Homework 8

Source Code:

```
package cse41321.containers;

import cse41321.algorithms.graph.BreadthFirstSearch;

public class Homework8 {

    public static boolean isExitReachable(
        Graph<BreadthFirstSearch.Server, Integer> maze,
        char entrance,
        char exit)
    {
        BreadthFirstSearch.countNetworkHops(maze, Character.toString(entrance));
        // if exist a path
        if(maze.getVertex(new
BreadthFirstSearch.Server(Character.toString(exit))).getData().getHops() >= 0){
            return true;
        }
        // if no path existing
        else{
            return false;
        }
    }

    private static Graph<BreadthFirstSearch.Server, Integer> maze1;
    private static Graph<BreadthFirstSearch.Server, Integer> maze2;
    public static void main(String[] args) {
        // nodes for maze 1
        BreadthFirstSearch.Server node1a = new BreadthFirstSearch.Server("A");
        BreadthFirstSearch.Server node1b = new BreadthFirstSearch.Server("B");
        BreadthFirstSearch.Server node1c = new BreadthFirstSearch.Server("C");
        BreadthFirstSearch.Server node1d = new BreadthFirstSearch.Server("D");
        BreadthFirstSearch.Server node1e = new BreadthFirstSearch.Server("E");
        BreadthFirstSearch.Server node1f = new BreadthFirstSearch.Server("F");
        BreadthFirstSearch.Server node1g = new BreadthFirstSearch.Server("G");
        // nodes for maze2
        BreadthFirstSearch.Server node2a = new BreadthFirstSearch.Server("A");
        BreadthFirstSearch.Server node2b = new BreadthFirstSearch.Server("B");
        BreadthFirstSearch.Server node2c = new BreadthFirstSearch.Server("C");
        BreadthFirstSearch.Server node2d = new BreadthFirstSearch.Server("D");
        BreadthFirstSearch.Server node2e = new BreadthFirstSearch.Server("E");
        BreadthFirstSearch.Server node2f = new BreadthFirstSearch.Server("F");
        BreadthFirstSearch.Server node2g = new BreadthFirstSearch.Server("G");

        // edges of maze1
        maze1.insertEdge(node1a, node1c, 0);
        maze1.insertEdge(node1c, node1a, 0);
        maze1.insertEdge(node1a, node1d, 0);
        maze1.insertEdge(node1d, node1a, 0);
```

```
        maze1.insertEdge(node1b, node1d, 0);
        maze1.insertEdge(node1d, node1b, 0);
        maze1.insertEdge(node1c, node1f, 0);
        maze1.insertEdge(node1f, node1c, 0);
        maze1.insertEdge(node1d, node1e, 0);
        maze1.insertEdge(node1e, node1d, 0);
        maze1.insertEdge(node1d, node1g, 0);
        maze1.insertEdge(node1g, node1d, 0);
        maze1.insertEdge(node1e, node1d, 0);
        maze1.insertEdge(node1d, node1e, 0);
        maze1.insertEdge(node1e, node1g, 0);
        maze1.insertEdge(node1g, node1e, 0);
        maze1.insertEdge(node1f, node1g, 0);
        maze1.insertEdge(node1g, node1f, 0);
        //edges of maze2
        maze2.insertEdge(node2a, node2c, 0);
        maze2.insertEdge(node2c, node2a, 0);
        maze2.insertEdge(node2a, node2d, 0);
        maze2.insertEdge(node2d, node2a, 0);
        maze2.insertEdge(node2b, node2d, 0);
        maze2.insertEdge(node2d, node2b, 0);
        maze2.insertEdge(node2c, node2f, 0);
        maze2.insertEdge(node2f, node2c, 0);
        maze2.insertEdge(node2e, node2g, 0);
        maze2.insertEdge(node2g, node2e, 0);

        System.out.println(isExitReachable(maze1, 'A', 'G'));
        System.out.println(isExitReachable(maze2, 'A', 'G'));
    }

}
```

Output:

True
False