CSE464 Course Project

Building the Project

To build the project, run the following command in the project root directory:

```
mvn package
```

This command will compile the code, run the tests, and create a JAR file in the `target` directory.

Running the Project

After building the project, you can run the `Main` class using the following command:

```
java -cp target/CSE464-2023-syadav42-1.0-SNAPSHOT.jar Main
```

Features and Usage

Example input

```
input.dot:

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

1. Parse a DOT graph file

```
GraphManager manager = new GraphManager();
manager.parseGraph("path/to/your/input.dot");
System.out.println(manager.toString());
```

Output:

```
Parsing input.dot file...
Parsing file: input.dot
Parsed graph: digraph "G" {
"A" -> "B"
```

```
"B" -> "C"
"C" -> "A"
}
Successfully parsed input.dot

Initial Graph Information:
Number of nodes: 3
Node labels: [A, C, B]
Number of edges: 3
Edges: [C -> A, B -> C, A -> B]
```

2. Add nodes

```
manager.addNode("NewNode");
manager.addNodes(new String[]{"Node1", "Node2", "Node3"});
```

Output:

```
Adding nodes and edges...
Added node 'NewNode': true
Added nodes: Node1, Node2
```

3. Add edges

```
manager.addEdge("Node1", "Node2");
```

Output:

```
Added edge 'Node1' -> 'Node2': true
```

4. Output to DOT file and PNG

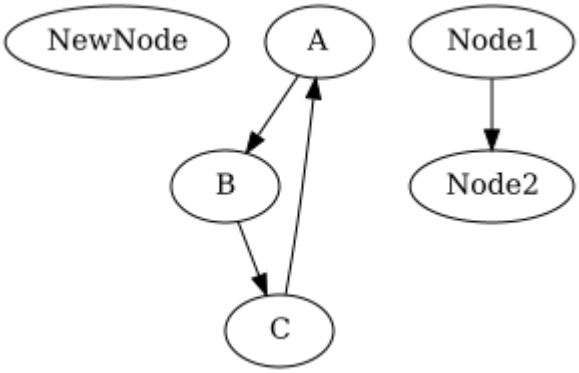
```
manager.outputDOTGraph("path/to/output.dot");
manager.outputGraphics("path/to/output.png", "png");
```

Output:

```
Outputting to DOT and PNG...
DOT output:
```

```
digraph {
  "Node1" -> "Node2"
  "A" -> "B"
  "C" -> "A"
  "B" -> "C"
}
Successfully output graph to output.dot
Graphics output saved to: output.png in PNG format
Graphics output saved to: output.png
```

Output PNG:



New Features in Part 2

5. Remove Nodes and Edges

```java
// Remove a single node
manager.removeNode("Node1");

// Remove multiple nodes
manager.removeNodes(new String[]{"Node2", "Node3"});

// Remove an edge
manager.removeEdge("A", "B");
```

Example Output:

```
=== Creating Initial Graph ===
Initial Graph:
Number of nodes: 4
Node labels: [C, D, B, A]
Number of edges: 4
Edges: [A -> B, A -> D, C -> D, B -> C]


=== Removing Node 'B' ===
After removing node 'B':
```

```
Number of nodes: 3
Node labels: [A, C, D]
Number of edges: 2
Edges: [C -> D, A -> D]


=== Removing Edge A->D ===
After removing edge 'A->D':
Number of nodes: 3
Node labels: [C, A, D]
Number of edges: 1
Edges: [C -> D]


=== Attempting to remove non-existent node ===
```

6. Graph Search (BFS and DFS)

```
// Search using BFS
GraphPath bfsPath = manager.GraphSearch("A", "D", Algorithm.BFS);

// Search using DFS
GraphPath dfsPath = manager.GraphSearch("A", "D", Algorithm.DFS);

// Print the found paths
System.out.println("BFS Path: " + bfsPath);  // Output: A -> B -> C -> D
System.out.println("DFS Path: " + dfsPath);  // Output: A -> D
```

Example Output:

```
=== Creating Graph for Path Finding ===
Graph Structure:
Number of nodes: 5
Node labels: [D, C, A, E, B]
Number of edges: 5
Edges: [A -> B, D -> E, B -> C, A -> E, C -> D]


=== Finding path from A to E using BFS ===
BFS Path: A -> E

=== Finding path from A to E using DFS ===
DFS Path: A -> B -> C -> D -> E

=== Finding path in cyclic graph ===
Cyclic Graph Structure:
Number of nodes: 3
Node labels: [A, C, B]
Number of edges: 3
```

```
Edges: [C -> A, B -> C, A -> B]

BFS Path A->C: A -> B -> C
DFS Path A->C: A -> B -> C
```

New Features in Part 3

7. Code Refactoring

Five main refactoring changes were implemented to improve code quality:

1. Extract File Operations:
    - Extracted file I/O operations into dedicated class
    - Improved separation of concerns
2. Extract Path Finding Logic:
    - Moved search algorithms to separate classes
    - Enhanced modularity and testability
3. Consolidate Node Removal:
    - Combined duplicate removal code
    - Reduced code duplication
4. Variable Extraction:
    - Improved readability in addEdge method
    - Enhanced code clarity
5. Method Renaming:
    - Updated method names to follow conventions
    - Improved code consistency

8. Template Pattern Implementation

The template pattern was implemented to standardize graph search algorithms:

