Readme for Part-3

Part - 1

- Refactoring 1: Adding region in GraphManager.java file to group APIs by features as mentioned in Project Part – 1
    - https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/e174e972a42ee4af7b5d634e8e4493bae41fe32d

```
//region Feature4
public void outputDOTGraph(String filepath) throws Exception{
    String text = g.outputDOTGraph();
    BufferedWriter writer = new BufferedWriter(new FileWriter(filepath));
    writer.write(text);
    writer.close();
}

public void outputGraphics(String path, String format) throws Exception{
    g.outputGraphics(path, format);
}
//endregion
```

- Refactoring 2: Removing unnecessary white spaces, grouping code performing similar logic, and making declaration at starting of APIs as mentioned in the below image for all the code.
    - https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/609be668db00530ab1b754e21ff2c29ee65a6526

```
public Path graphSearch( Graph g, String src, String dst, Algorithm algo){
    StringBuilder sb  = new StringBuilder();
    List<String> pathFound = null;
```

- Refactoring 3: Removing unnecessary code like commented lines, Printing messages, etc. for example: commented lines shown in below image throughout the code for better code readability.
    - https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/c17036345b9089592fe753b6106e61778a6ac740

```java
@Test
public void testOutputGraphicsJPG() throws Exception {
    String expectedFile = "expected.jpg";
    String actualFile = "actual.jpg";

    g.outputGraphics(actualFile, format: "jpg");
  //  g.outputGraphics(expectedFile, "jpg");

    BufferedImage actualImage = ImageIO.read(new File(actualFile));
    DataBuffer actualDataBuffer = actualImage.getData().getDataBuffer();
    int actualImageSize = actualDataBuffer.getSize();

    BufferedImage expectedImage = ImageIO.read(new File(expectedFile));
    DataBuffer expectedDataBuffer = expectedImage.getData().getDataBuffer();
    int expectedImageSize = expectedDataBuffer.getSize();

    Assert.assertEquals(expectedImageSize, actualImageSize);
    if (actualImageSize == expectedImageSize) {
        for (int i = 0; i < actualImageSize; i++) {
            Assert.assertEquals(expectedDataBuffer.getElem(i), actualDataBuffer.getElem(i));
        }
    }
}
```

- Refactoring 4: Renamed variable to their meaningful names like below image.
  - https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/0fb8401076c56ac4f1fa4b9e853be2f12674ecef

```java
public class GraphManager {

    Graph graph;
    Path path;

    public GraphManager (){
        graph = new Graph();
        path = new Path();
```

- Refactoring 5: Adding comments throughout the code where necessary for better understanding of code.
  - https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/3b1303dfd2b0acab69da029d2309986afc3f8263

```java
    }

    // testing outputDotGraphics() method Unsupported types thrpws error or not
    @Test(expected= IOException.class)
    public void testOutputGraphicsUnSupportedTypes() throws Exception {
        String actualFile = "actual.pdf";
        g.outputGraphics(actualFile, format: "pdf");
    }
}
```

Part – 2

- I created an abstract class “Path.java” which contains the common variables names, common method “graphSearch”, and one toString() method used by both the classes.
- I created a class named “BFS” which inherits the Path class and graphSearch method is implemented there. This class uses all the variables declared on Path class (parent class) and also toString() method to convert the Path into String.
- I created a class named “BFS” which inherits the Path class and graphSearch method is implemented there. This class uses all the variables declared on Path class (parent class) and also toString() method to convert the Path into String.
- This part is implemented into 2 commits for which links are given below:
    - Commit 1: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/223cf41c9397200391a6c9d1601408f3e8314ad7
    - Commit 2: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/51597135395065b5d67fd8ce189a54a386558d90
- Code of abstract class is given below: And whole code can be found in the commits.

```
abstract class Path {
    public String path = null;
    List<String> pathFound = null;
    StringBuilder sb  = new StringBuilder();
    List<String> pathList = new ArrayList<>();
    int[] visited = new int[26];

    public Path() {
    }

    public abstract Path graphSearch(Graph g, String src, String dst, Algorithm algo) ;

    public Path(String path){
        this.path = path;
    }

    public void listToString(){
        if(pathFound!=null){
            for(String node : pathFound){
                sb.append(node + "->");
            }
```

