Readme

I have implemented the features of the project by making a class for graph named myGraphClass.

GitHub Link: https://github.com/Divy2000/CSE464_2023_dpate164

Run the **src/main/application/Main.java** file to access the command-line interface.
To run the test cases run **test/myGraphClass.java**

Following are the commands I have used to run my code :-

“src/main/java/testGraph.dot” -> Parses the .dot file

“0” -> To print the graph details

“1” -> To enter the details of the graph to a file

“src/main/java/testGraph.txt” -> File-path for the output

“2” -> To add node to the graph

“node0” -> Label of the node to add

“3” -> To add multiple nodes to the graph

“node1,node2,node3” -> Three nodes to be added to the graph

“4” -> To add an edge to the graph

“white” -> First node of the edge

“node3” -> Second node of the edge

“5” -> To remove a node from the graph

“yellow” -> Label of the node to remove

“6” -> To remove multiple nodes from the graph

“node1,black” -> 2 nodes to remove

“7” -> To remove an edge from the graph

“blue” -> Label of the first node of the edge to remove

“cyan” -> Label of the second node of the edge to remove

“8” -> To output the graph to .dot file

“src/main/java/testGraph_o.dot” -> Path of the output file

“9” -> To output the graph to an image

“src/main/java/testGraph.png” -> Path of the output image file

“Q” -> To exit the program

In addition to this, now we have added feature for graph search :-

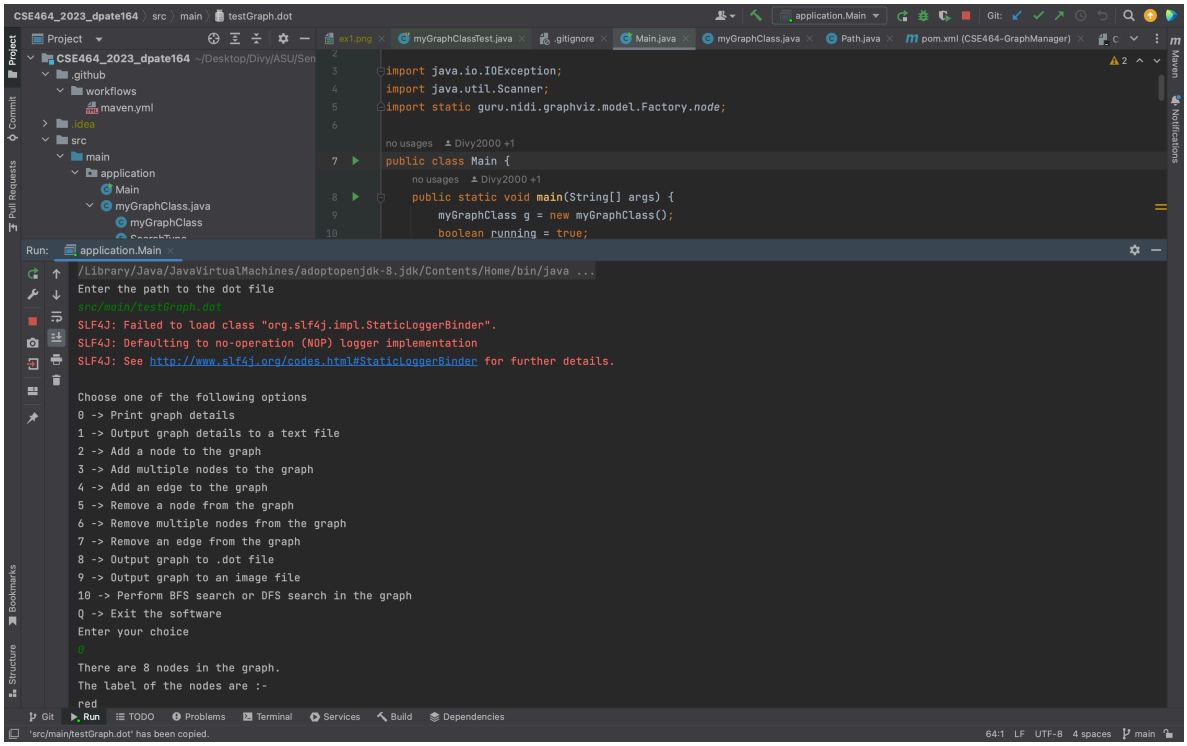
“10” -> Perform BFS search or DFS search in the graph

“bfs” -> For Breadth First Search
OR
“dfs” -> For Depth First Search

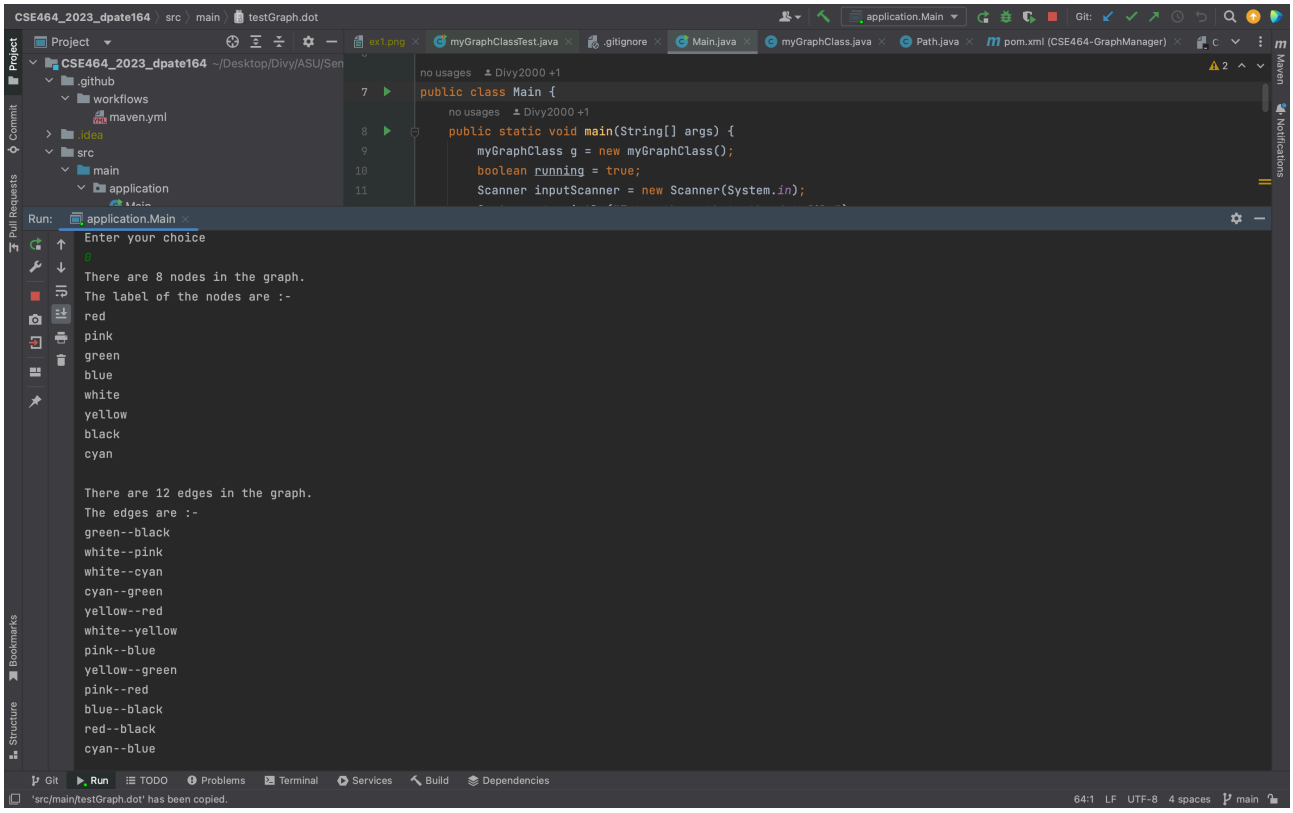
Running code screenshots and commit links :-

