

Object Oriented Programming with Java - Advanced Course

Project Documentation

Robin Müller-Bady

December 26, 2019

Contents

1. Introduction
2. Group Members
3. Project Documentation
4. Milestones
 - 4.1 Milestones1
 - 4.2 Milestones2

1. Introduction

2. Group Members

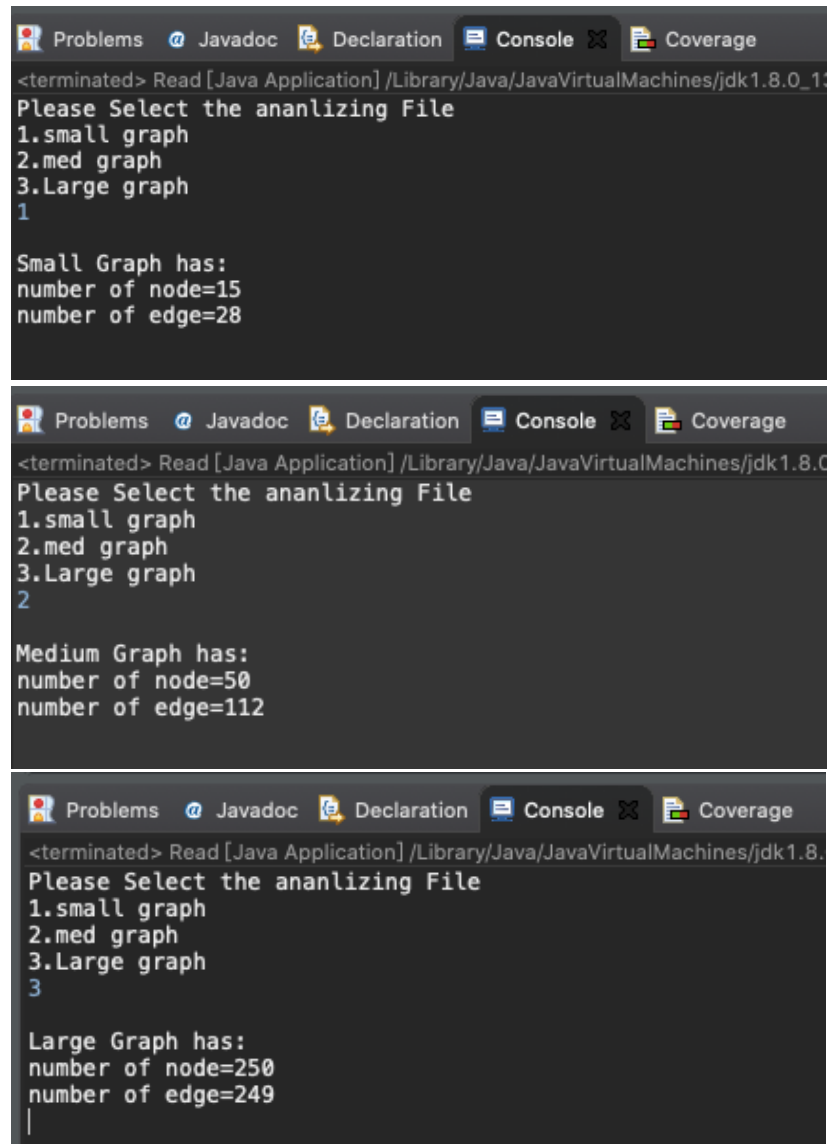
Name	Main Responsibilities
SU TZU CHENG	Programing
MAO YU QING	Programing
Marti Nuñez Muñoz	Cleaning
OH YEONSOO	Document Writing

3. Program Description

- Technical Description

The Description is shown in the comment of each screenshot

(1) Result:



```
<terminated> Read [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111
Please Select the analyzing File
1.small graph
2.med graph
3.Large graph
1

Small Graph has:
number of node=15
number of edge=28

<terminated> Read [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111
Please Select the analyzing File
1.small graph
2.med graph
3.Large graph
2

Medium Graph has:
number of node=50
number of edge=112

<terminated> Read [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_111
Please Select the analyzing File
1.small graph
2.med graph
3.Large graph
3

Large Graph has:
number of node=250
number of edge=249
|
```