Part – 3

- To implement strategy pattern in SO that we can choose BFS or DFS based on the algo parameter in the API: Path GraphSearch(Node src, Node dst, Algorithm algo), I created a Search class which takes a variable type of Path. So if user chooses the in parent class variable child class object will be assigned and respective class graphSearch method will be implemented.
- I have attached the screenshot of searchClass and GraphSearch API below.
- Commit Id: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/6085fd57ca7f502bdfad386ef89a8b1d37bd6c5d

```
class Search{
    Path path;
    public Search(Path path){
        this.path = path;
    }

    public Path graphSearchByAlgo(Graph g, String src, String dst, Algorithm algo){
        return path.graphSearch(g, src, dst, algo);
    }
}
```

```
    //region GraphSearch
@   public Path graphSearch(String src, String dst, Algorithm algo) throws Exception{
        Search search = null;

        if(algo.toString().equals("BFS")){
            search = new Search(new BFS());
        }else if(algo.toString().equals("DFS")){
            search = new Search(new DFS());
        }else{
            search = new Search(new Random());
        }

        return search.graphSearchByAlgo(graph, src, dst, algo);
    }
    //endregion
}
```

Part -4:

- Commit id: https://github.com/DhavalPatodiya/CSE-464-2023-dpatodiy/commit/5da6a86904a7f0dc87d47ee28f6dbd644cfa2e6a

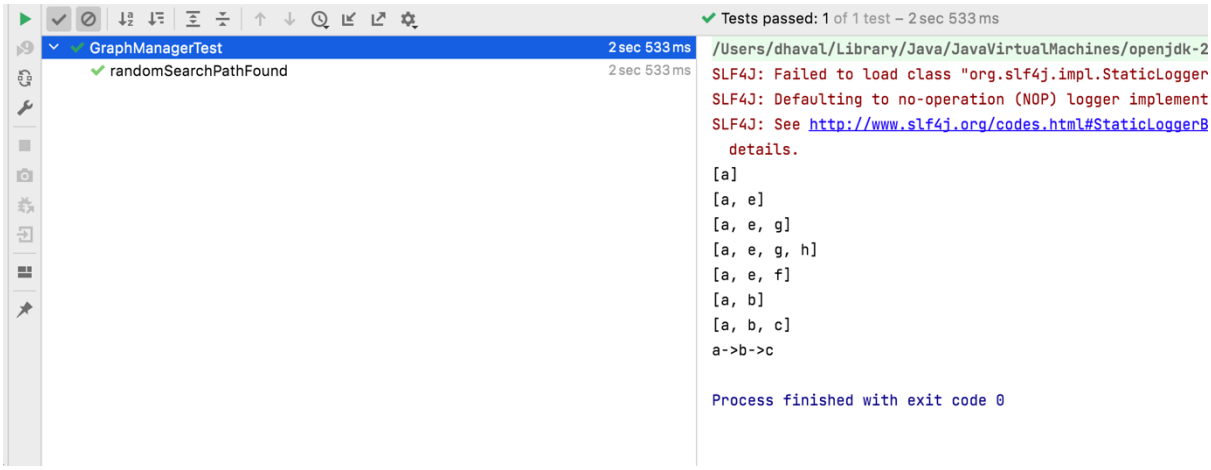
- Screenshot of randomSearch logic

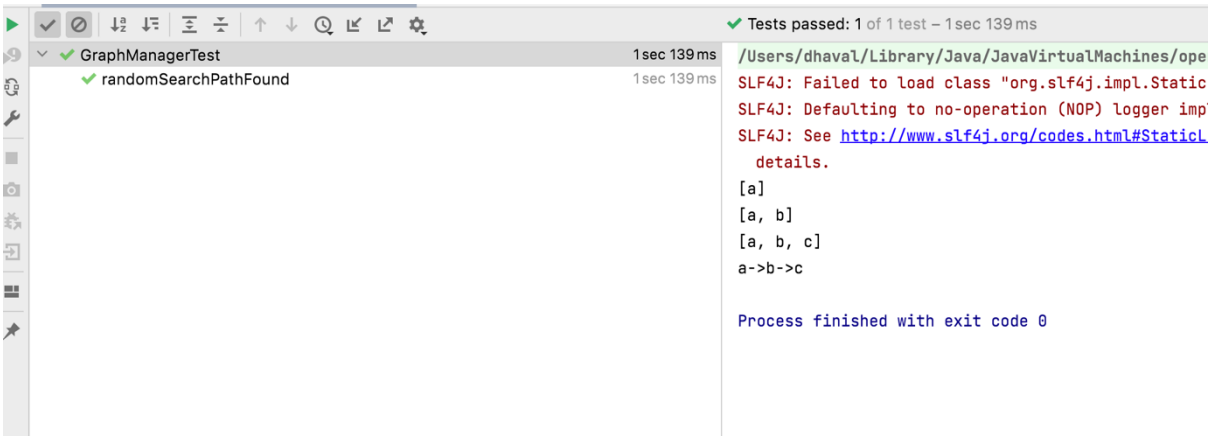
```
public List<String> algo(Graph g,String curr, String dst, int[] visited, List<String> path){
    System.out.println(path.toString());
    if(dst.equals(curr)){
        return path;
    }

    List<String> unvisitedChild = new ArrayList<>();
    for(String child : g.map.get(curr)){
        if(!(visited[child.charAt(0) - 'a'] == 1)){
            visited[child.charAt(0) - 'a'] = 1;
            unvisitedChild.add(child);
        }
    }

    while(!unvisitedChild.isEmpty()) {
        int size = unvisitedChild.size();
        double i = Math.random() * size;
        String last = unvisitedChild.remove((int)i);
        List<String> newPath = new ArrayList<>(path);
        newPath.add(last);

        List<String> cpath = algo(g, last, dst, visited, newPath);
        if(cpath!=null){
            return cpath;
        }
    }

    return null;
}
}
```