Feature 1

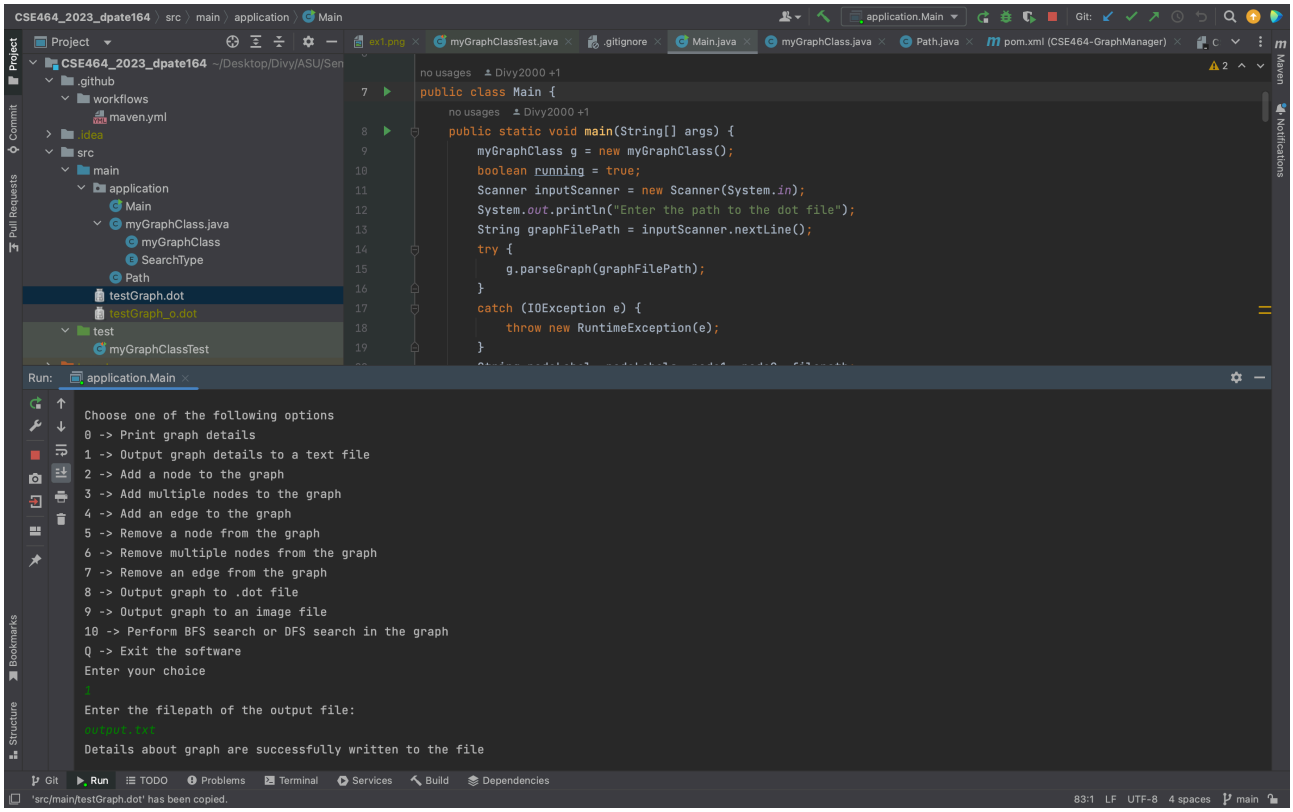
- Parsing a DOT graph file



- Print the details of the graph



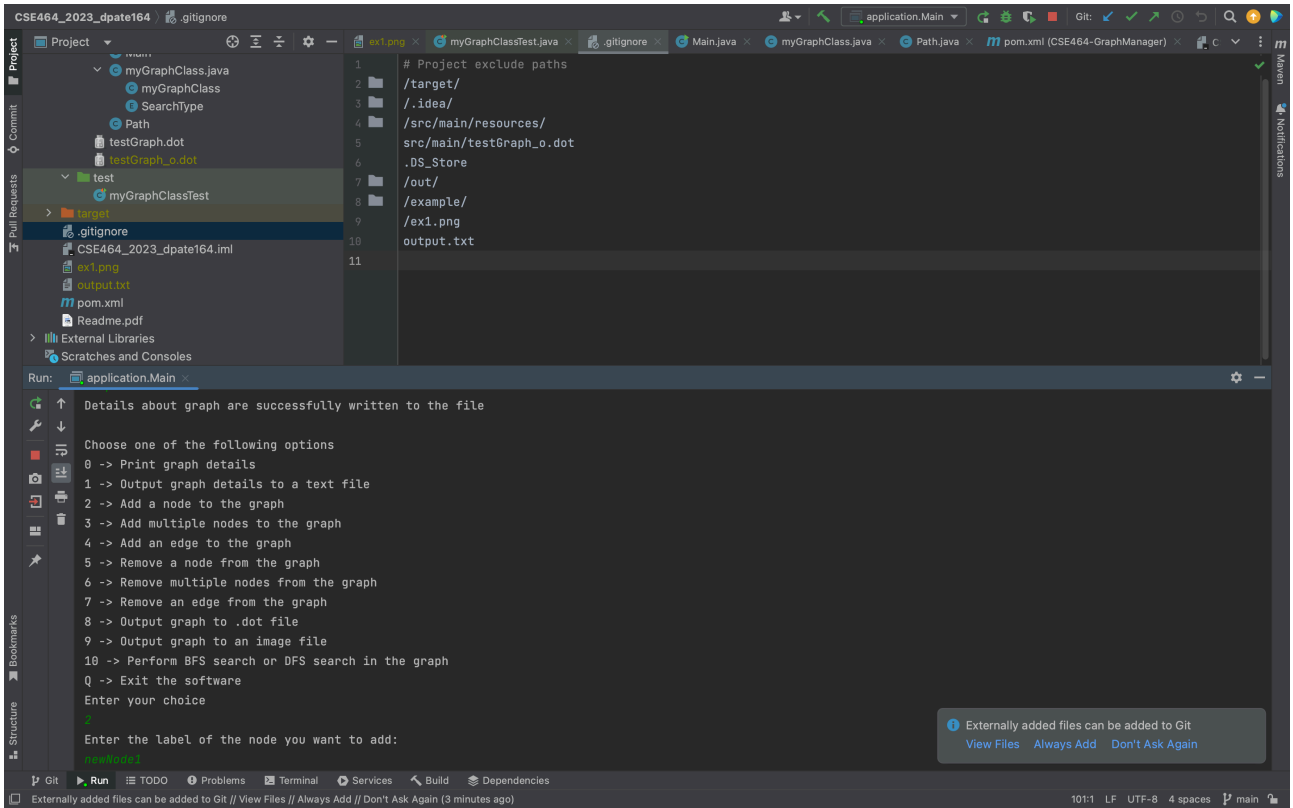
