CSE 464 Project Part 3

John Mitchell
CSE 464 Software QA and Testing
Fall 2025
https://github.com/johnmitchelll/CSE464_2025_jmmitc15

1. Project Overview

This project extends the previous graph search implementation by applying five refactorings and incorporating three major design patterns. The project now supports Breadth First Search, Depth First Search, and Random Walk Search using both the Template Method Pattern and the Strategy Pattern. The Random Walk algorithm demonstrates randomized traversal behavior across multiple executions.

2. Build Instructions

To build the project from the root directory, run:

mvn clean package

A successful build will display:

BUILD SUCCESS

3. Run Instructions

To run the program after a successful build:

java -cp target/classes edu.asu.graph.Main

This command:

- Loads the graph from sample.dot
- Prints graph statistics
- Runs BFS, DFS, and Random Walk search

4. Performed Refactorings (5 Total)

Each refactoring was committed separately.

Refactor 1 – Encapsulate Field

Description:
The internal name field inside the Node class was changed to private with a public getter to protect internal state and improve object oriented design.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/0ce5c86b6df61c67d5bf0bd91c569eda4c56ea6d

Refactor 2 – Extract Method

Description:
The duplicated neighbor expansion logic in BFS and DFS was extracted into a reusable helper method to reduce redundant code.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/b836373672e608dd3768227ddc75e5304f09ca72

Refactor 3 – Move Method

Description:
Search responsibilities were moved out of the Graph class and relocated into the Algorithm and strategy system to ensure that Graph only manages structure.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/c92ffaaa6d813af69f5f7666bd98faa6d2a2534f

Refactor 4 – Rename Variable

Description:
The internal variable names inside the Path class were renamed for clarity and maintainability.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/3d6f2e5738d4b2bcbe0521eecb229f5d071bc631

Refactor 5 – Remove Duplicate Object Creation

Description:
A node cache was added to the GraphParser to ensure all nodes are created only once, eliminating broken equality conditions.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/c3ceb48390c0f1d617bb64a1485ab8e5eb005d0b

5. Template Method Pattern Implementation

The Template Method Pattern was implemented using an abstract base class called AbstractGraphSearch. This class defines the shared algorithm structure for graph traversal while allowing subclasses to customize how the search frontier is managed.

Classes Used:

- AbstractGraphSearch
- BFSSearch
- DFSSearch

BFS uses a queue as its frontier while DFS uses a stack. Only the container behavior differs. The core traversal logic stays in the base class.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/af9b8d07bbd36ab717956e0a0b5d981c71dca414

6. Strategy Pattern Implementation

The Strategy Pattern was implemented using a SearchStrategy interface. Concrete strategies were created for BFS, DFS, and Random Walk. The Algorithm enum is used to dynamically choose the algorithm at runtime.

Strategy Classes:

- SearchStrategy
- BFSSearchStrategy
- DFSSearchStrategy
- RandomWalkSearchStrategy

The Graph class delegates search execution to the selected strategy using:

graphSearch(Node src, Node dst, Algorithm algo)

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/07595573e0682c7f4a9bde3d0c1b418deba268e6

7. Random Walk Search Implementation

The Random Walk algorithm performs a step by step randomized traversal from the source node by selecting a random neighbor at each step. This behavior continues until the destination is found or a safety limit is reached.

Multiple executions show different results, proving that the search is truly randomized.

Commit Link:
https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commit/0a3c6305488c34bc0a7f6e2421416658851e6adb

8. Graph Structure Output

The output below is generated using:

java -cp target/classes edu.asu.graph.Main > sample.txt

Expected output format:

Number of nodes: 7

Nodes: [a, b, c, d, e, g, h]

Number of edges: 8

Edges:

a -> b

a -> e

b -> c

b -> d

c -> a

e -> g

g -> h

h -> b

sample.txt U

```
Number of nodes: 7
Nodes: [a, b, c, d, e, g, h]
Number of edges: 8
Edges:
a -> b
a -> e
b -> c
b -> d
c -> a
e -> g
g -> h
h -> b
```

9. BFS, DFS, and Random Walk Output

BFS Output Screenshot

```
BFS Search:
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

DFS Output Screenshot

```
DFS Search:
Path{nodes=[Node{a}, Node{e}, Node{g}, Node{h}, Node{b}, Node{c}]}
```

Random Walk Output Screenshot 1

