

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 8 REPORT

**NURULLAH GÜNDOĞDU
161044108**

AYŞE ŞERBETÇİ TURAN

1 INTRODUCTION

1.1 Problem Definition

In this project, we have a group of people with a popular relationship. We are asked to find the number of people who are popular with everyone. This relationship "A" is created by the person "B" is popular. Or "A-B" and "B-C" in the case of the "A-C" is a popular find. Using these relationships, we can find out how many people are populated by each person.

1.2 System Requirements

You have to download IntelliJ IDEA for run the programs. The program requirements are listed below.

Microsoft Windows 10/8/7/Vista/2003/XP (incl.64-bit)

2 GB RAM minimum, 4 GB RAM recommended

1.5 GB hard disk space + at least 1 GB for caches

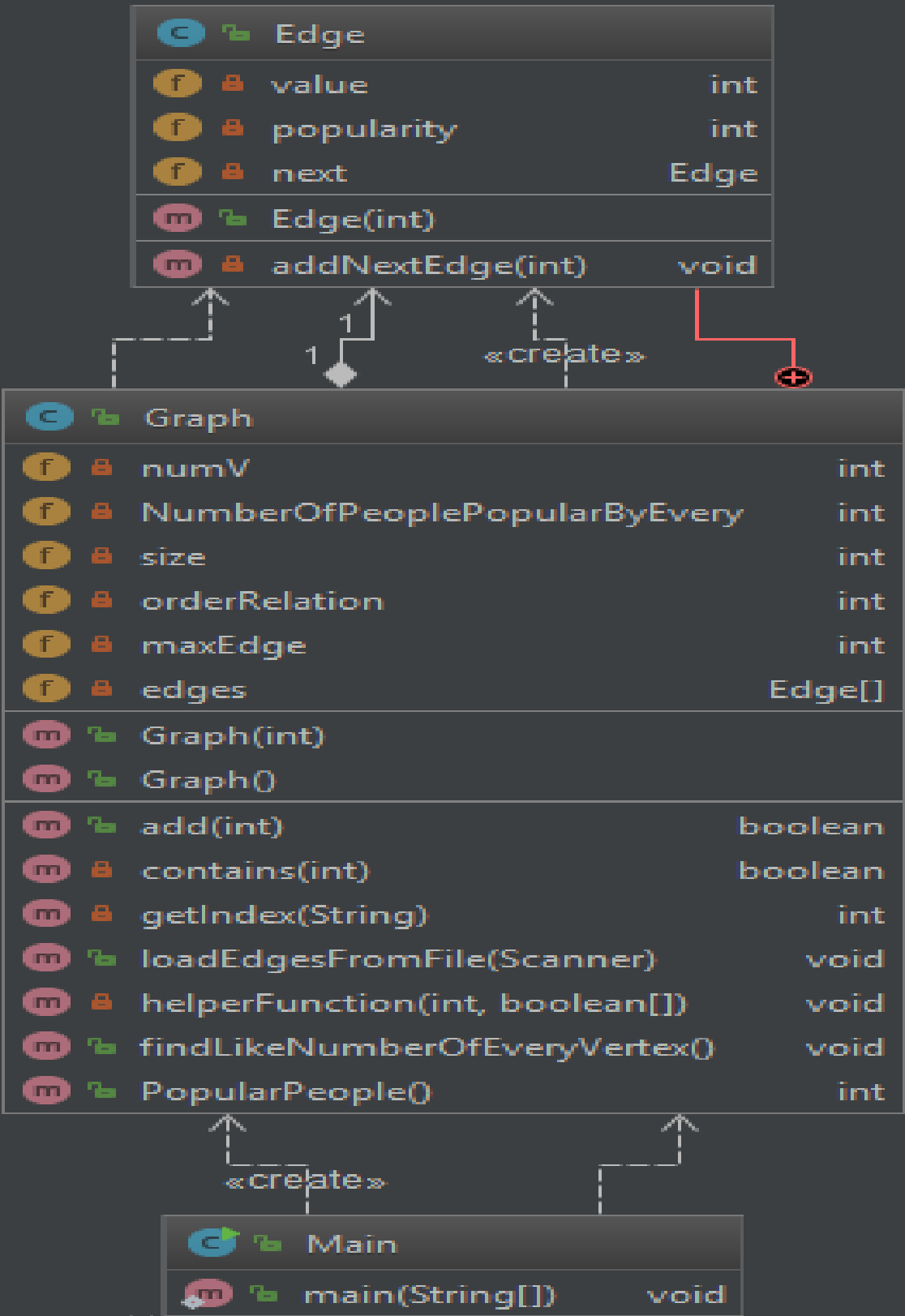
1024x768 minimum screen resolution

You can download program here;

<https://www.jetbrains.com/idea/download/#section=windows> (Community Version)

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

User have to give a file to program. File name must be "input.txt". In this file there should be vertex number, order relation number and relations.

Example;

input.txt

```
'Vertex number' 'order relation number'  
vertex1 vertex2  
vertex1 vertex3  
vertex3 vertex2  
vertex1 vertex4  
vertex4 vertex3  
....  
....
```

2.3 Problem Solution Approach

Firstly I read input file and allocate edges array. After that I read vertex number from file and add to edges array. I use hashcode to find index of every vertex. I write edge class for vertex. Edge class keeps name of vertex and popular people. There is a linked list structure in edge class, it keeps next edge. After all of that I find number of popularity every people. If person to find popular every other people, program increase NumberOfPeoplePopularByEvery. Finally program print number of popular people to the screen.