- Output the details to a file.



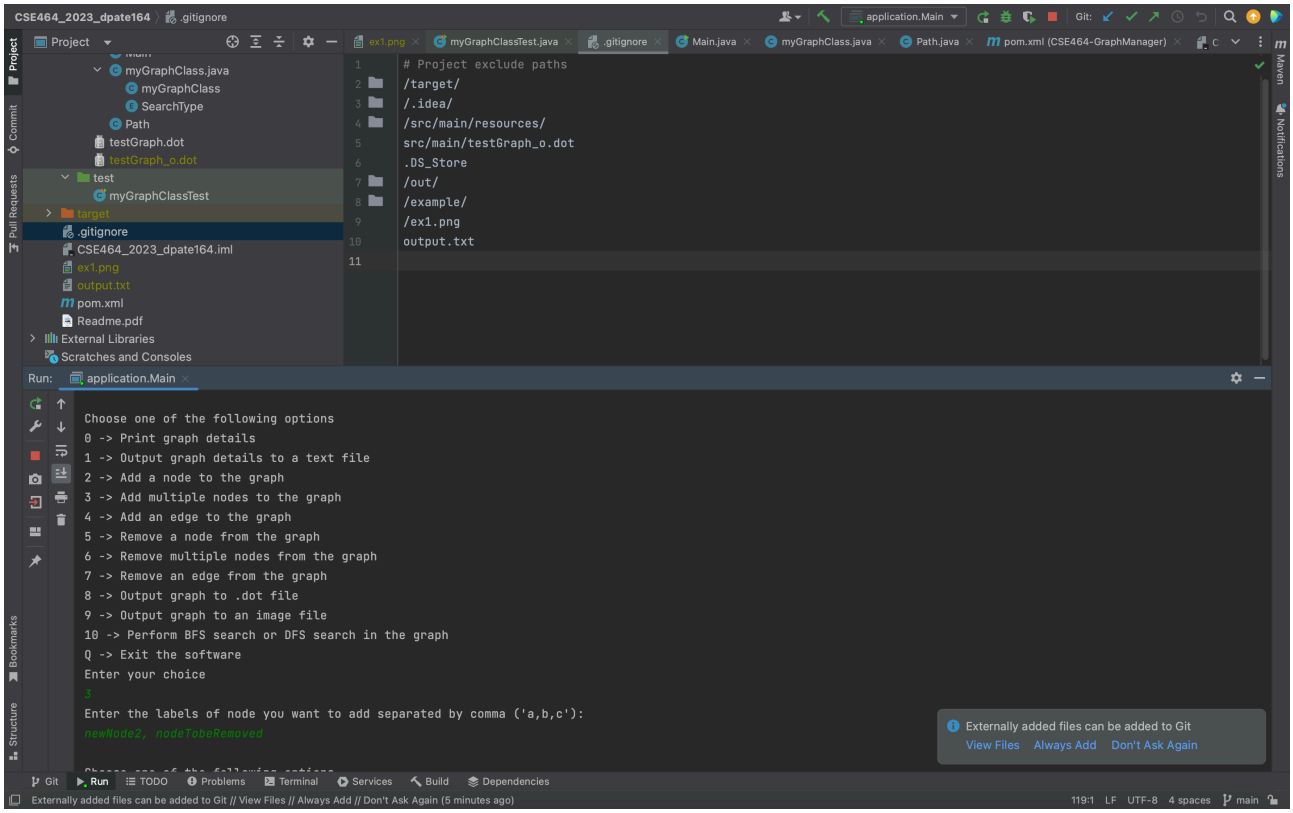
Commit link: https://github.com/Divy2000/CSE464_2023_dpate164/commit/992acecb2f21363762b58f52b81f04d088277959