```
// Using template pattern for graph search
GraphSearchTemplate bfsSearch = new BFSSearch(graph);
GraphSearchTemplate dfsSearch = new DFSSearch(graph);
```

9. Strategy Pattern Implementation

Strategy pattern allows runtime algorithm selection:

```
// Using strategy pattern for algorithm selection
GraphSearchStrategy strategy =
SearchStrategyFactory.getStrategy(Algorithm.BFS);
GraphPath path = strategy.findPath(graph, "A", "B");
```

10. Random Walk Search

New random walk implementation using both patterns:

```
// Using random walk search
GraphPath randomPath = manager.GraphSearch("A", "C",
Algorithm.RANDOM_WALK);
```

Random Walk Output:

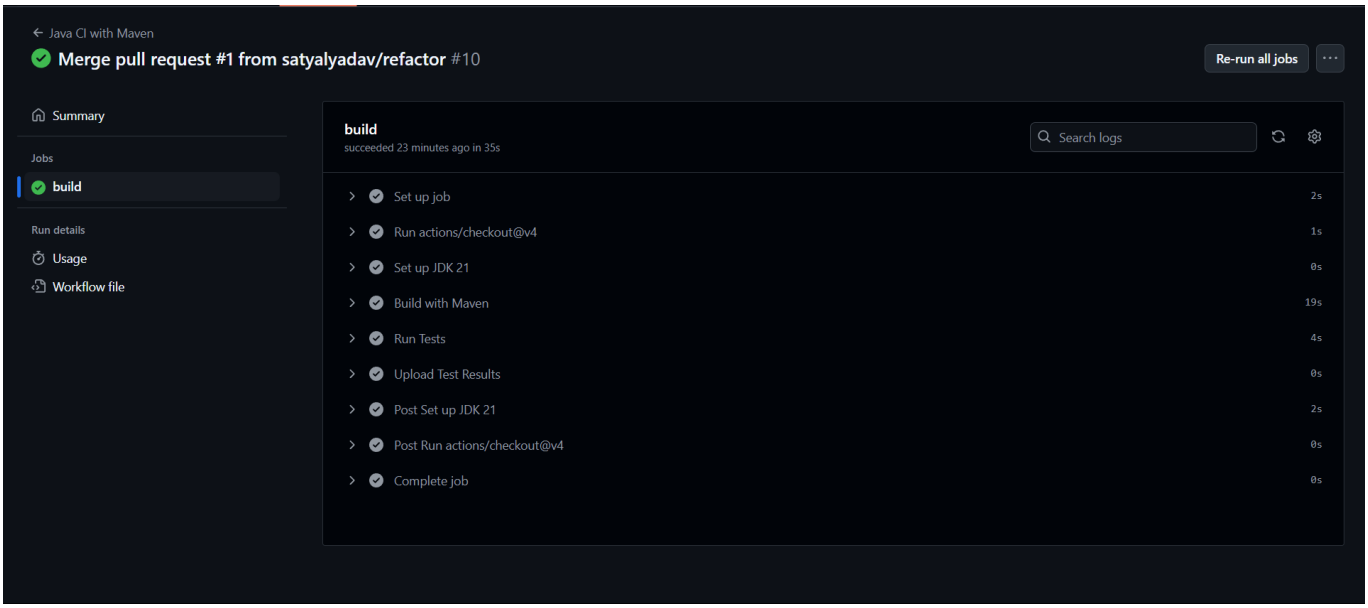
```
random testing
visiting Path{nodes=[Node{a}]}
visiting Path{nodes=[Node{a}, Node{b}]}
visiting Path{nodes=[Node{a}, Node{b}, Node{c}]}
Path{nodes=[Node{a}, Node{b}, Node{c}]}

random testing
visiting Path{nodes=[Node{a}]}
visiting Path{nodes=[Node{a}, Node{e}]}
visiting Path{nodes=[Node{a}, Node{e}, Node{f}]}
visiting Path{nodes=[Node{a}, Node{b}]}
visiting Path{nodes=[Node{a}, Node{b}, Node{c}]}
Path{nodes=[Node{a}, Node{b}, Node{c}]}
```

Continuous Integration

This project uses GitHub Actions for continuous integration. Every push to the repository automatically:

- Builds the project
- Runs all tests
- Reports test results



Running Tests

To run the tests, use the following command:

```
mvn test
```

GitHub Commits

Part 1 Features

- Parse Graph: 6864b69
- Add Nodes: c75c667
- Add Edges: 86ce05d
- Output Graph: 29c02e6

Part 2 Features

- Remove APIs: 4fa5ea5
- CI Setup: 58e0e20
- BFS Implementation: 2cea339
- DFS Implementation: 3c43d5a
- Algorithm Selection: a3e4f7c

Part 3 Features

- Code Refactoring
  - Extract File Operations: 6997db1
  - Extract Path Finding Logic: e16b967
  - Consolidate Node Removal: 3d8d723
  - Variable Extraction: 3b9b73d
  - Method Renaming: b086ddb
- Template Pattern Implementation

    - Base Template: a297e92
    - Template Refinements: 15d4e1e
- Strategy Pattern Implementation: 9afa0ad
- Random Walk Search: a64a1a0

Pull Request

Part 3 Pull Request: Pull Request #1

Branches

- Main: main
- BFS Branch: bfs
- DFS Branch: dfs
- Refactor: refactor

Merges

- BFS Merge: bf55837
- DFS Merge with Algorithm Selection: a3e4f7c
- Refactor Merge: fcbe375