

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 8 REPORT

**ALI HAYDAR KURBAN
151044058**

Course Assistant: Ayse Serbetci Turan

1 INTRODUCTION

1.1 Problem Definition

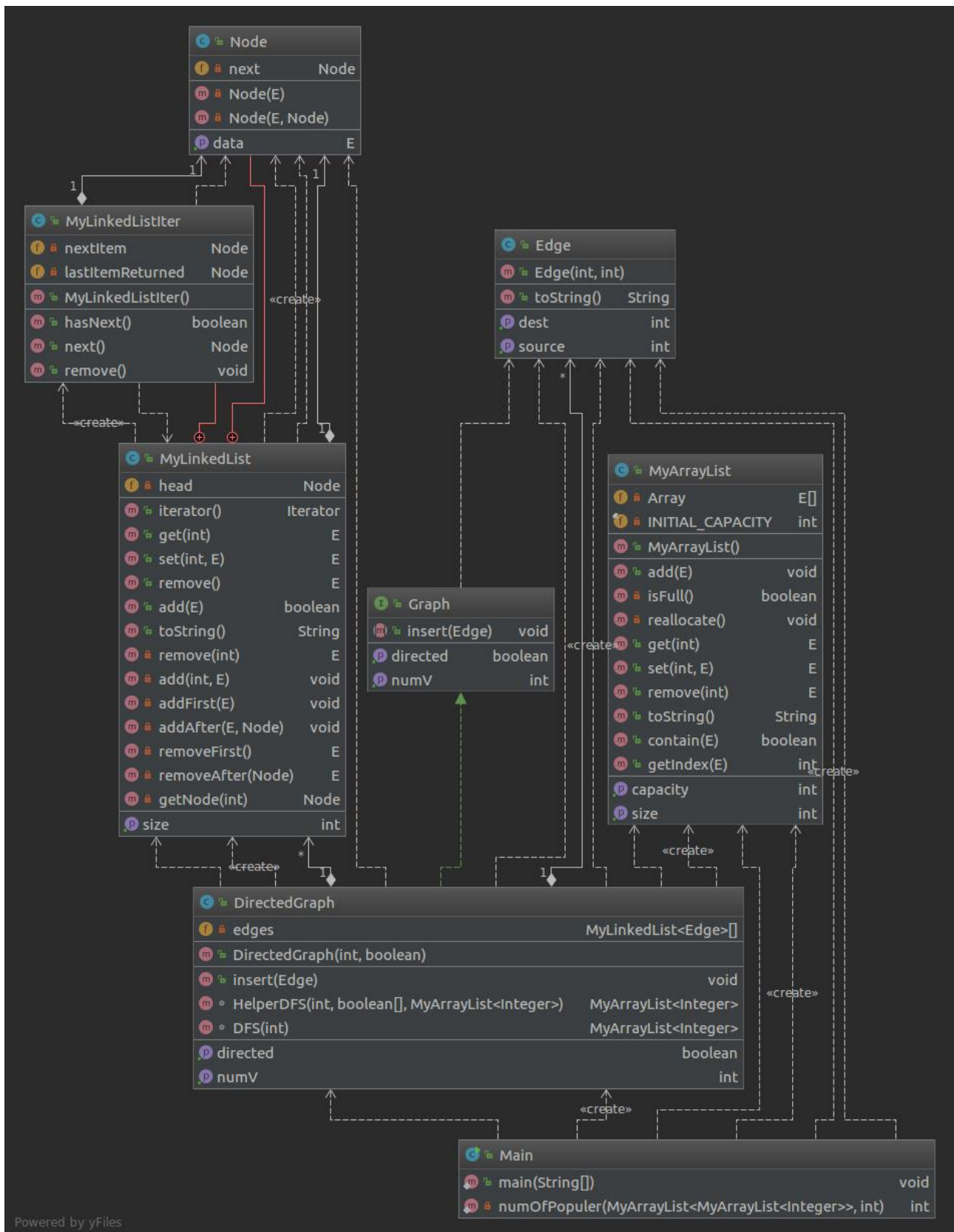
The problem is implementing your graph ADT and searching in this graph. There are some people and they show a person which is popular for them. Let's think 5 people, if a person is popular for 4 people, this person is the most popular person. In this homework I tried to find numbers of the most popular people.

1.2 System Requirements

The solution does not require a hardware, or certain minimal amount of memory. To solve this problem I used some helper data structures. These are LinkedList and ArrayList. These are implemented by me. I used some Java libraries, these are java.io.File, java.io.FileNotFoundException, java.util.Scanner, java.util.NoSuchElementException and java.util.Iterator. It can work on a machine which has JVM.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

User have to create a txt file with name "input.txt". In this file there are some numbers. The meaning of numbers is following:

X Y

A B

B A

B C

Meaning of X is number of numbers.