Feature 2

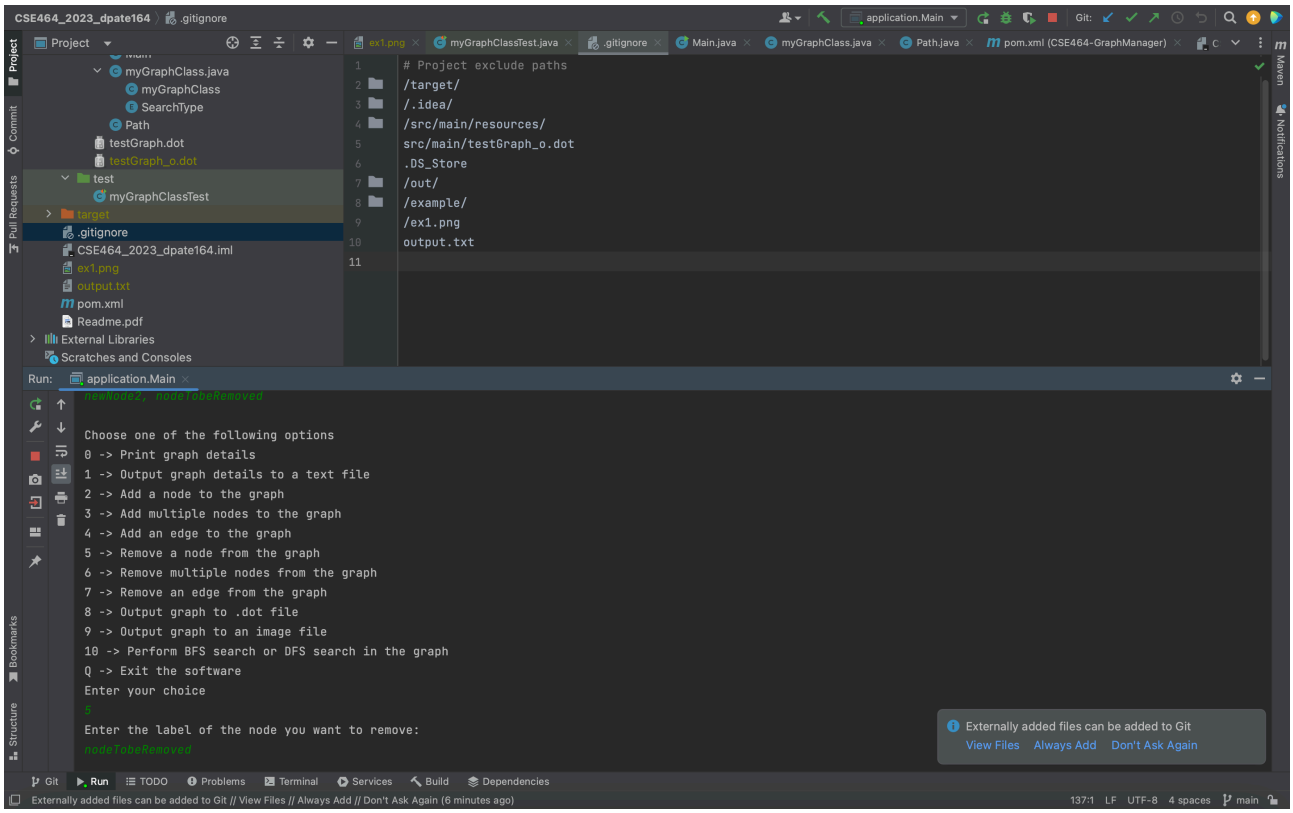
- Add a node to the graph



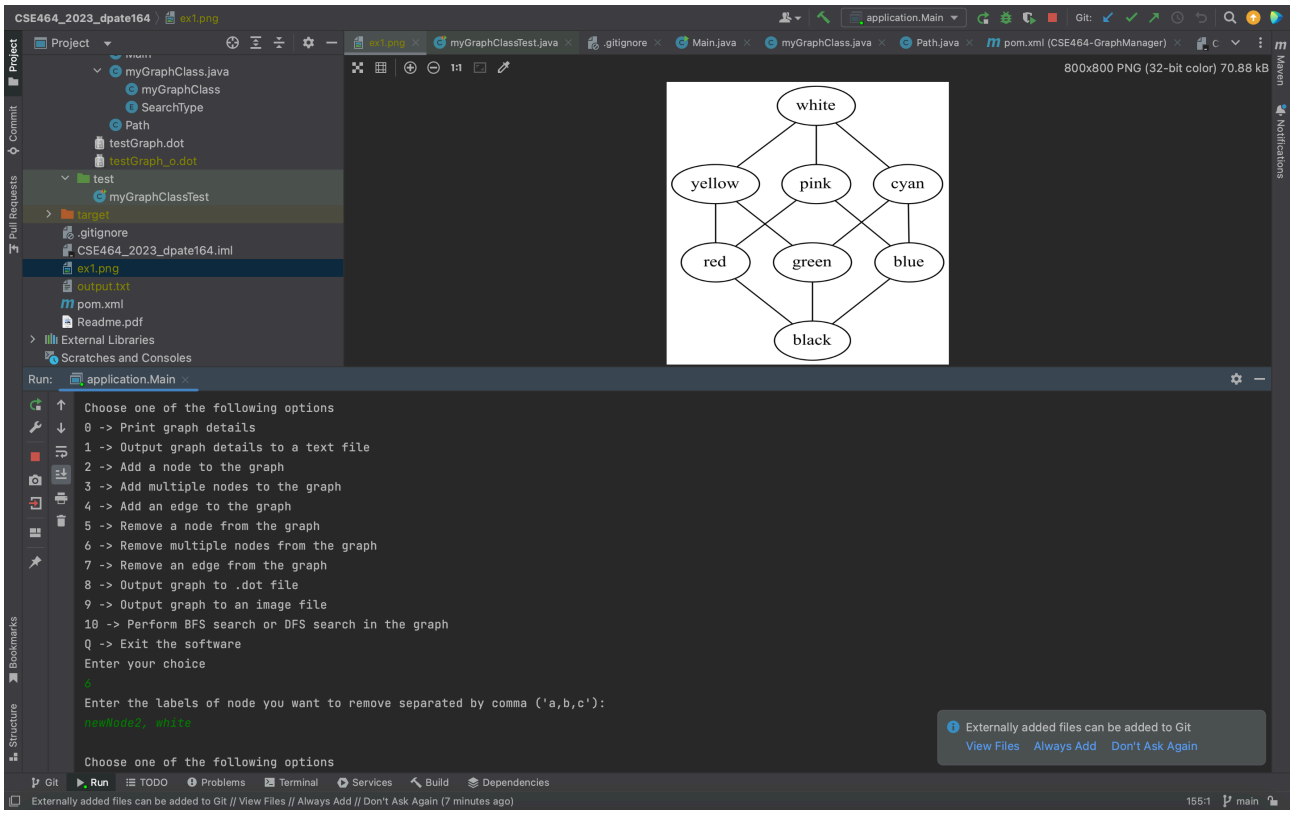
- Add list of nodes to the graph



- Remove a node from the graph



