graphDot.GraphApp input.dot input.png

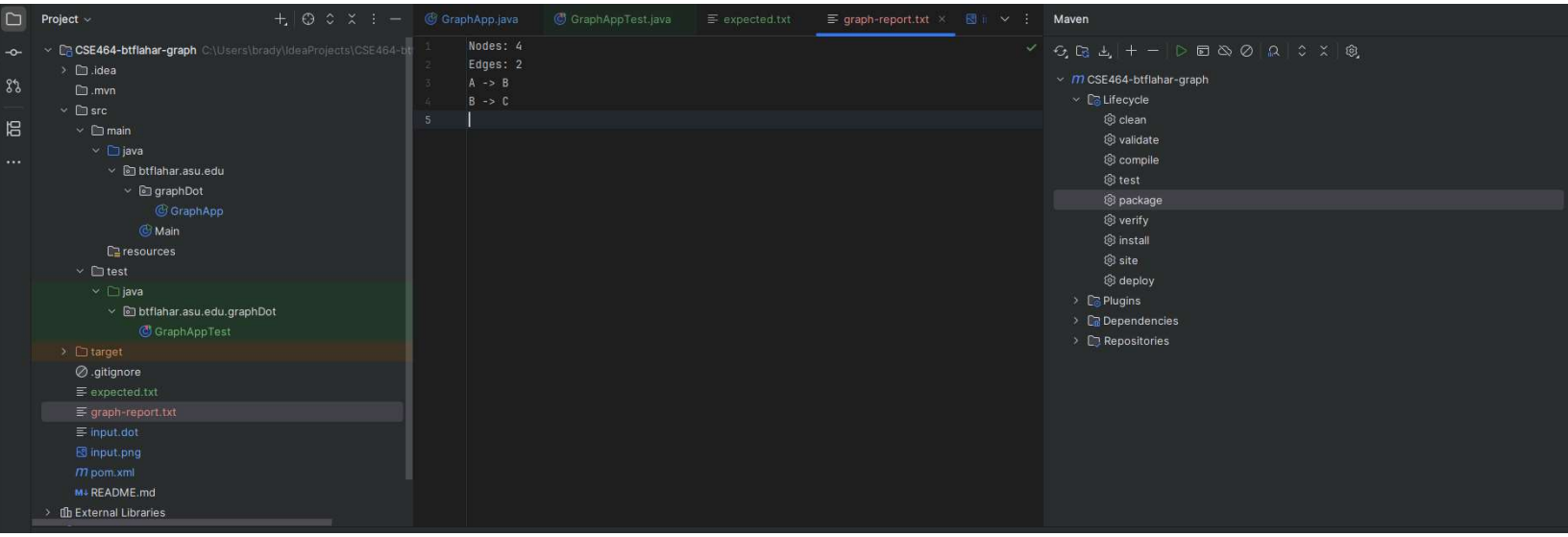
It should produce a graph-report.txt file (a text report of what the expected node and edge count is as it is meant to be a parseGraph test.)
As well as an input.dot (DOT file)
And an input.png (the png of the dot file).

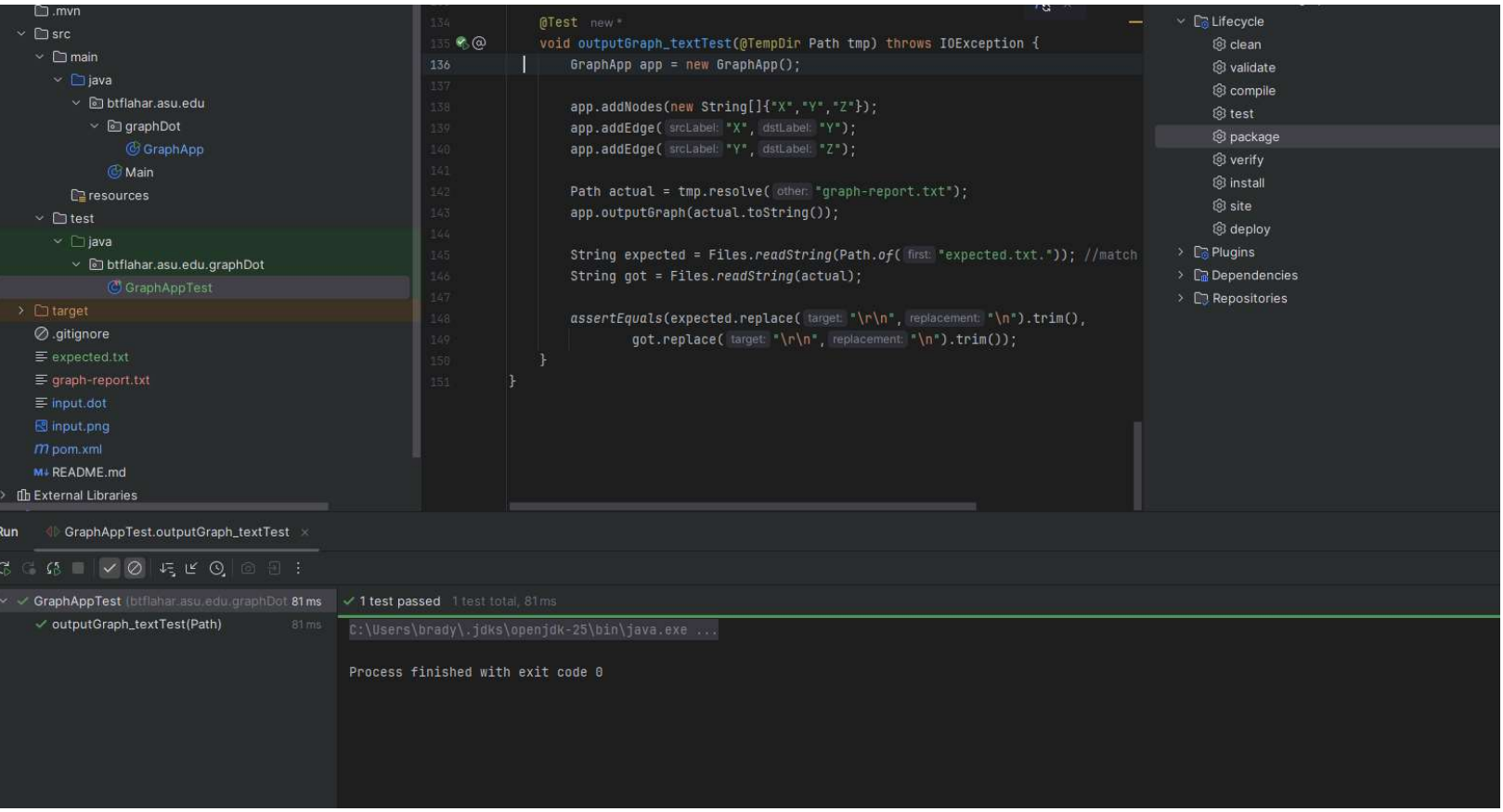
It is important to note that the input.Dot file will be overwritten after being parsed to contain the normal form of the dot graph. It will list all unique nodes, then their relationships (edges) after. This was so it would be easier to print and test later on. The example input.dot I gave is very simple

```
digraph G {
 A -> B;
 B -> C;
 D;
}
```