Meaning of Y is number of ordered relation. (A thinks B is popular, B thinks A and C are popular, so that A thinks C is popular.)

User must be careful about the numbers. Numbers have to be positive. User use only numberOfPopular method. This method finds the numbers of most popular person.

2.3 Problem Solution Approach

To solve this problem I implemented a graph. To implement graph I chose List methods. So that I implemented LinkedList data structure. And I implemented ArrayList data structure to help me. There is a graph interface, it shows that the methods of graph. These methods are getNumV, isDirected, insert, and DFS. Lets explain these methods. getNumV returns the number of vertex in the graph, isDirected returns the graph is directed or not, insert add vertex to graph, DFS makes a deep first search and returns a ArrayList which is implemented by me. I create a class with name Edge, it holds numbers of input file. Like source and destination. In Main, first of all I read the input file, then for all line I create a new Edge reference. Then I added into a ArrayList of Edge. I found the number of vertex. It is inside of first Edge reference. Then I create a graph reference then insert the Edges. Then I run the DFS method, it returned a ArrayList, I create another ArrayList which holds ArrayList reference, then I add all result of DFS (It means deep first search). To reach the goal I have to make these ArrayList unique, so that I did unique operation and add a new real ArrayList reference these ArrayList is unique. Then I run numOfPopular and I found the numbers of most popular numbers. This method takes an ArrayList reference and number of vertex. Lets explain the numOfPopular. This method creates a new ArrayList and assigned the first ArrayList of argument of this method. In a loop with range of the size of new ArrayList, I control other ArrayList if it contains the number or not, if it contains the number I increased the temp_counter, if it reaches the number of vertex – 1 it is most popular then I increased the counter which is return value of this method. Lets analyses the time complexity of methods. In LinkedList, best case adding, removing... $O(1)$ worst case $O(n)$, in ArrayList get and set $O(1)$ adding and removing are $O(1)$ in best case, in the worst case they are $O(n)$. Graph methods are in the best case $O(1)$, in the worst case $O(n^2)$.

3 RESULT

3.1 Test Cases

First of all I tested my program with the given input file in homework pdf. Then I test different cases. I understood if program is correct or not in this way: I write a DFS method and print the result I controlled it each numbers. If a there is 5 numbers and a number is inside 4 DFS that means it is most popular.

My test inputs are following:

Test 1:

3 3

1 2

2 1

2 3

Test 2:

8 8

8 7

1 3

2 3

4 3

5 3

3 4

6 3

7 6

Test 3:

4 4

1 2

2 3

4 2

1 4

Test 4:

6 6

6 5

2 1

3 1

4 1

1 3

5 1

3 5

Test 5:

