Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 8 REPORT

STUDENT NAME
Can BEYAZNAR
STUDENT NUMBER
161044038

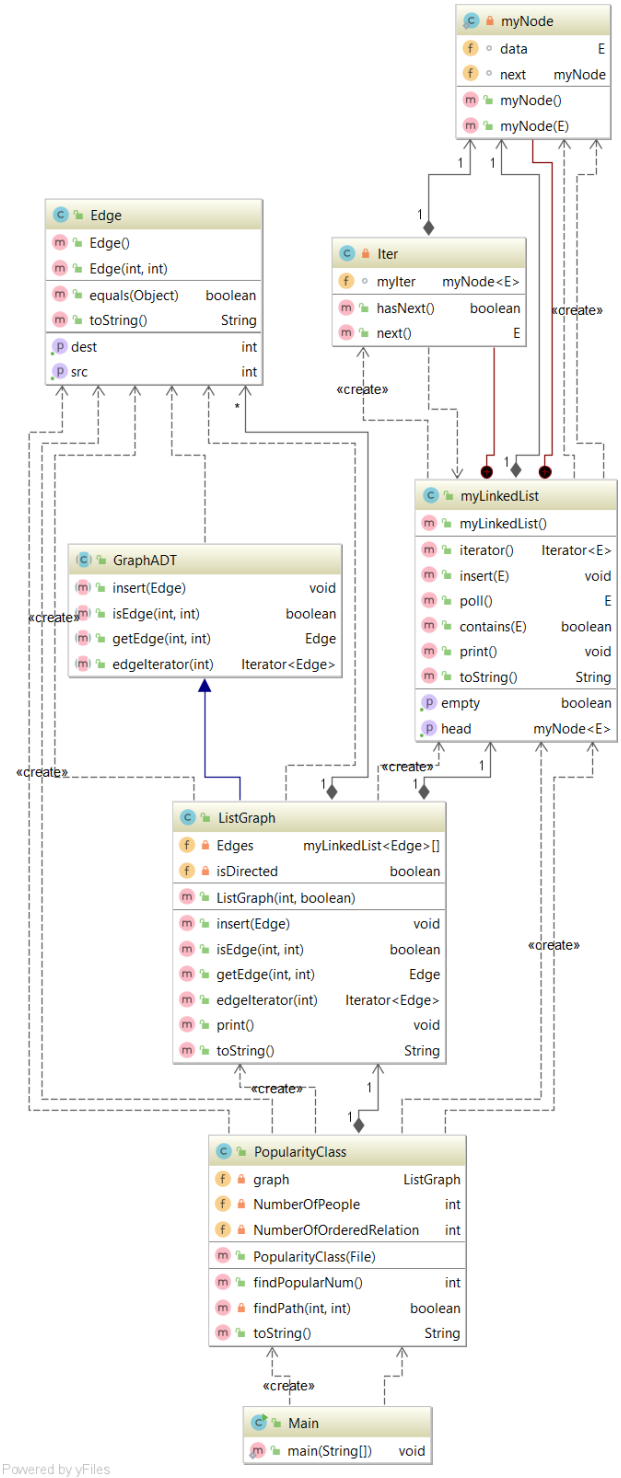
1 INTRODUCTION

1.1 Problem Definition

In this homework, we have an edge, let’s say our source element is A, and the destination element is B, for this homework we have to think that, A thinks B is popular. And in this homework we will read every edges from input file one by one and we will insert the to our graph. Our main problem is, we have to find the number of people who are considered popular by every other person. Also, let’s say we have (A,B) and (B,C) the A element additionally thinks B and C is popular (Transitive). And finally we will print the result.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

In this assignment, we need to create the Edge class to keep the elements that will be given to us. This simple class holds only two integer elements. Then we have to create the ListGraph class to keep all the elements bound to the Edge class. To create the ListGraph class, we have to extend the abstract GraphADT class that we created earlier. And then we need to override all the methods in the GraphADT class. Before overriding these methods, we need to create the myLinkedList class to help with our ListGraph class. This class is like the same LinkedList is also iterable. The ListGraph class holds the isDirected element in the boolean type and the Edges element in the myLinkedList <Edge> [] type. In the constructor of the ListGraph class, the number of elements to be added as parameters and whether they are in the front or not will be taken. Then the number of elements to be added in the Edges array is reserved. And then we implement our methods like the Graph logic, our ListGraph class is implemented.

2.2.1 PopularityClass Class

In this class, we have to find our result. The ListGraph-type graph and the NumberOfPeople, NumberOfOrderedRelation elements in two integer types are created in this class. The constructor method of our class takes a file as a parameter. In this file, in the first line we have the number of elements that the user wants to add and the number of links, and int the other lines we have the elements that he wants to add are included in the file. After obtaining these elements, the elements in our class are kept. Then all the elements that the user wants to add are added to the graph. After the addition, the last operation remains. Before explaining the main method, it is better to explain the ancillary method. The findPath method will take two integer parameters (which is firstInt and secondInt). And we will return true if there is a way between these two integers in our graph. And after implementing this auxiliary method we will implement our main methof which finds the result. The findPopularNum() method takes no parameter. In this method, firstly we will create an integer array (Let’s say its name PointedElements) which will keep the number of elements connected to it. And after that we will call findPath method in a nested loop and we will try all possibilities. If there is a path between A to B and A is not equal to

B, we will increase PointedElements[B-1] once. And after this nested loop ends, we will look every elements of the PointedElements array one by one. If the (PointedElements[i] >= NumberOfPeople-1) is true increase result once. After the end of this loop we finally find our result. And return the result. The time complexity of findPath method is O(n^2), and findPopularNum's time complexity is O(n^4).

2.3 Time Complexity

2.3.1 Time Complexity of ListGraph

insert()	isEdge()	getEdge()	edgeIterator()	print()
O(1)	O(n)	O(n)	O(1)	O(n)

3 RESULT

3.1 Test Cases

```java
/**
 * @author Can BEYAZNAR 161044038
 */

import java.io.File;
import java.io.FileNotFoundException;

public class Main {

    public static void main(String args[]) throws FileNotFoundException {

        PopularityClass TEST = new PopularityClass(new File( pathname: "src/input.txt"));

        System.out.println(TEST);

        int Result = TEST.findPopularNum();
        System.out.println(Result);
    }

}
```

Main.java | input.txt

```
3 3
1 2
2 1
2 3
```

3.2 Running Results

```
Vertex 1's elements are: Src : 1 Dest : 2
Vertex 2's elements are: Src : 2 Dest : 3, Src : 2 Dest : 1

1

Process finished with exit code 0
```