- Remove multiple node from the graph

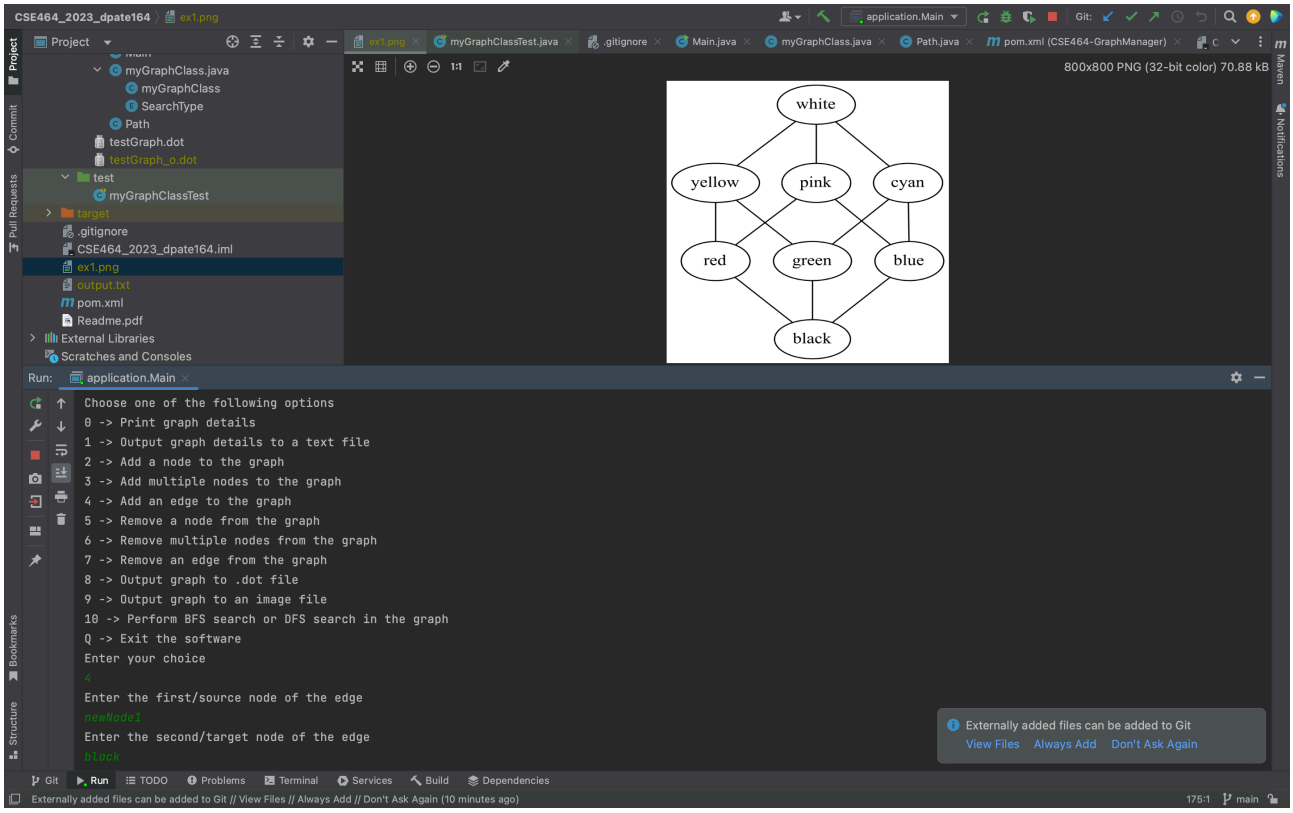


Commit Link: https://github.com/Divy2000/CSE464_2023_dpate164/commit/c2e5e0723c60eead8c52ad7ac50c373458055bff