(2) Main function

```
//Main Function
public static void main(String[] args) {
    BufferedReader objReader = null;

    try {
        Read Final = new Read();
        String mode = Final.Address();
        String data=Final.File(mode);
        System.out.print("number of node=");
        System.out.println(Final.NodeCount(data));
        System.out.print("number of edge=");
        System.out.println(Final.EdgeCount(data));
    }

    catch (IOException e) {

        e.printStackTrace();
    } finally {

        try {
            if (objReader != null)
                objReader.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

(3) Functions

1. Address deciding

```
/*Function 1 : String Address()>>> Select the Graph that needs to be analyzed

* There's three graph needs to be analyzed. This is the user menu to choose the graph.
* After the user select, this part of program will return the address to the File reading function.

String Address() {

    System.out.println("Please Select the analyzing File");
    System.out.println("1.small graph\n2.med graph\n3.Large graph");
    String mode=null;
    Scanner selectmod = new Scanner(System.in);
    String a =selectmod.nextLine();
    selectmod.close();
    if(a.contentEquals("2")) {
        System.out.println("\nMedium Graph has:");
        mode = "/Users/denzelsu/Downloads/medium_graph.graphml";
    }
    if(a.contentEquals("1")) {
        System.out.println("\nSmall Graph has:");
        mode = "/Users/denzelsu/Downloads/small_graph.graphml";
    }
    if(a.contentEquals("3")) {
        System.out.println("\nLarge Graph has:");
        mode = "/Users/denzelsu/Downloads/large_graph.graphml";
    }
    return mode;
}
```

2. File Reading

```
/*Function 2 : String File(String) >> Read the file from the computer

* Method:
* 1. input the address of the file returned from the previous function(address)
* 2. Use the StringBuffer to store all the data into a StringBuffer
* 3. Transfer the StringBuffer to String
* 4. Return the String

String File(String a) throws IOException {
    BufferedReader brd= new BufferedReader(new FileReader(a));
    StringBuffer sbf= new StringBuffer();
    String l=null;
    while((l= brd.readLine())!=null) {
        sbf.append(l);
    }
    brd.close();
    l=sbf.toString();
    return l;
}
```

3. Node Counting

a. Method 1

```
/*Function 3 : int NodeCount(String)>> The Algorithm to count the node
* This part of function is made for counting the node inside the graphML that selected.
*
* The method is:
* 1.String a is the graphML file that has already Transfer in the function File.
* 2. We scanned the file if it contains "node id", if it does counter(nodeee) add one and
* replace that "node id" to prevent the search next time
* 3. After we can't find any "node id" in the String a, return the counter(nodeee)
*/
int NodeCount(String a){
    int nodeee = 0;
    while(a.contains("node id")) {
        nodeee++;
        a = a.replaceFirst("node id", "HHH");
    }
    return nodeee;
}
```

b. Method 2

```
/*count nodes
* invoke readFile() to get the sbff(StringBuffer)
* use indexOf(String,fromIndex) to count the number of codes
* return the number of nodes
*/
@SuppressWarnings("null")
public static int culcuateNodes() throws IOException{
    StringBuffer fl = new StringBuffer(readFile());
    int countNode = 0;
    int index = 0;
    String findIndex="node id = ";
    while ((index = fl.indexOf("node id", index)) != -1) {
        index = index + findIndex.length();
        countNode++;
    }
    return countNode;
}
```

4. Edge Counting

a. Method 1