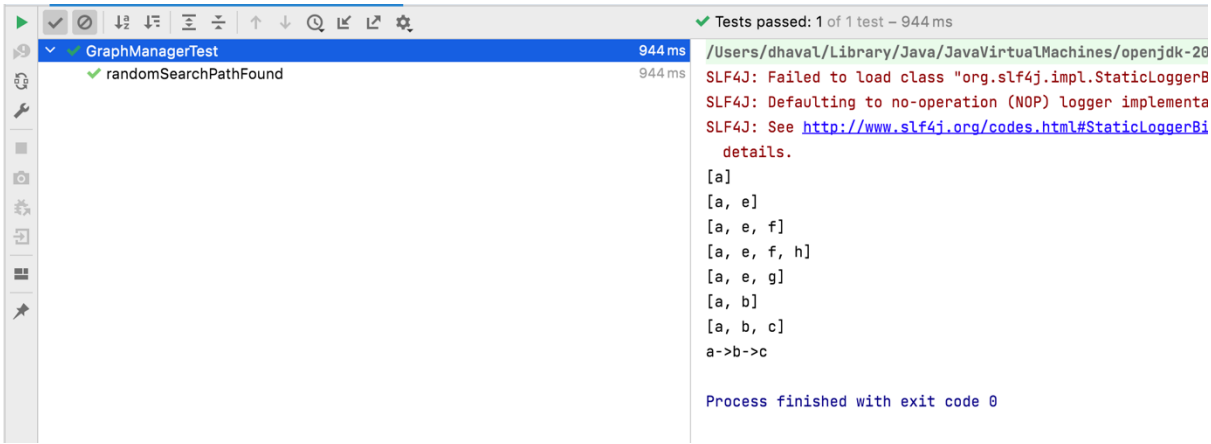
- Created a Random class and implemented the logic as shown above in that class.
- Random class inherits the DFS class, as the graphSearch method is same for both the class. Algo method of the DFS class is override in random class to implement randomsearch algorithm.
- Random class is called from graphSearch in the same way as the DFS and BFS.
- I have renamed the dot file given by professor to ‘randomsearch.dot’ file to avoid conflict with original ‘input.dot’ files.
- Driver test case or code for the graph is given below and written in GraphManagerTest class file.

```
// a-c path exists
@Test
public void randomSearchPathFound() throws Exception{
    Path actual = randomg.graphSearch( src: "a", dst: "c", Algorithm.RANDOM);
    System.out.println(actual.toString());
}
```

- Example outputs of random search:


Tests passed: 1 of 1 test – 2 sec 533 ms
GraphManagerTest 2 sec 533 ms
randomSearchPathFound 2 sec 533 ms
/Users/dhaval/Library/Java/JavaVirtualMachines/openjdk-2
SLF4J: Failed to load class "org.slf4j.impl.StaticLogger
SLF4J: Defaulting to no-operation (NOP) logger implement
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerB
details.
[a]
[a, e]
[a, e, g]
[a, e, g, h]
[a, e, f]
[a, b]
[a, b, c]
a->b->c
Process finished with exit code 0


Tests passed: 1 of 1 test – 1 sec 139 ms
GraphManagerTest 1 sec 139 ms
randomSearchPathFound 1 sec 139 ms
/Users/dhaval/Library/Java/JavaVirtualMachines/ope
SLF4J: Failed to load class "org.slf4j.impl.Static
SLF4J: Defaulting to no-operation (NOP) logger imp
SLF4J: See http://www.slf4j.org/codes.html#StaticL
details.
[a]
[a, b]
[a, b, c]
a->b->c
Process finished with exit code 0


Tests passed: 1 of 1 test – 944 ms
GraphManagerTest 944 ms
randomSearchPathFound 944 ms
/Users/dhaval/Library/Java/JavaVirtualMachines/openjdk-20
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerB
SLF4J: Defaulting to no-operation (NOP) logger implementa
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBi
details.
[a]
[a, e]
[a, e, f]
[a, e, f, h]
[a, e, g]
[a, b]
[a, b, c]
a->b->c
Process finished with exit code 0