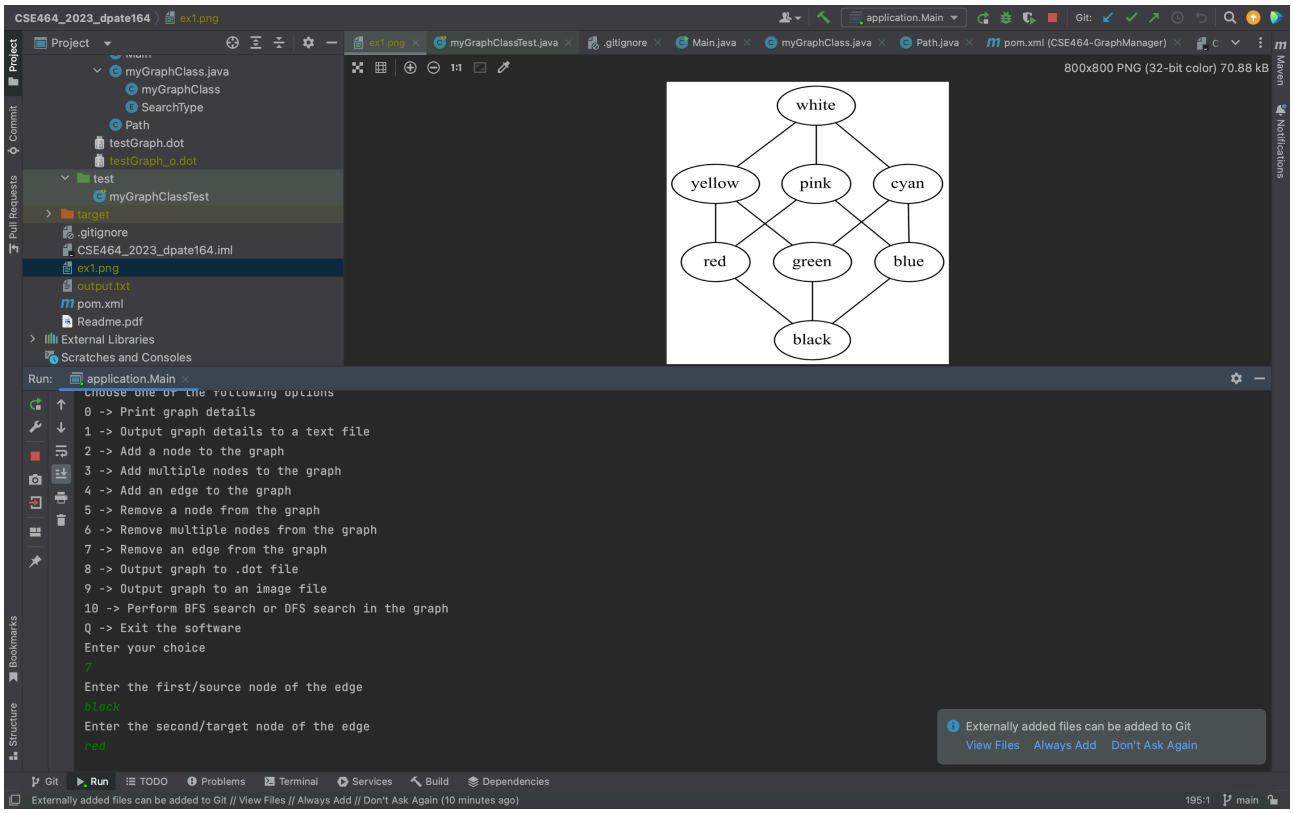
---

Feature 3

- Add an edge to the graph



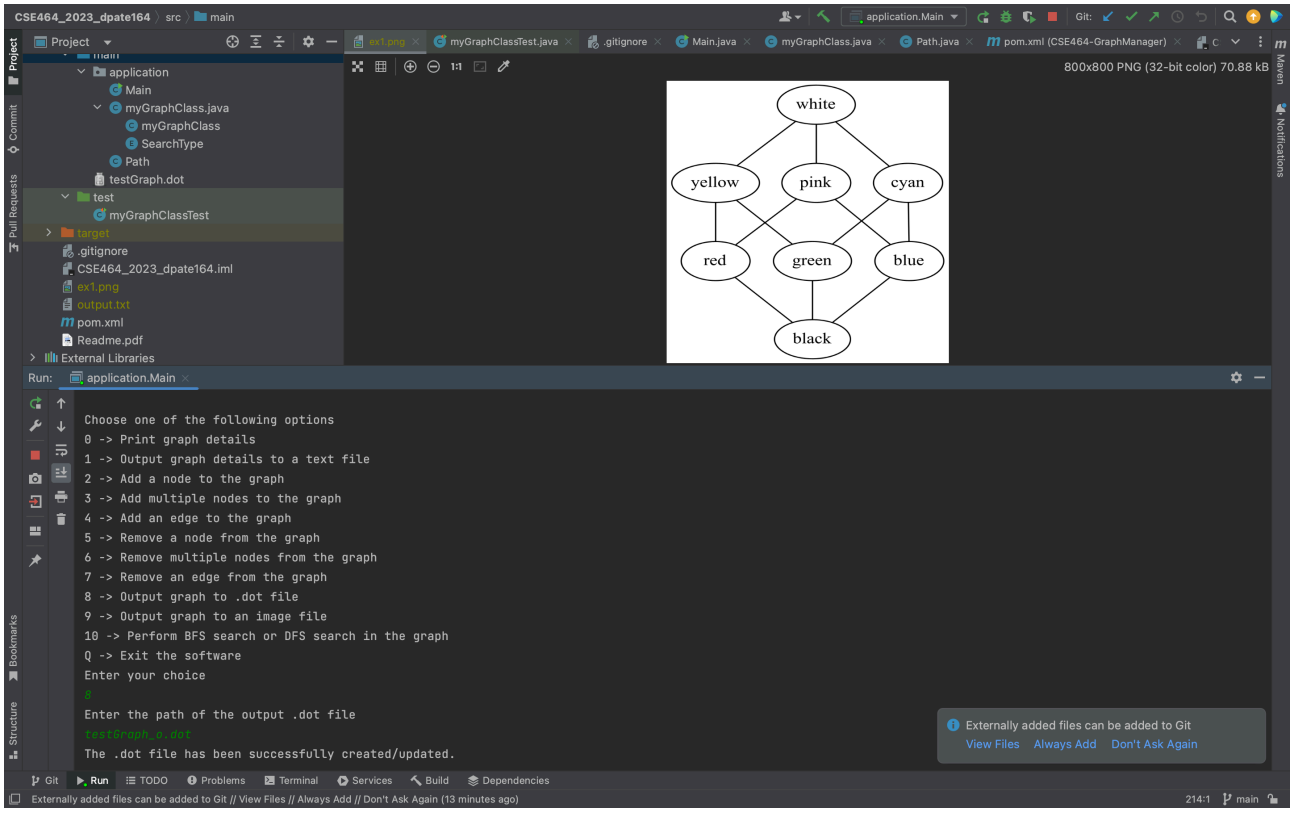
- Remove an edge from the graph



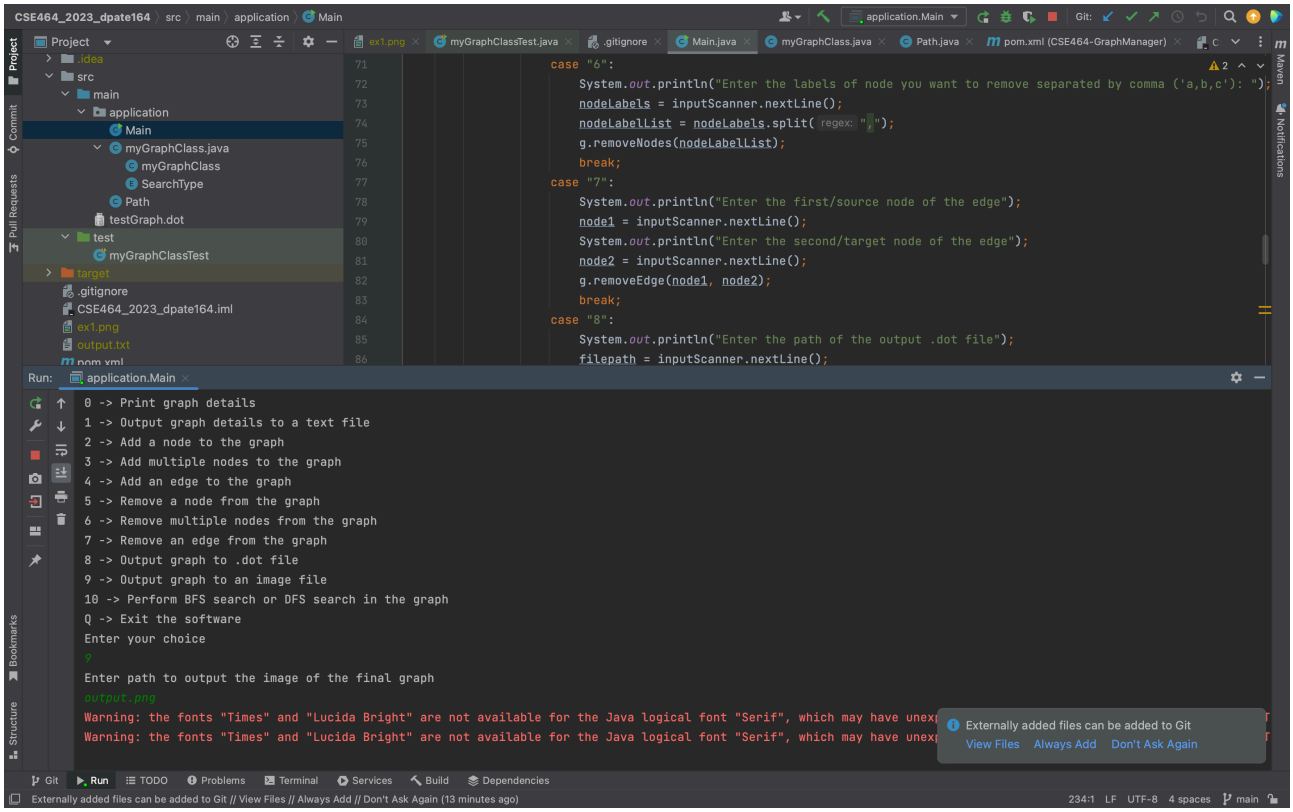
Commit Link: https://github.com/Divy2000/CSE464_2023_dpate164/commit/4cfc172714d5db0b8b19bc522782d1f3f1fa26b0

Feature 4

- Output the DOT graph file



- Output the image file



Commit Link: https://github.com/Divy2000/CSE464_2023_dpate164/commit/f36ef90b4ae229af14958bf57da1d9c8179503c5

Final commit of finishing the testing portion: https://github.com/Divy2000/CSE464_2023_dpate164/commit/82b8581aa5068c9804c18beb58abe7cc01e21766