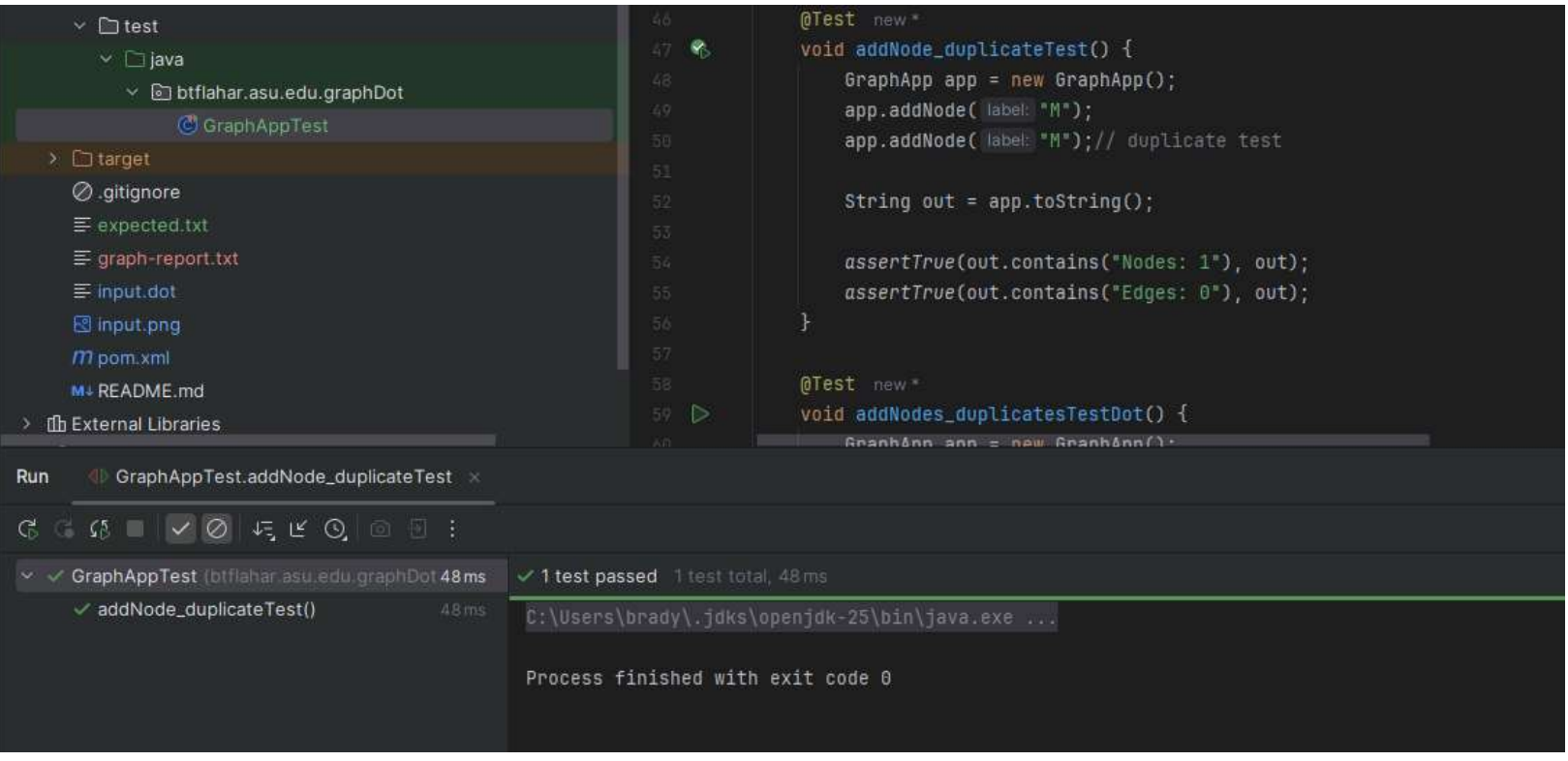
The expected number of nodes would be 4, and the number of edges would be 2.
All of my Test Functions are located in
CSE464-btflahar-graph/src/test/java/btflahar/asu/edu/graphDot/GraphAppTest.java
Below are the screenshots for the feature 1 function tests showing it works as it should, (first two pictures are parseGraph and the last is outputGraph)
(You may have to zoom in I apologize)