private void addNextEdge(int value):

This function adds edge who you think popular to given edge.

Time complexity :

n = Number of found popular people

$T(n) = O(n)$.

public boolean add(int people):

This function add edge to edges array.

Time complexity :

Best case

$T(1) = O(1)$.

Worst case

n = Number of people in graph

$T(n) = O(n)$.

private boolean contains(int value):

This function checks if the incoming person is in the graph.

Time complexity :

Best case

$$T(1) = O(1).$$

Worst case

n = Number of people in graph

$$T(n) = O(n).$$

private int getIndex(String value):

This function return index of given value. This function using hashcode to find index of value.

Time complexity :

Best case

$$T(1) = O(1).$$

Worst case

n = Number of people in graph

$$T(n) = O(n).$$

public void loadEdgesFromFile(Scanner scan):

This function reads graph size from scan and allocate the edges array. After that it reads vertex from scan and create a graph

Time complexity :

n = Number of people

$$T(n) = O(n).$$

public void findLikeNumberOfEveryVertex():

This function find number of found popular people of every vertex.

Time complexity :

n = Number of people

$$T(n) = O(n).$$

private void helperFunction(int people,boolean like[]):

This function helper function for findLikeNumberOfEveryVertex. It search every vertex to find popular people.

Time complexity :

n = Number of found popular people
 $T(n) = O(n)$.

public int PopularPeople():

This function return popular people number

Time complexity :

$T(1) = O(1)$.

3 RESULT

3.1 Test Cases

Test 1

```
1 7 10
2 1 2
3 2 1
4 2 3
5 3 4
6 4 5
7 4 6
8 5 7
9 7 6
0 7 5
1 6 5
```