Final commit for adding maven support: https://github.com/Divy2000/CSE464_2023_dpate164/commit/8c7363bda391c40d0865358f37e1ee66b59063dc

Final commit for completing the continuous integration: https://github.com/Divy2000/CSE464_2023_dpate164/commit/669e2d62a7a94fbc0a038a561ab0c1f0aa705246

BFS Branch: https://github.com/Divy2000/CSE464_2023_dpate164/tree/bfs

DFS Branch: https://github.com/Divy2000/CSE464_2023_dpate164/tree/dfs

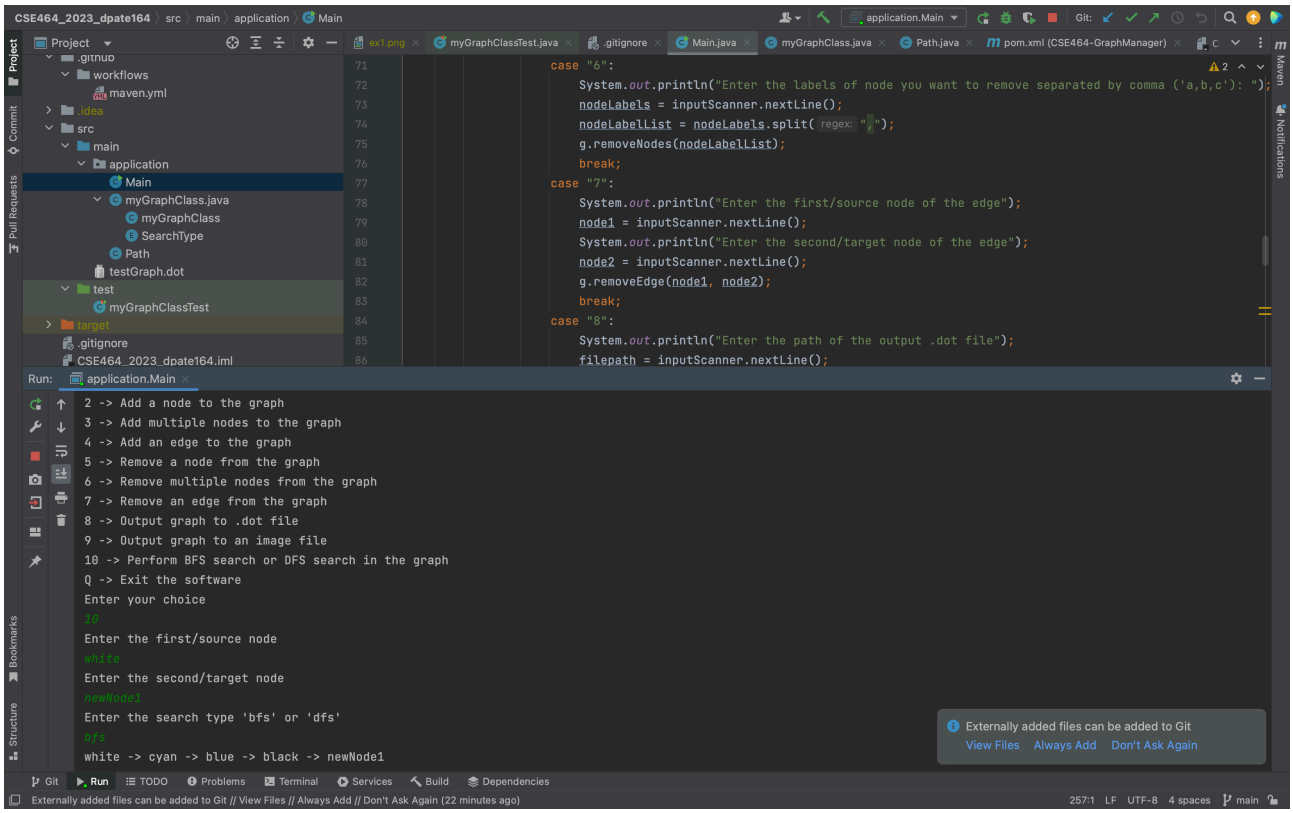
Adding BFS implementation: https://github.com/Divy2000/CSE464_2023_dpate164/commit/c65d58291562086c2838ff501920aae9e405d5bf