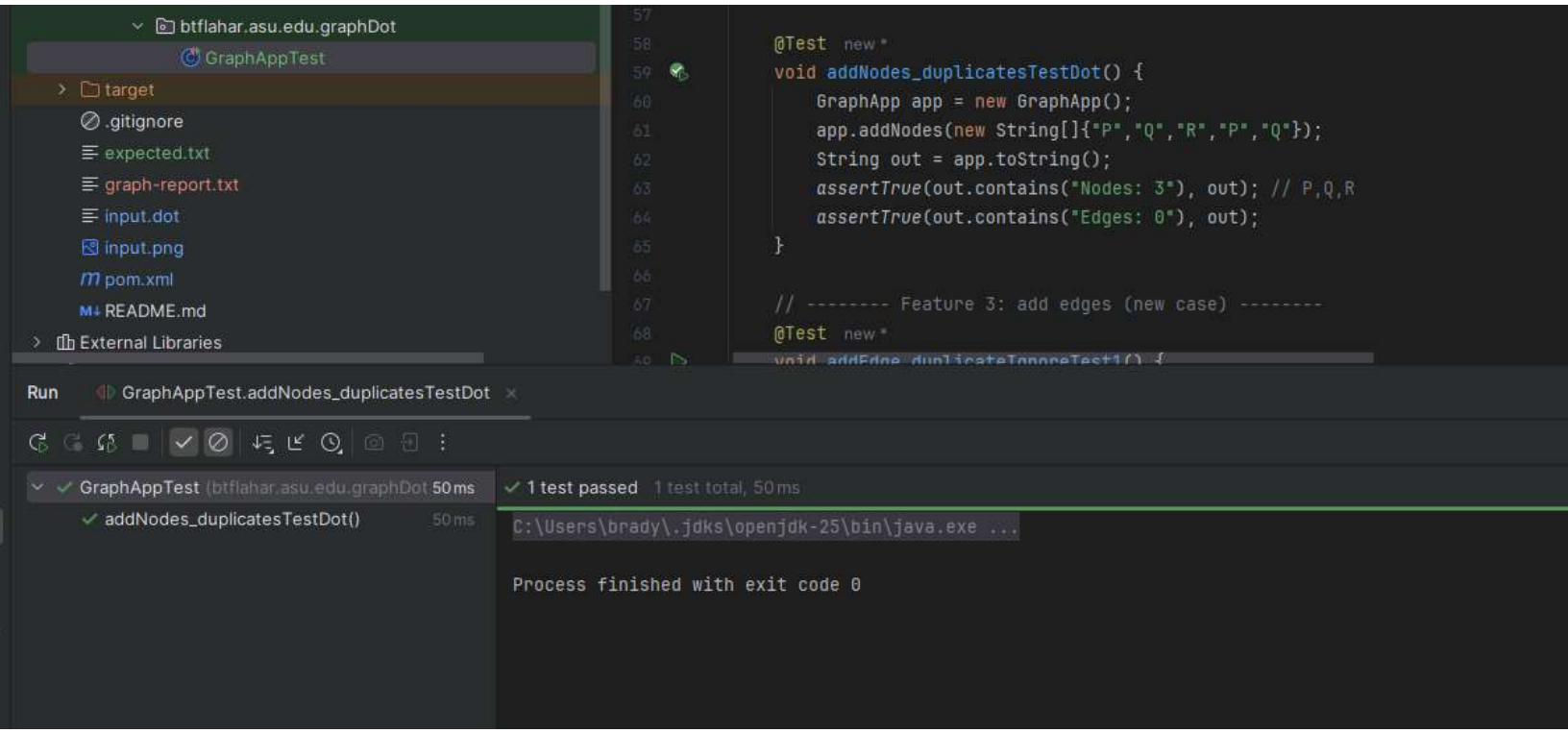

main
java
btflahar.asu.edu
graphDot
GraphApp
Main
resources
test
java
btflahar.asu.edu.graphDot
GraphAppTest
target
.gitignore
expected.txt
graph-report.txt
input.dot
input.png
pom.xml
README.md
External Libraries
public static void main(String[] args) throws IOException {
GraphApp app = new GraphApp();
String inputDotF = (args.length >= 1) ? args[0] : "input.dot";
String outputPngF = (args.length >= 2) ? args[1] : "input.png";
app.parseGraph(inputDotF);
System.out.println(app.toString()); //.toString print in terminal
app.outputGraph(filepath: "graph-report.txt"); //output to graph file
app.outputDOTGraph(path: "input.dot");
app.outputGraphics(outputPngF, format: "png");
System.out.println("dot file path -> " + Paths.get(first: "input.dot").toAbs
System.out.println("png file path -> " + Paths.get(outputPngF).toAbsoluteP
}
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies
Repositories
Run
GraphApp
C:\Users\brady\.jdks\openjdk-25\bin\java.exe ...
Nodes: 4
Edges: 2
A -> B
B -> C