```
/*Function 4: int EdgeCount(String)>> The Algorithm to count the edge
* This part of function is made for counting the node inside the graphML that selected.
*
* The method is:
* 1.String a is the graphML file that has already Transfer in the function File.
* 2. We scanned the file if it contains "edge source", if it does counter(nodeee) add one
* and replace that "node id" to prevent the search next time
* 3. After we can't find any "node id" in the String a, return the counter(nodeee)
*/
int EdgeCount(String a){
    int edgeee = 0;
    while(a.contains("edge source")) {
        edgeee++;
        a = a.replaceFirst("edge source", "HHH");
    }
    return edgeee;
}
```

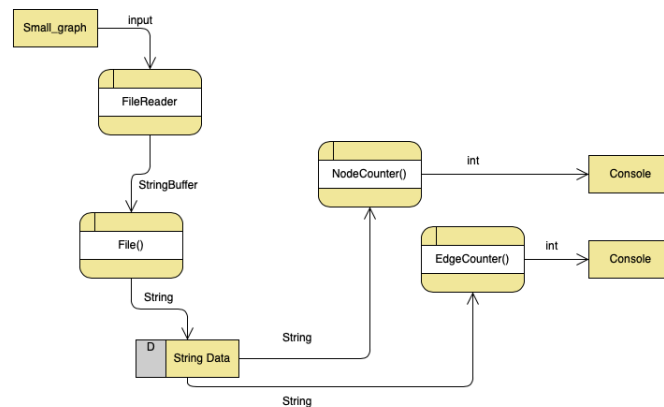
b. Method 2

```
/*count edges
* invoke readFile() to get the sbff(StringBuffer)
* use indexOf(String,fromIndex) to count the number of edges
* return the number of edges
*/
public static int culcuateEdges() throws IOException{
    StringBuffer fl = new StringBuffer(readFile());
    int countEdges = 0;
    int index = 0;
    String findIndex = "edge source = ";
    while ((index = fl.indexOf("edge source", index)) != -1) {
        index = index + findIndex.length();
        countEdges++;
    }
    return countEdges;
}
```

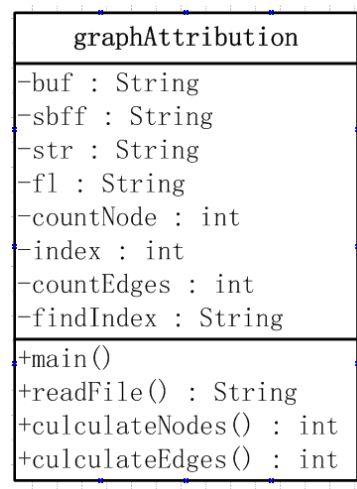
- What was implemented?

We have already implemented the File Reading, Node Counting, Edge Counting, and an easy user interface to select graph.

- Data flow diagram



- Architecture



- Description of individual parts

1. SU TZU CHENG: Node/Edge Counting Method, OOP Designing.
2. MAO YU QING: Node/Edge Counting Method, File Reading Method.
3. OH YEONSOO: Milestone Writing, User Hand Book Writing
4. Marti Nuñez Muñoz: Program Cleaning/organizing

4. Milestone

4.1 Milestone 1

- Things that we have achieved:
We already achieve
 1. Read the .graphml file into the Eclipse
 2. Easy user interface to choose the analyzing diagram
 3. Calculates the Node and Edge
 4. Slightly OOP design
- Who did what?
MAO YU QING and SU TZU CHENG are mainly in charge of the JAVA program writin.
OH YEONSOO is mainly in charge of Documentation writing.
MATI is mainly in charge of program cleaning.
- What to achieve next?
We are planning to achieve the algorism of node to node distant and the algorism of the shortest path calculation.
- Problem encountered:
 1. **Problem:** We were thinking to create an object to read the file using the BufferedReader. But we found out that we can only return one line of the file
Solution: We transform the data from the BufferedReader into StringBuffer using while to sum it up and transform into String. So that we can store the whole data into a String object.
 2. **Problem:** When we are writing the NodeCounter, trying to put the if(xx.contains(String)) in a while loop, the content search index will always point to the start of the string. And we come up with two solutions.
Solution1: Use indexOf() function to get the current

index, and point the last search index at the next search.

Solution2: Use `replace()` function to replace the search target, so the next search won't get the same thing.

4.2 Milestone 2