Adding DFS implementation: https://github.com/Divy2000/CSE464_2023_dpate164/commit/b6c9a76f85e7a4d1f695e29629fe0a0943fe39b5

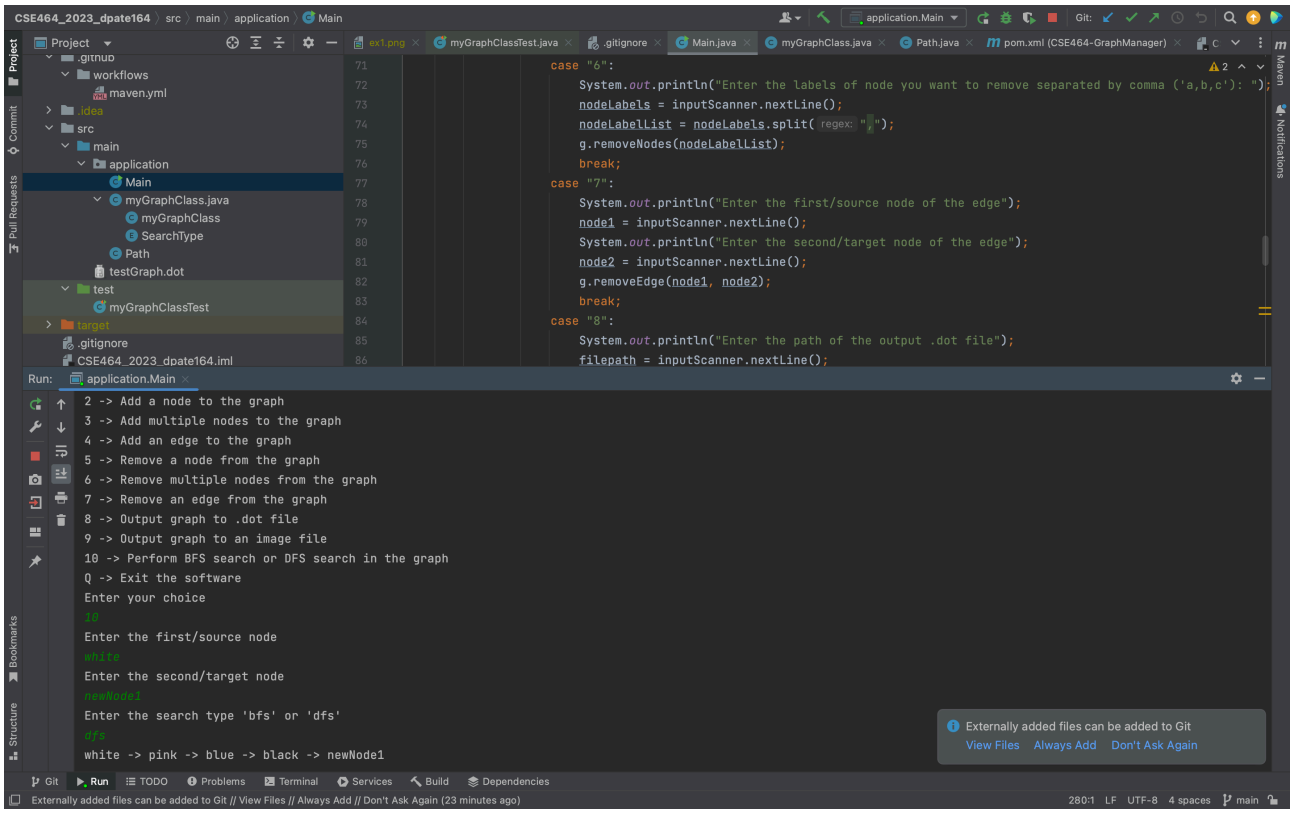
Merge branch BFS to main: https://github.com/Divy2000/CSE464_2023_dpate164/commit/119d64372419fbe526ec6756e19c5ca274fc16e6

Merge branch DFS to main: https://github.com/Divy2000/CSE464_2023_dpate164/commit/0192df91cf231a6e8ce38f5ef6508d49a49af2bc

- BFS search example



- DFS search example



Project Part 3

Part 1 - Code Refactoring

1st Commit
Extract Method Refactoring: https://github.com/Divy2000/CSE464_2023_dpate164/commit/df43dd44ff8433b66ca8657331e6dc46126e20eb

2nd Commit
Inline Refactoring: https://github.com/Divy2000/CSE464_2023_dpate164/commit/d3f43769b68d95cb1adb44e9cc0315068fc2512b

3rd Commit
Renaming Variables: https://github.com/Divy2000/CSE464_2023_dpate164/commit/da24acdaf5b617fd184807aed35d445859a9f898

4th Commit
Replaced control flag with return statement: https://github.com/Divy2000/CSE464_2023_dpate164/commit/20f9b45535ba22629360444e9d89ee9c0f0cb905

5th Commit
Removing the dead code: https://github.com/Divy2000/CSE464_2023_dpate164/commit/9d85ac57609bcbfa33e74323049b1a3b75d63fd8

Part 2 - Template Design Pattern

- In implementing template design pattern, I have extracted the common steps in "**src/main/application/GraphSearch.java**".
- Both BFS and DFS have only one difference, BFS uses Queue while DFS uses stack.
- Therefor I have implemented everything in GraphSearch.java except the part where we add and remove from the queue/stack and the part where we check if queue/stack is empty.
- I have made 3 different abstract methods for that part :-
  - addToDataStructure
  - removeFromDataStructure
  - ifEmpty **(Note: I changed the method name to isEmpty in next commit)**
- In BFS class I declare a Queue and implement abstract methods according to that.
- In DFS class I declare a Stack and implement abstract methods according to that.

Commit Link: https://github.com/Divy2000/CSE464_2023_dpate164/commit/52c7f334f1fcf076175025eda21d9968f53e1f56