Project
CSE464-btflahar-graph C:\Users\brady\IdeaProjects\CSE464-bt
.idea
.mvn
src
main
java
btflahar.asu.edu
graphDot
GraphApp
Main
resources
test
java
btflahar.asu.edu.graphDot
GraphAppTest
target
.gitignore
expected.txt
graph-report.txt
input.dot
input.png
pom.xml
README.md
External Libraries
GraphApp.java
GraphAppTest.java
expected.txt
graph-report.txt
Nodes: 4
Edges: 2
A -> B
B -> C
Maven
CSE464-btflahar-graph
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies
Repositories

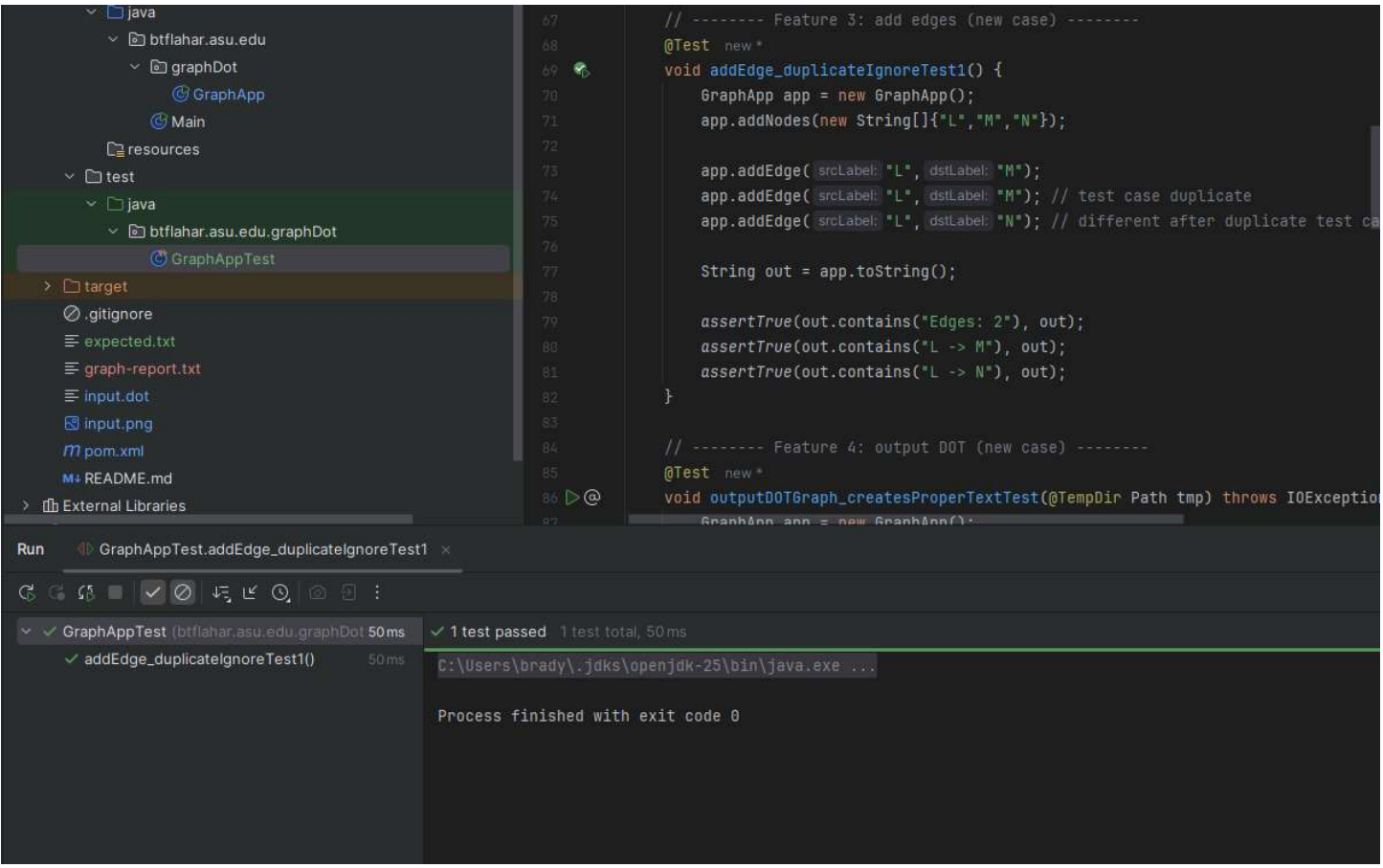


From here we move to Feature 2, addNode, addNodes Tests. For these I simply gave a test graph, then added duplicates to ensure my program wouldn't add them, but instead ignore. The expected results are just below it in the same class.