4 4

1 3

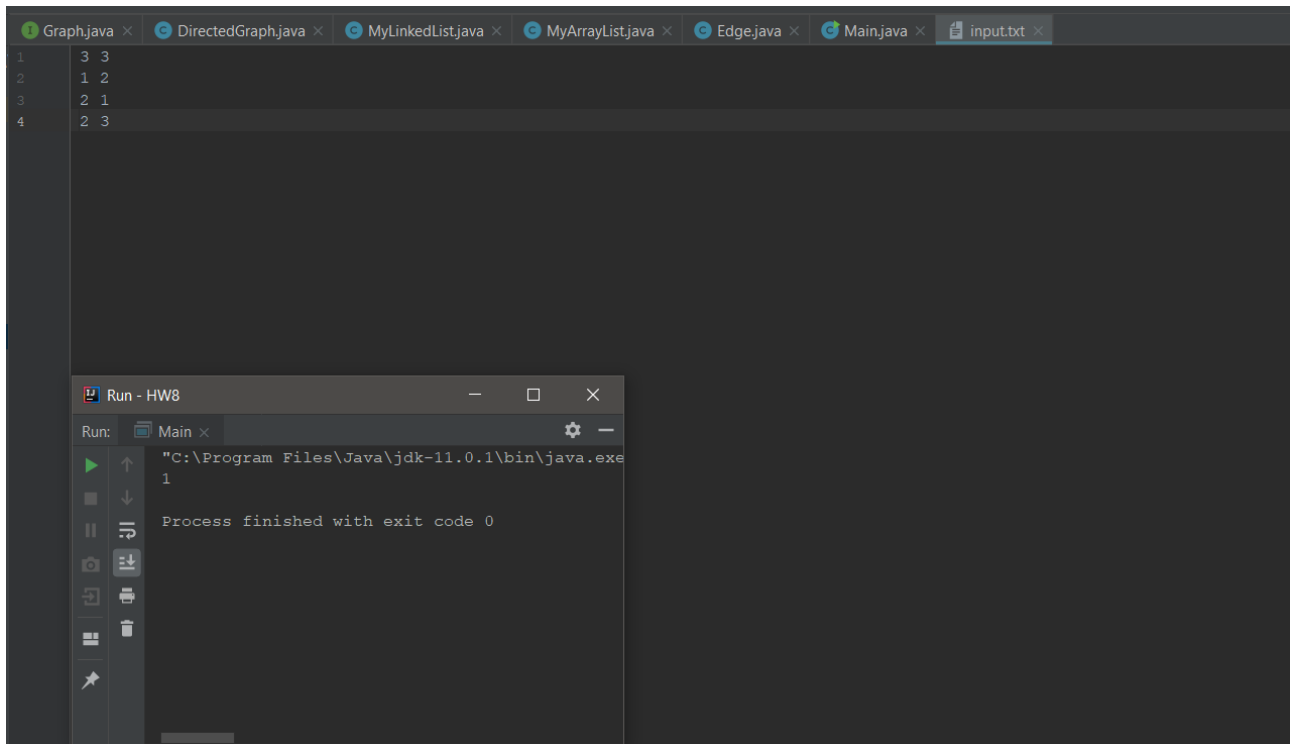
3 1

2 4

4 2

3.2 Running Results

Result 1:



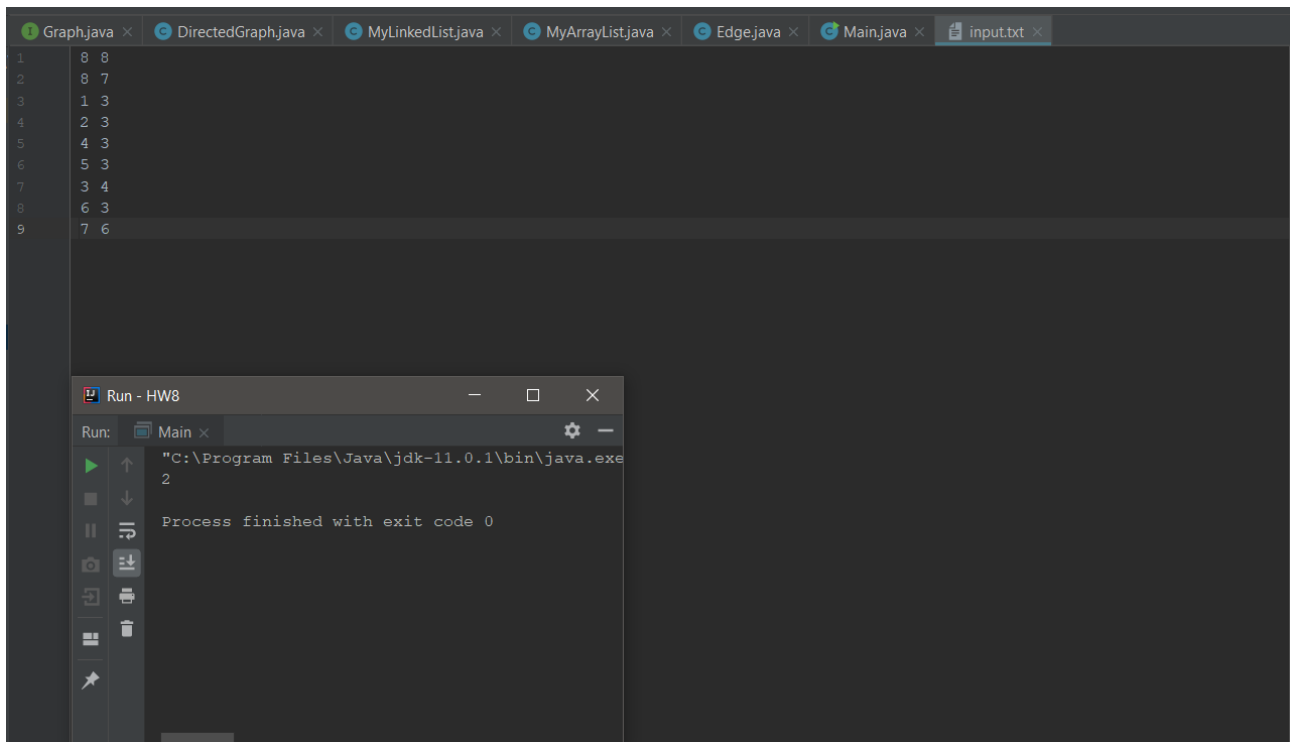
```
1 3 3
2 1 2
3 2 1
4 2 3
```

Run - HW8

Run: Main ×

"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"
1
Process finished with exit code 0