```
Random Walk Search:
Path{nodes=[Node{a}, Node{e}, Node{g}, Node{h}, Node{b}, Node{c}]}
```

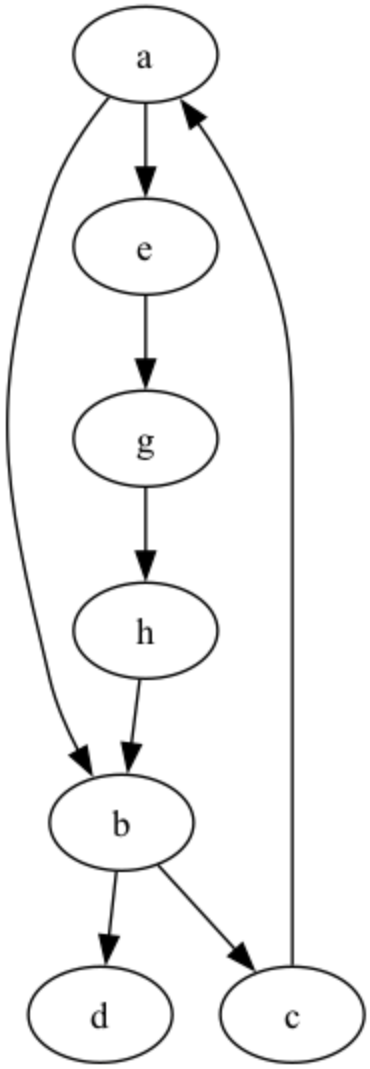
Random Walk Output Screenshot 2

```
Random Walk Search:
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

10. Graph Visualization Output

Graph image generated using:

