Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 3 REPORT

CANER KARAKAŞ
131044061

Özgü Göksu

PART 1

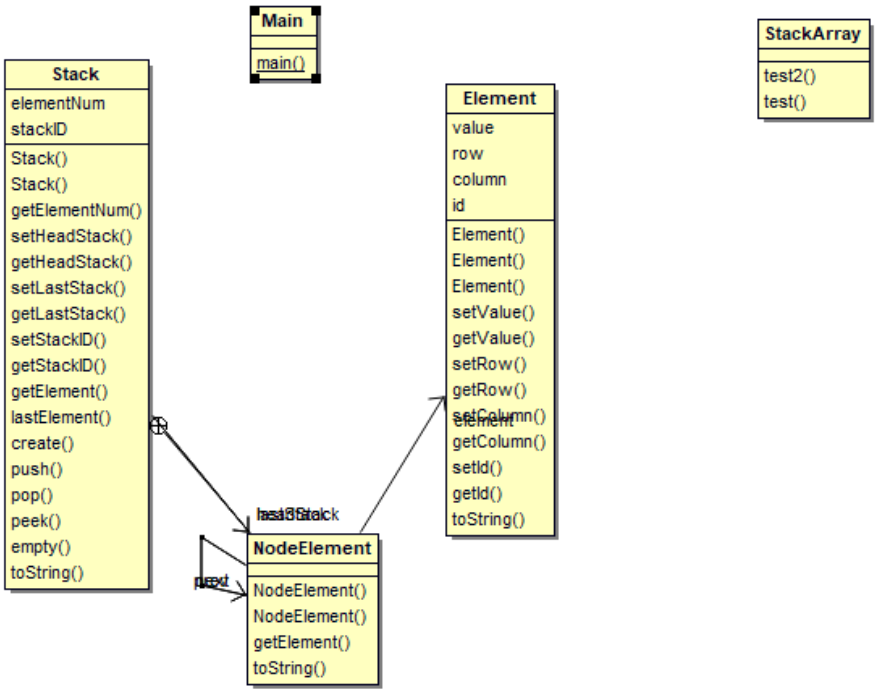
1 INTRODUCTION

1.1 Problem Definition

To keep the data from the file with the help of the stack and doing operations on it. This information is a digital image created with a binary number system. Our goal is to find the number of white components. The desired ones are those which are adjacent to each other.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

Stack Class

This class allows us to keep information as simple link list nodes. The class has inner class NodeElement. NodeElement has three type. First type is Element(class). Second is NodeElement and one more like that. They are next and prev. Stack class works as collection stack. Element adds to the top, the resulting parent is removed.

Element Class

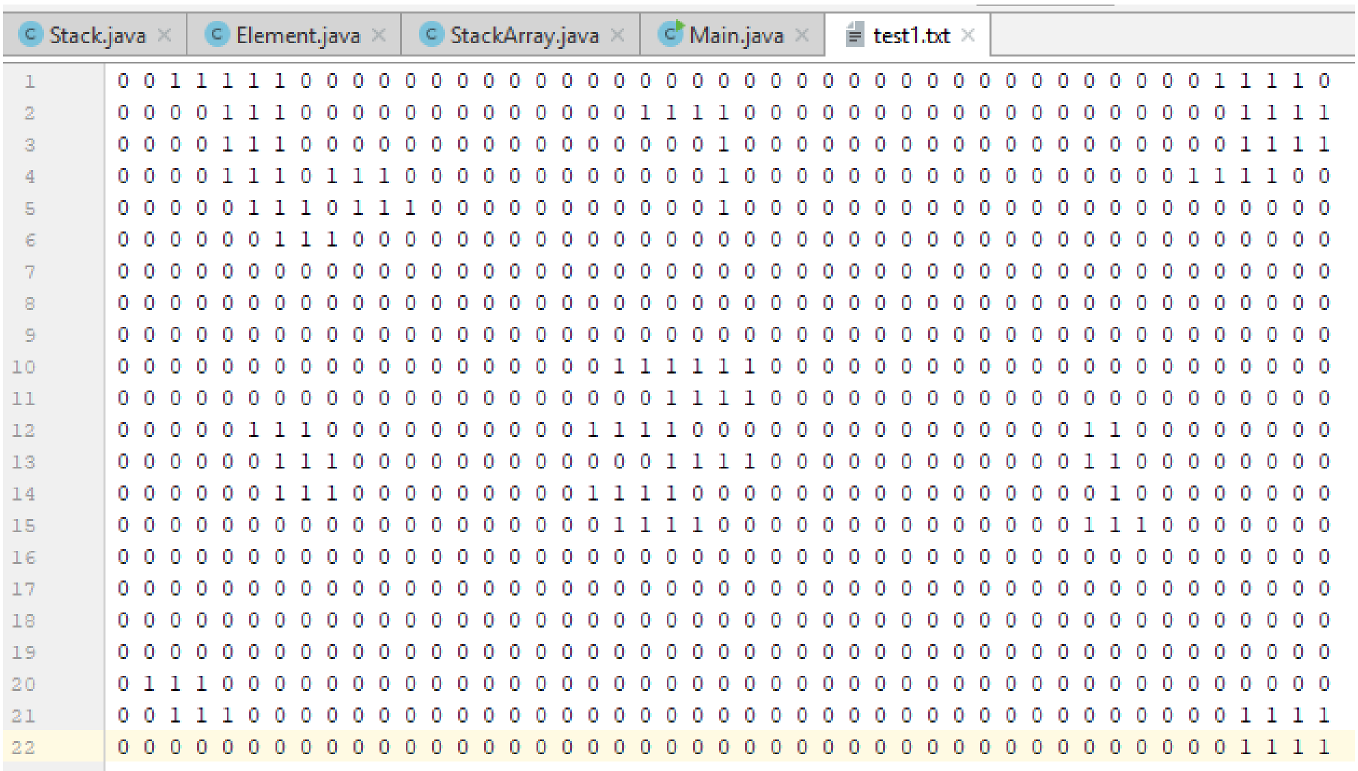
Binary numbers are kept as objects of this class. The class holds the value, ID, row and column information.