Result 2:



```
1 8 8
2 8 7
3 1 3
4 2 3
5 4 3
6 5 3
7 3 4
8 6 3
9 7 6
```

Run - HW8

Run: Main ×

"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"
2
Process finished with exit code 0

Result 3:

```
1 4 4
2 1 2
3 2 3
4 4 2
5 1 4
```

Run - HW8

Run: Main x

"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"
1
Process finished with exit code 0

Result 4:

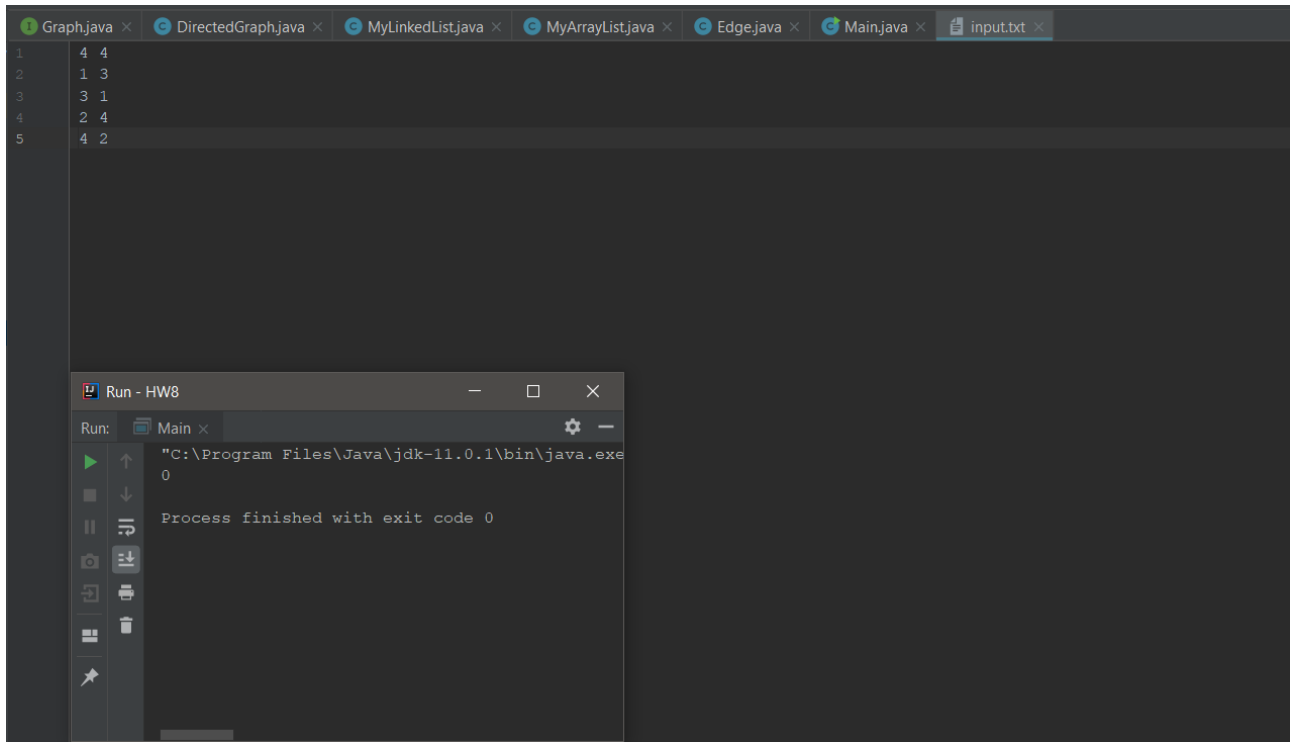
```
1 6 6
2 6 5
3 2 1
4 3 1
5 4 1
6 1 3
7 5 1
8 3 5
```

Run - HW8

Run: Main x

"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"
3
Process finished with exit code 0

Result 5:



The screenshot shows an IDE with several tabs: Graph.java, DirectedGraph.java, MyLinkedList.java, MyArrayList.java, Edge.java, Main.java, and input.txt. The input.txt tab is active, displaying the following content:

```
1 4 4
2 1 3
3 3 1
4 2 4
5 4 2
```

Below the code editor, a 'Run - HW8' window is open, showing the execution details for 'Main'. The command executed is `"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"` with an exit code of 0. The output indicates that the process finished successfully.

```
Run: Main x
Process finished with exit code 0
```