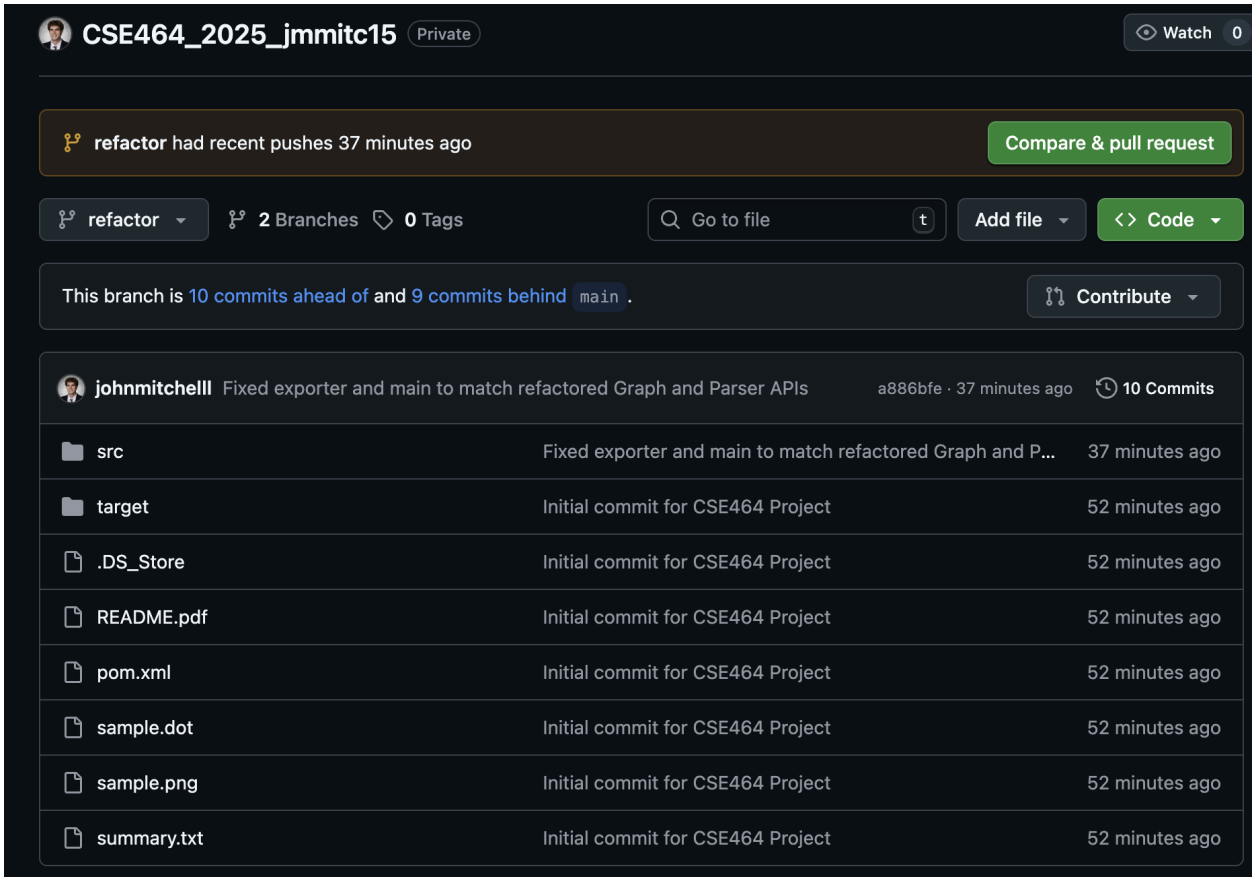
dot -Tpng sample.dot -o sample.png


a
e
g
h
b
d
c

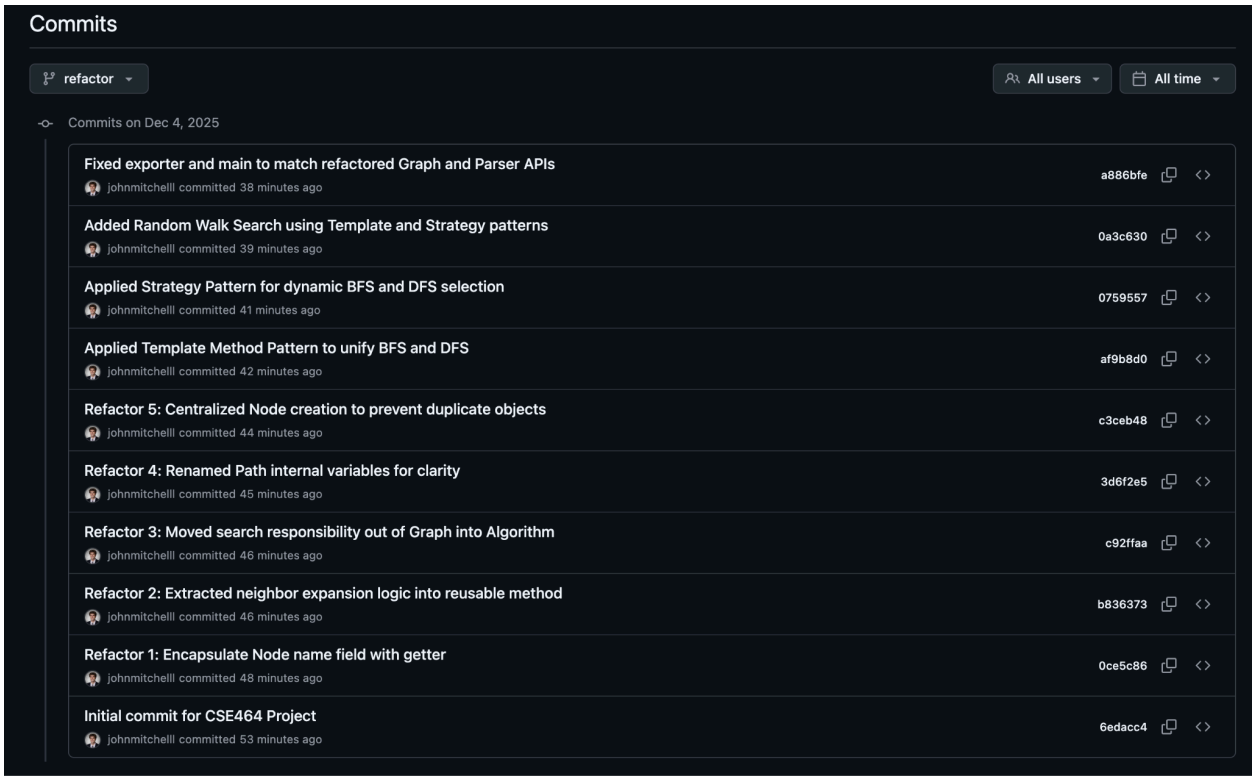
11. GitHub Workflow Proof

This project was developed using a refactor branch and merged using a pull request.

Code Base: https://github.com/johnmitchelll/CSE464_2025_jmmitc15/tree/refactor



Commits: https://github.com/johnmitchelll/CSE464_2025_jmmitc15/commits/refactor/



Pull request: https://github.com/johnmitchelll/CSE464_2025_jmmitc15/pull/1