The values read from the file are assigned to he two-dimensional array. The type of the array is Element class. Solutions were made with the classes described above.

First, the array is scanned and the value of the element object is stopped by one. Neighbors are inspected. Element's value by one which added to the stack. After changed its ID. İf it does not find, it is kicked out of stack and returns. Where it is returned, inspection is done again. This time, the neighbors with unidentified ID will go and repeat the same process.

3 RESULT

3.1 Test Cases



```
0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0
3	0	1	1	1	1	0	0	1	0	1	0
4	0	1	1	0	1	0	0	0	0	1	1
5	0	0	0	1	1	0	0	0	1	1	0
6	0	0	0	0	0	0	0	0	0	0	0

```java
public void test() throws Exception {
    Element element = new Element( v: '1',  r: 1,  c: 2, i: 'a');
    Stack stack = new Stack();
    System.out.println(stack.getElementNum());
    stack.push(stack.create(element));
    System.out.println(stack.getElementNum());
    stack.push(stack.create(element));
    System.out.println(stack.getElementNum());
    stack.push(stack.create(element));
    System.out.println(stack.getElementNum());
    stack.push(stack.create(element));
    System.out.println(stack.getElementNum());
    stack.pop();
    System.out.println(stack.getElementNum());
    stack.pop();
    System.out.println(stack.getElementNum());
    stack.pop();
    System.out.println(stack.getElementNum());

    System.out.println(stack.pop());
    System.out.println(stack.peek());
}
```

3.2 Running Results

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-java
Stack Test
----------
0
1
2
3
4
3
2
1
id= a row=1 column=2
null

White Comp Test
---------------
9
00AAAAA0000000000000000000000000000000000BBBB0
0000AAA00000000000000CCCC00000000000000000000BBBB
0000AAA0000000000000000C0000000000000000000BBBB
0000AAA0DDD00000000000C00000000000000000BBBB00
00000AAA0DDD00000000000C0000000000000000000000
000000AAA000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000EEEEEE000000000000000000000
00000000000000000000000EEEE000000000000000000000
00000FFF0000000000EEEE00000000000000000GG00000000
000000FFF00000000000000EEEE00000000000000GG00000000
000000FFF000000000000EEEE0000000000000000G00000000
0000000000000000000000EEEE00000000000000000GGG0000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
0HHH0000000000000000000000000000000000000000000
00HHH00000000000000000000000000000000000000000IIII
000000000000000000000000000000000000000000000IIII
```

```
"C:\Program Files\Java\jdk
Stack Test
----------
0
1
2
3
4
3
2
1
id= a row=1 column=2
null

White Comp Test
---------------
4
00000000000
00A000B0000
0AAAA00C0D0
0AA0A0000DD
000AA000DD0
00000000000
```

TIME COMPLEXITY

Class	Method	Complexty
Element	All methods	O(1)
Stack	All methods	O(1)
Stack	toString	O(n)
StackArray	test()	O(1)
StackArray	test2()	O(n)(row*column)
Main	main()	O(n)(row*column)

