Project Part 3 Readme

Refactors

- refactor: Encapsulate graphObject and create a getter function.
  This refactor commit aims to encapsulate the `graphObject` object by making it private to the `GraphData` class. A getter function called `getGraph` is now used to safely access the object.
- refactor: Rename opt variable to graphString.
  The variable name 'opt' does not provide a good idea about its purpose, so I rename it to `graphString` to make it clearer.
- refactor: Convert test filepaths to static final variables.
  This commit converts test filepaths to `static` `final` variables. This helps to organise the test parameters a bit more easily.
- refactor: Add nullcheck in printPath function.
  This commit aims to avoid a `NullPointerException` by adding a null check to the function printPath.
- refactor: Add comments in multiple functions
  This commit introduces comments at various parts of the codebase to explain written code better.

Template Pattern

The Template Pattern involves:

- Creating an abstract class (`GraphSearch`) to define common graph search steps.
- Extending this template with concrete subclasses (`BFS` and `DFS`) that implement specific traversal methods by overriding abstract template methods.
- This structure provides a shared algorithm skeleton while allowing `BFS` and `DFS` to have their unique implementations.

Strategy Pattern

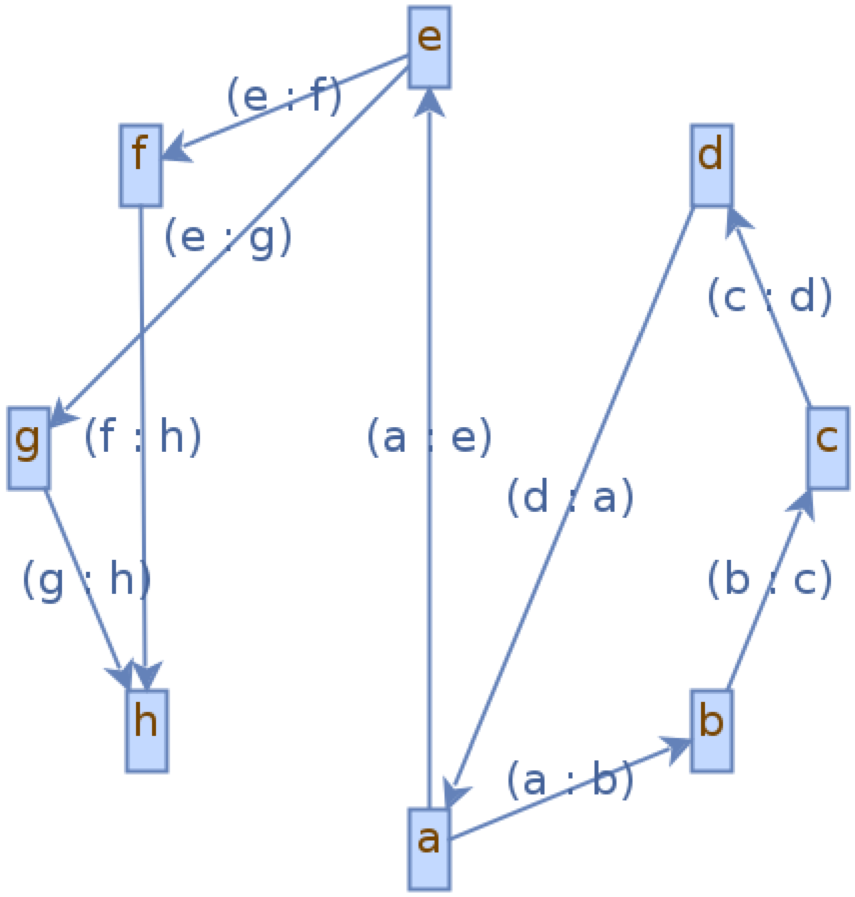
The Strategy Pattern involves:

- Defining a strategy interface (`SearchStrategy`) that declares a method signature for the algorithm.
- Creating concrete strategy classes (`BFS` and `DFS`) that implement the strategy interface with their specific algorithm implementations.

- Implementing a context class (`Context`) that holds a reference to the strategy interface and utilizes it to execute the selected strategy.
- Utilizing the `Context` class in the main code to dynamically switch between different strategies (`BFS` or `DFS`) based on input.

Random Walk Search

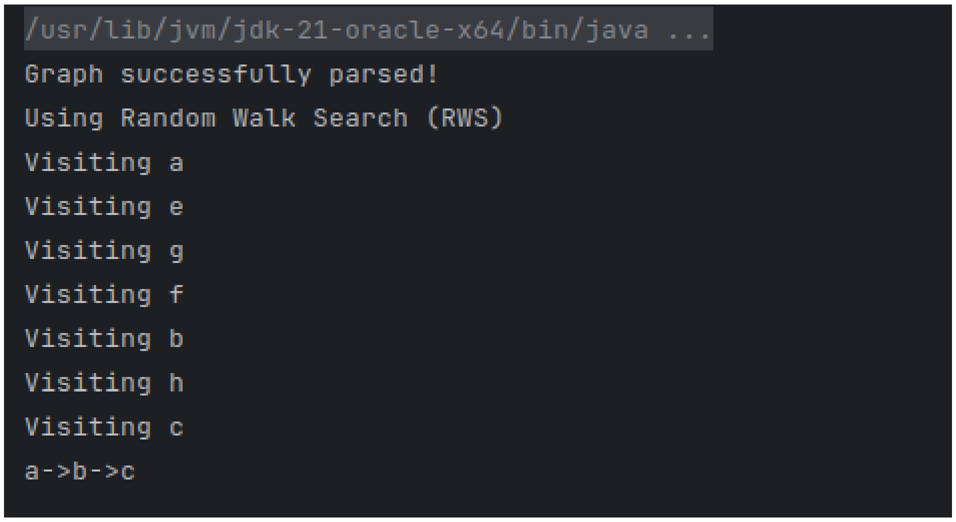
Exampe graph (from Canvas)