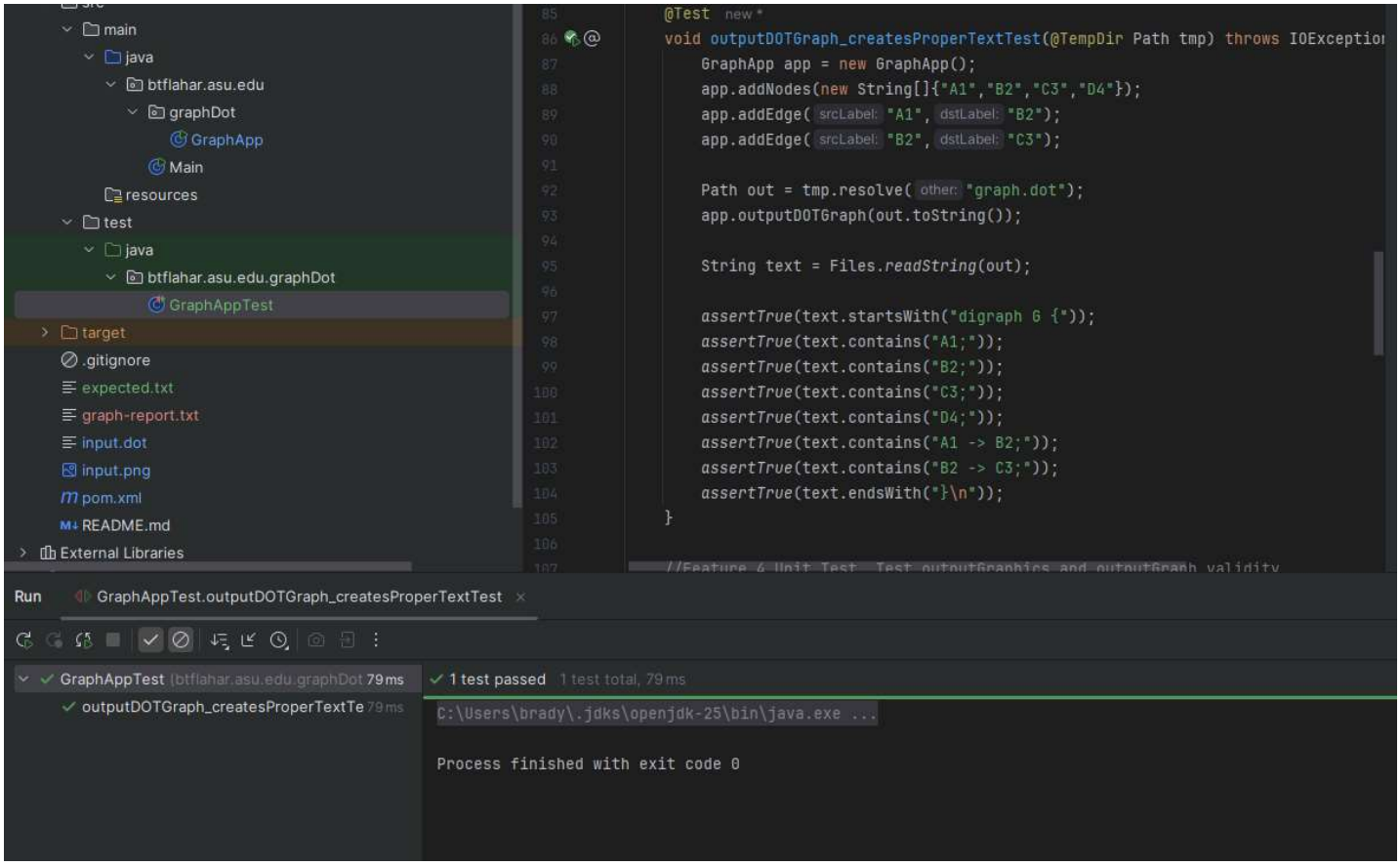


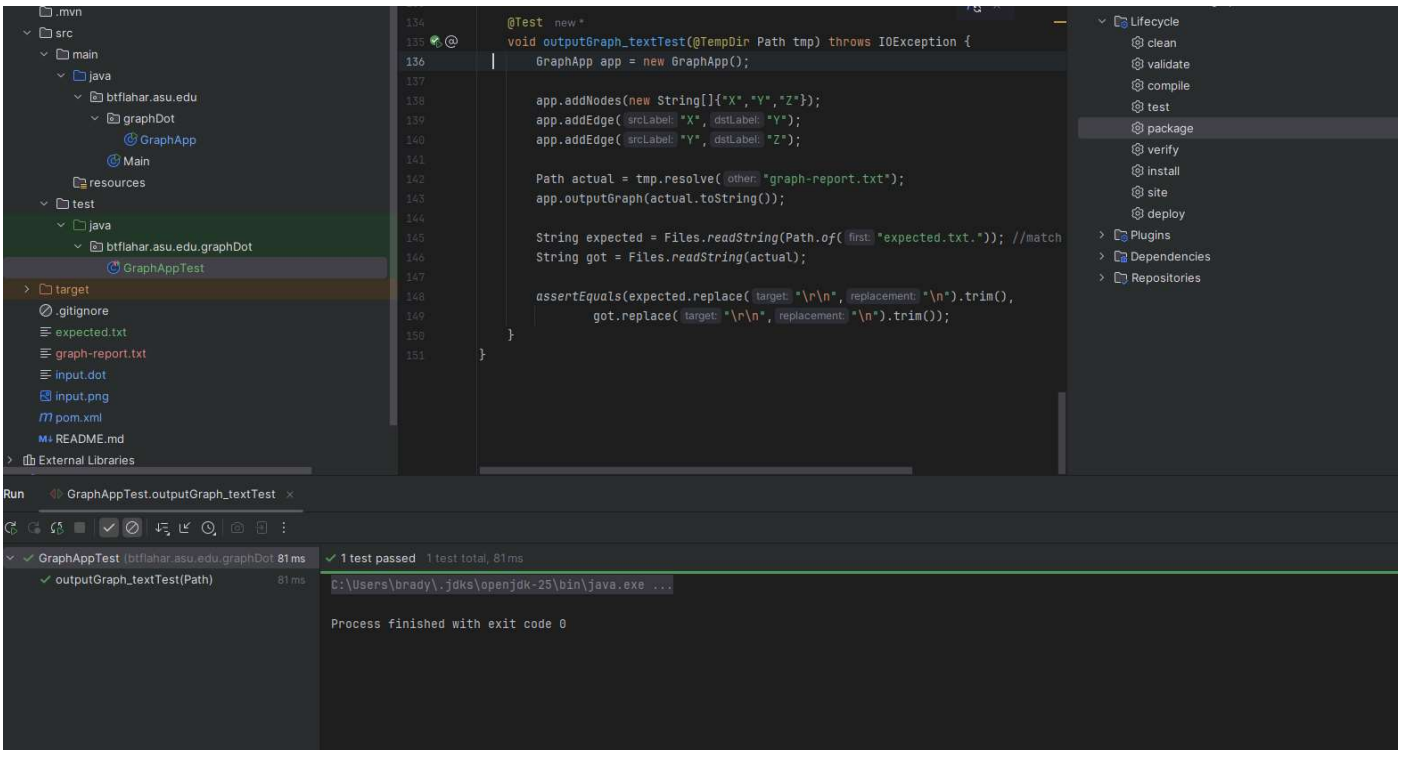


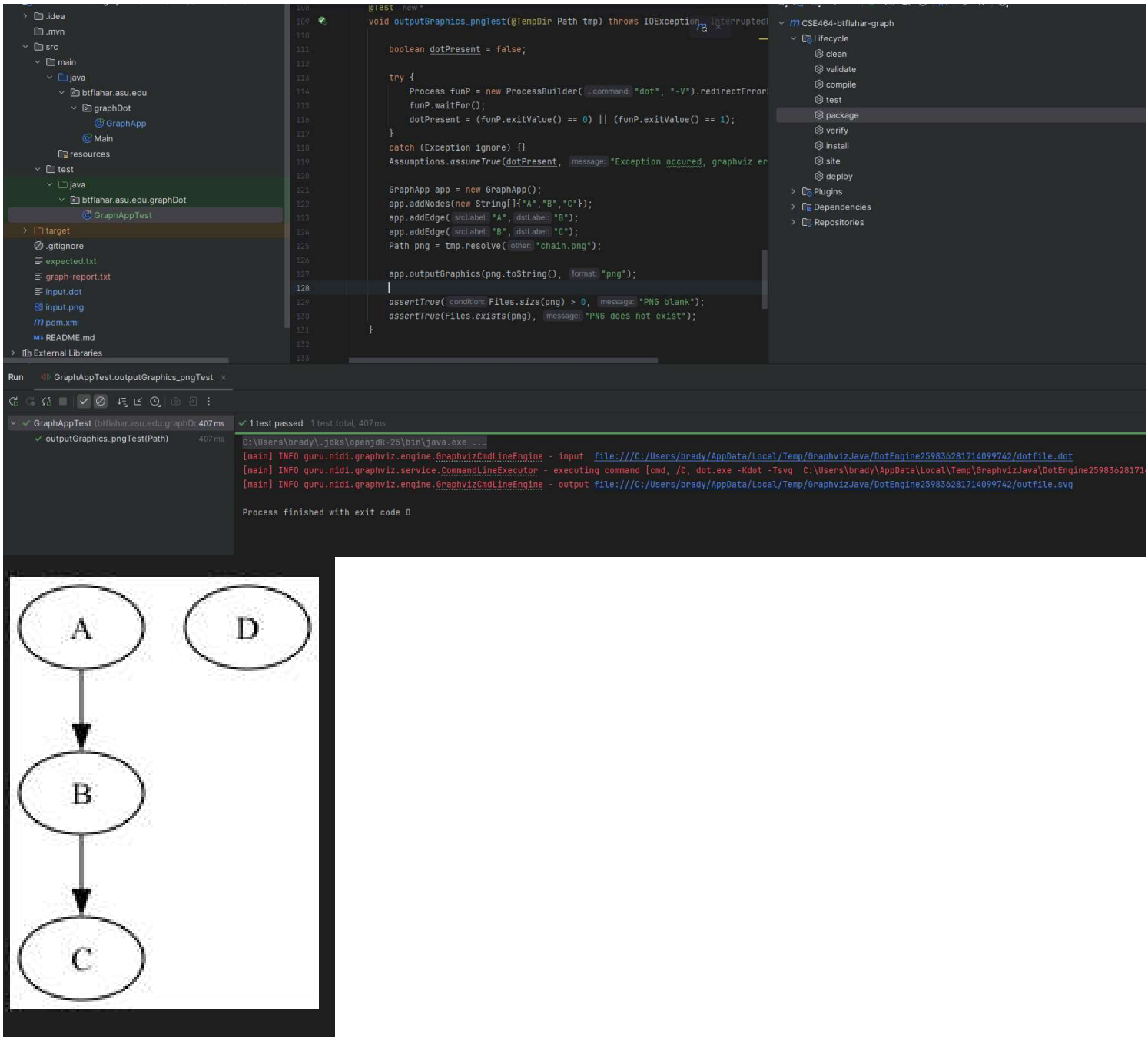
Next we have Feature 3, addEdge where I followed a similar pattern. I tried to add duplicate edges and made an example dot to test the function. I placed the simple expected output below, where is counts the number of edges (shouldnt add the duplicate edge so 2)



