**Gebze Technical University**

**Computer Engineering**

**CSE 222 - 2019 Spring**

**HOMEWORK 3 REPORT**

**CANER KARAKAŞ**

**131044061**

Özgü Göksu

PART 1

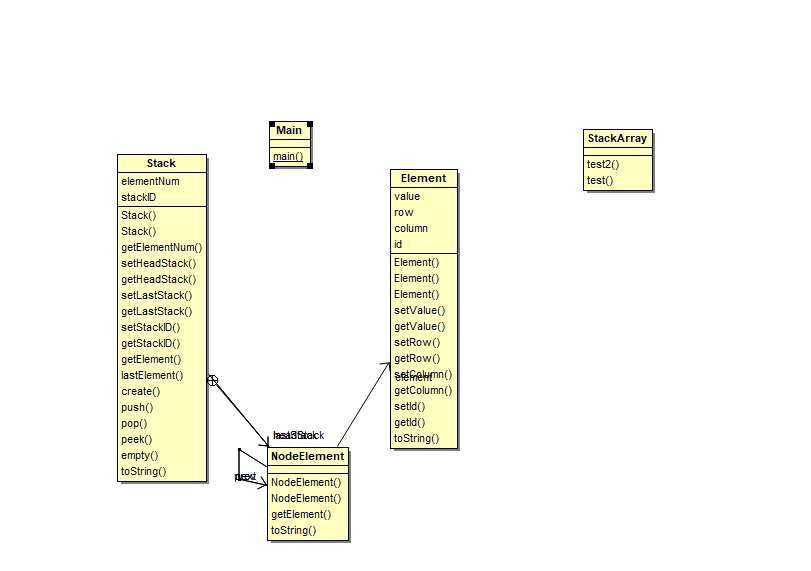
# INTRODUCTION

## 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.

# METHOD

## Class Diagrams



## 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.

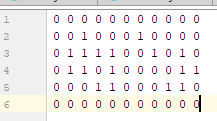
The values read from the file are assigned to he two-dimensional array. The type of the array is Element class. Solutions were made with the classes described above.

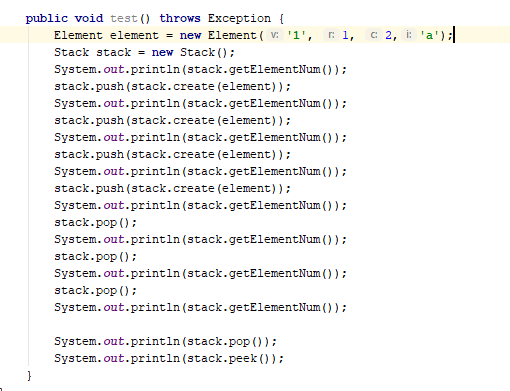
First, the array is scanned and the value of the element object is stopped by one. Neighbors are inspected. Element’s value by one which added to the stack. After changed its ID. İf it does not find, it is kicked out of stack and returns. Where it is returned, inspection is done again. This time, the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

# RESULT

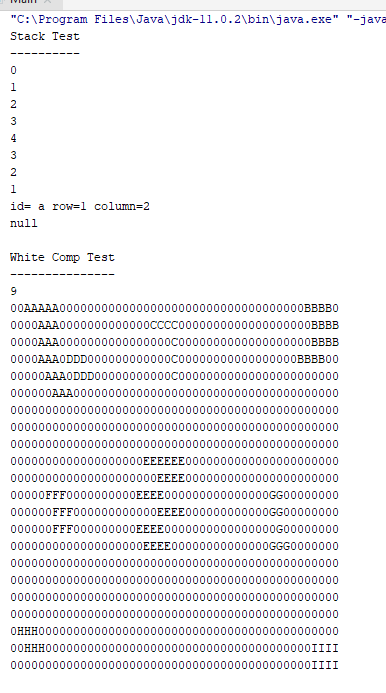
## Test Cases

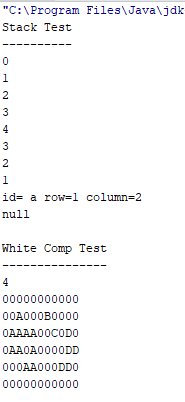






## Running Results





**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**

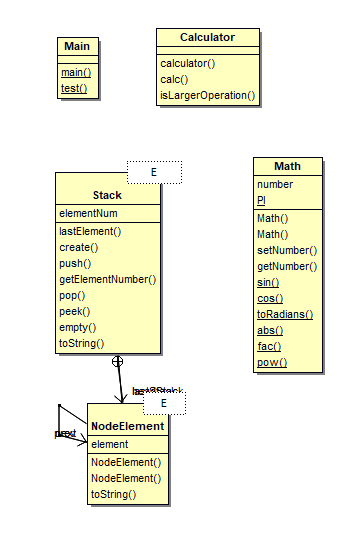
# INTRODUCTION

## Problem Definition

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

# METHOD

## Class Diagrams



## 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.

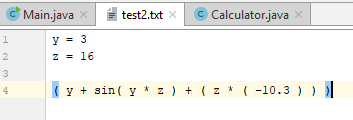
**Math Class**

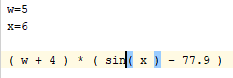
This class contains the methods of sin abs and cos.

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.

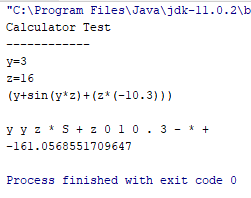
# RESULT

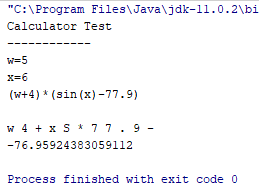
## Test Cases





## Running Results

****

****

**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) |