PART 2

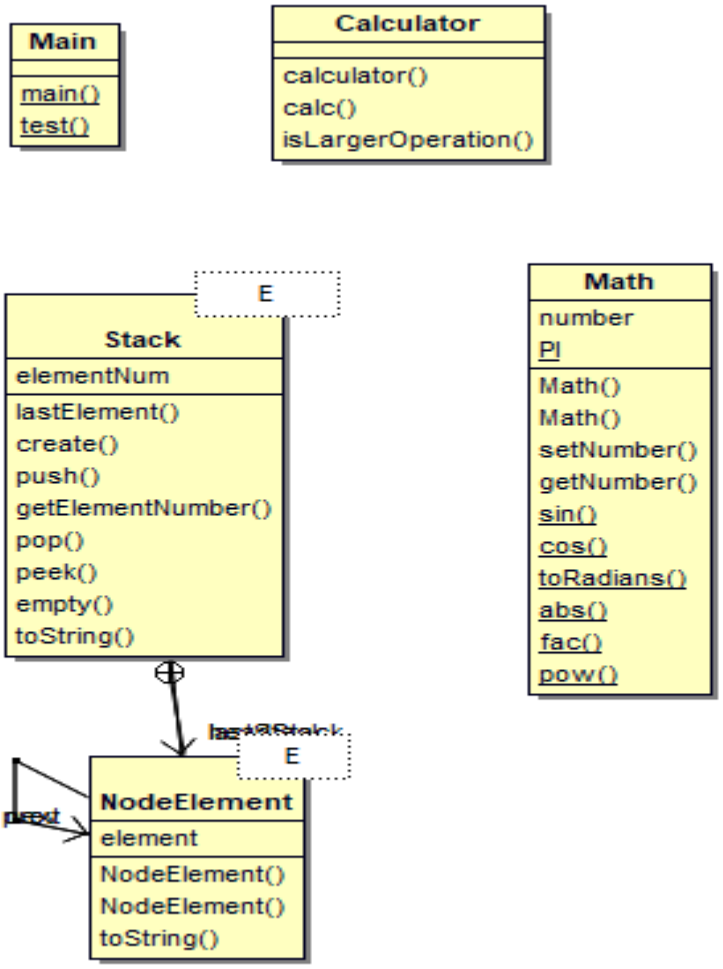
4 INTRODUCTION

4.1 Problem Definition

To convert the data in the form of infix to postfix with own stack data structure.

5 METHOD

5.1 Class Diagrams



5.2 Problem Solution Approach

Stack Class

İt is similar to the first part's Stack. However, it has a node without an element object. İn addition, this class is generic. The node class within this class is generic as well as this class.

Math Class

This class contains the methods of sin abs and cos.

All the data from the file is kept in stack array. The first line corresponds to the first element of the stack array. The last line contains the infix process. This corresponds to the last element of the stack array. A temporary stack was first used to avoid throwing up the stack. Then transferred to the stack array. The stack element, which is the last element of the stack array, has been converted to postfix with the postfix sequence and a temprorary stack. Operatör we in the temporary stack, other data were kept in the array. İt is converted to postfix by looking at the process priority and parentheses.

6 RESULT

6.1 Test Cases

Main.java | test2.txt | Calculator.java

```
y = 3
z = 16

( y + sin( y * z ) + ( z * ( -10.3 ) ) )
```

```
w=5
x=6

( w + 4 ) * ( sin( x ) - 77.9 )
```

6.2 Running Results

```
"C:\Program Files\Java\jdk-11.0.2\b
Calculator Test
------------
y=3
z=16
(y+sin(y*z)+(z*(-10.3)))

y y z * S + z 0 1 0 . 3 - * +
-161.0568551709647

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk-11.0.2\bi
Calculator Test
------------
w=5
x=6
(w+4)*(sin(x)-77.9)

w 4 + x S * 7 7 . 9 -
-76.95924383059112

Process finished with exit code 0
```

TIME COMPLEXITY

Class	Method	Complexty
Stack	All methods	O(1)
Stack	toString ()	O(n)(stack.size)
Math	Setters and getters	O(1)
Math	Sin()	O(20)==O(1)
Math	Cos()	O(20)==O(1)
Math	toRadians()	O(1)
Math	abs()	O(1)
Math	fac()	O(n)

Math	Pow()	O(n)
Calculator	calculator()	O(n)
Calculator	calc()	O(n)
Calculator	isLargerOperation()	O(1)