Finally we move onto Feature 4 which is more advanced as it needs to output a png. The sample file I gave (input.dot) can be modified to make a different png but I simply used a 4 node, 2 edge dot graph for my testing. The first pictures show the function properly, the last shows the proper PNG output.

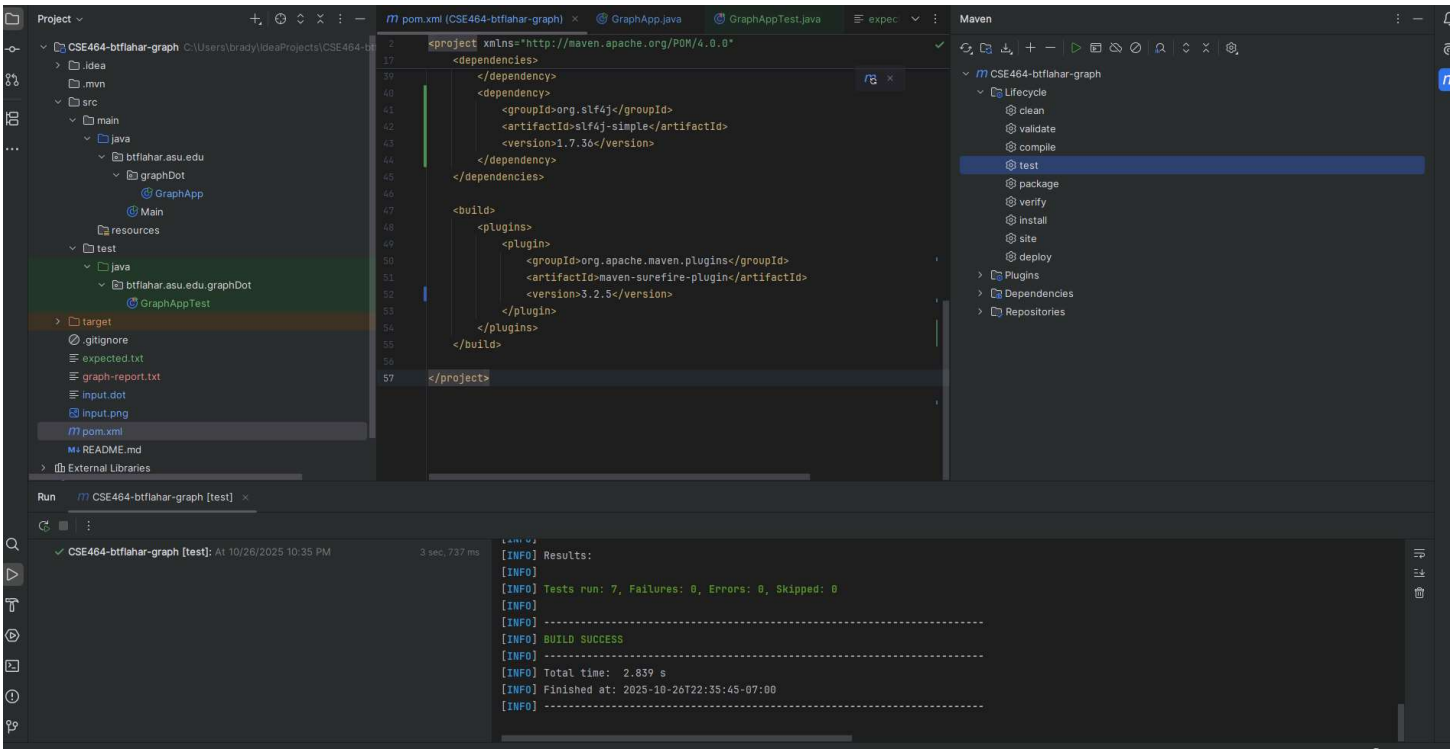




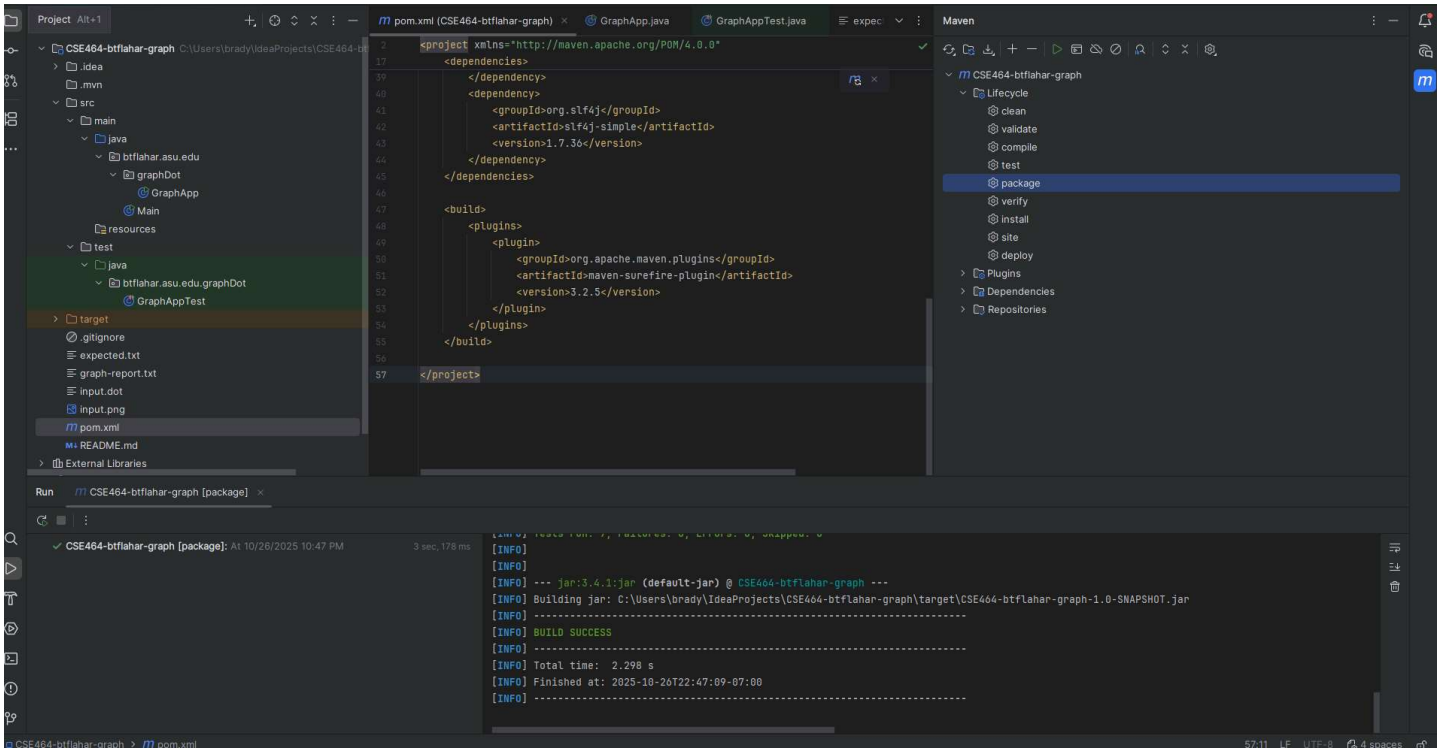


I simply did A -> B and B -> C then added an additional node that was not an edge.

For the last images, it is showing all test cases in the test path passing.



And lastly, showing that mvn package builds successfully



So to recap -
Feature 1: Tested using expected.txt and graph-report.txt, can simply click start on main function or type command I gave

Feature 2: Tests are built into test functions, they can be easily modified with desired values but are relatively simple currently. The function not only performs the action, but also compares to what the correct output should be and fails if it does not obtain the correct answer. Can run by simply going to the test path and clicking on any of the start buttons next to any function to run individually.

Feature 3: Very similar to Feature 2: Tests are simple and easy to modify and work with, already have expected outputs and test them when running them.