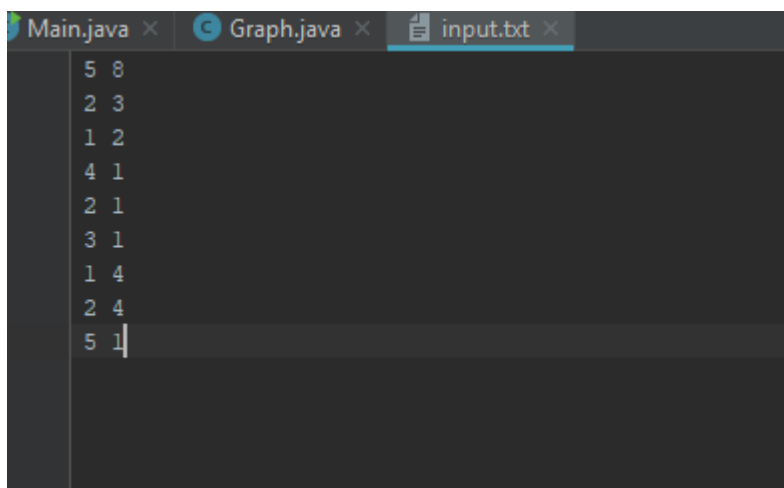
Test 2

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

Test 3

```
Main.java x Graph.java x inp
3 3
1 2
2 1
2 3
```

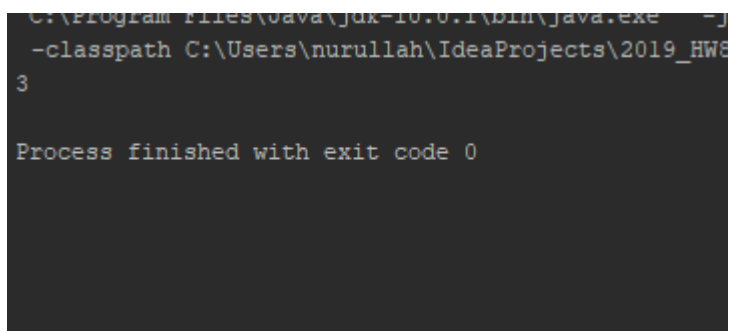
Test 4



```
Main.java × Graph.java × input.txt ×
5 8
2 3
1 2
4 1
2 1
3 1
1 4
2 4
5 1
```

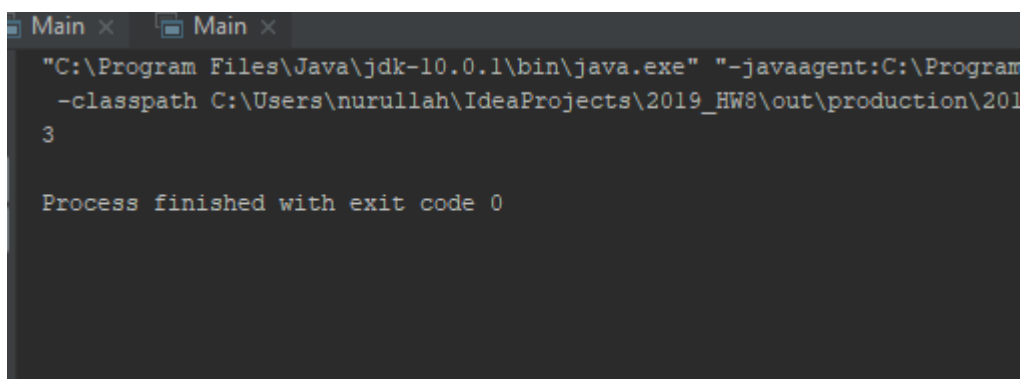
3.2 Running Results

Test 1 Result



```
C:\Program Files\Java\jdk-10.0.1\bin\java.exe -j
-classpath C:\Users\nurullah\IdeaProjects\2019_HW8
3
Process finished with exit code 0
```

Test 2 Result



```
Main × Main ×
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-javaagent:C:\Program
-classpath C:\Users\nurullah\IdeaProjects\2019_HW8\out\production\201
3
Process finished with exit code 0
```


Test 3 Result

```
"C:\Program Files\Java\jdk-10.0.1\bin\java
-classpath C:\Users\nurullah\IdeaProjects
1

Process finished with exit code 0
```

Test 4 Result

```
"C:\Program Files\Java\jdk-10.0.1\bin\java.exe" "-j
-classpath C:\Users\nurullah\IdeaProjects\2019_HW8
4

Process finished with exit code 0
```