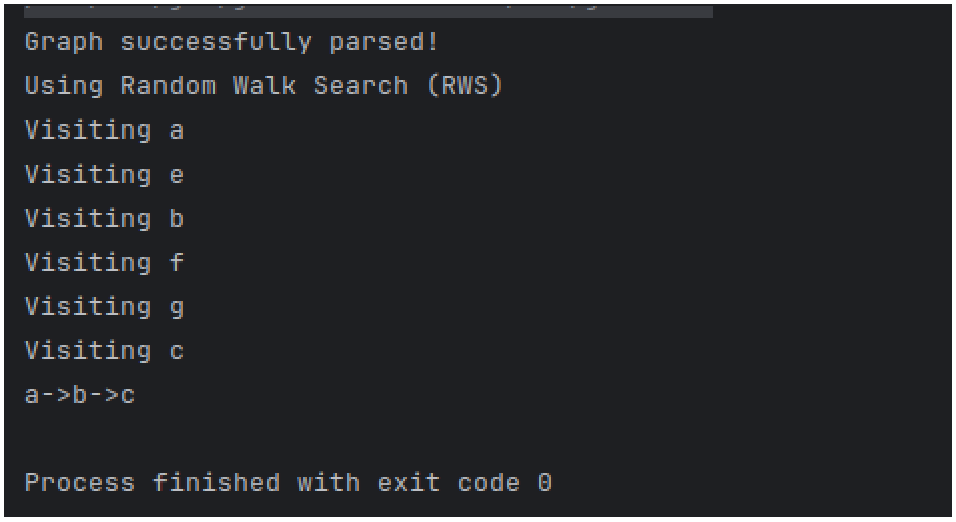
Running this code,

```
Path path = graphApi.GraphSearch("a","c", Algorithm.RWS);
path.printPath();
```

gives the following outputs -

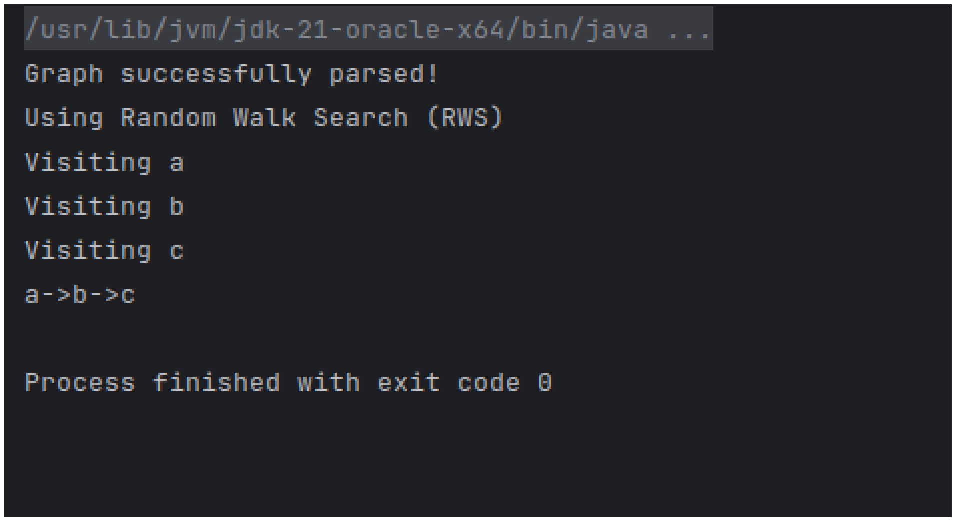


```
/usr/lib/jvm/jdk-21-oracle-x64/bin/java ...
Graph successfully parsed!
Using Random Walk Search (RWS)
Visiting a
Visiting e
Visiting g
Visiting f
Visiting b
Visiting h
Visiting c
a->b->c
```



```
Graph successfully parsed!
Using Random Walk Search (RWS)
Visiting a
Visiting e
Visiting b
Visiting f
Visiting g
Visiting c
a->b->c

Process finished with exit code 0
```



```
/usr/lib/jvm/jdk-21-oracle-x64/bin/java ...
Graph successfully parsed!
Using Random Walk Search (RWS)
Visiting a
Visiting b
Visiting c
a->b->c

Process finished with exit code 0
```

Commits

refactors

- refactor: Encapsulate graphObject and create a getter function.
- refactor: Rename opt variable to graphString.

- refactor: Convert test filepaths to static final variables.
- refactor: Add nullcheck in printPath function.
- refactor: Add comments in multiple functions

Template & Strategy patterns

- Implement template pattern for graph search algorithms.
- Add strategy pattern to graph search functionality.

Random Walk Search

- Add Random Walk search algorithm as option.

PR Review commits

- Extract getPath function to template class.
- Add random walk search tests.
- Add input test dot file for RWS test.