Feature 4: Outputs graph when program is run. The file it outputs will be input.png, the input file is input.dot. I have the expectation for both in the screenshots.

Commit Links:

Began Working on Project, mapping first few features out Commit 1 and editing pom.xml:
https://github.com/btflahar/CSE-464-btflahar-graph/commit/647925f2a562917d90befedd2e29aa6620b061de

Tried to get Feature 4 started along with a test case (finally got pom.xml working too):
https://github.com/btflahar/CSE-464-btflahar-graph/commit/8b2f644c314c5f16dbab74d321b05a3d26a7d06d

Feature 1: Full implementation and functionality
https://github.com/btflahar/CSE-464-btflahar-graph/commit/a52127e3e585b8a984a80b93e88b9a8ec011b4ec

Feature 2: Full implementation and functionality (also did .gitignore + started test case)
https://github.com/btflahar/CSE-464-btflahar-graph/commit/a40db89e18bfdd925dbf1c5ff23af39fdc7336dd

Feature 3: Full implementation and functionality (minor test case implementation)
https://github.com/btflahar/CSE-464-btflahar-graph/commit/8cd2851e7b8bc27fe5e64fbfc7029e5a799cfd89

Feature 4: Full implementation and functionality
https://github.com/btflahar/CSE-464-btflahar-graph/commit/502547b40cb91a2d4ff266ed7a7083e655f16c45

Final Commit - All Tests finished, everything functioning
https://github.com/btflahar/CSE-464-btflahar-graph/commit/fd91722e1e824cc4581802a80c776b92679eb7fa