Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 3 REPORT

HAMZA YOĞURTCUOĞLU
171044086

Course Assistant: ÖZGÜ GÖKSU

----------PART – 1----------

1 INTRODUCTION

1.1 Problem Definition

There are 10 problems below to solve and algorithm that is thought.

1- We have to get O(n) complexity while using Stack Data Structure. That is used hard .

2- How to travel with white point in Image.

3- Will white point travel again and again. If I travel in controlled white point stack complexity that would be O(n) complexity or not.

4- If Current point that is white point whose down , up , left , right could check each directions.

5- While ckecking to up , down ,left ,right bounds that can be no side.

6- How to use stack in this problem.

7- Could I check Each black point or not ?

8- Own Stack would be with linklist or vector data structure.

9-Which method is needed in stack.

10- Each point should have a cordinate such a (x , y).

1.2 System Requirements

+ Total of Algoirthm Complexity is work with O(n) (Meaning is depend of your dara (White Point in Image)).

+ All Operating Systems handle this program.

+ Doesnt need a lot of memory acually if have alot of data , memory usage will raise linearly.

+ Pogram can work 128KB of memory (That can be change your data.)

+ This program doest make properly in smartphone it can be work in computer.

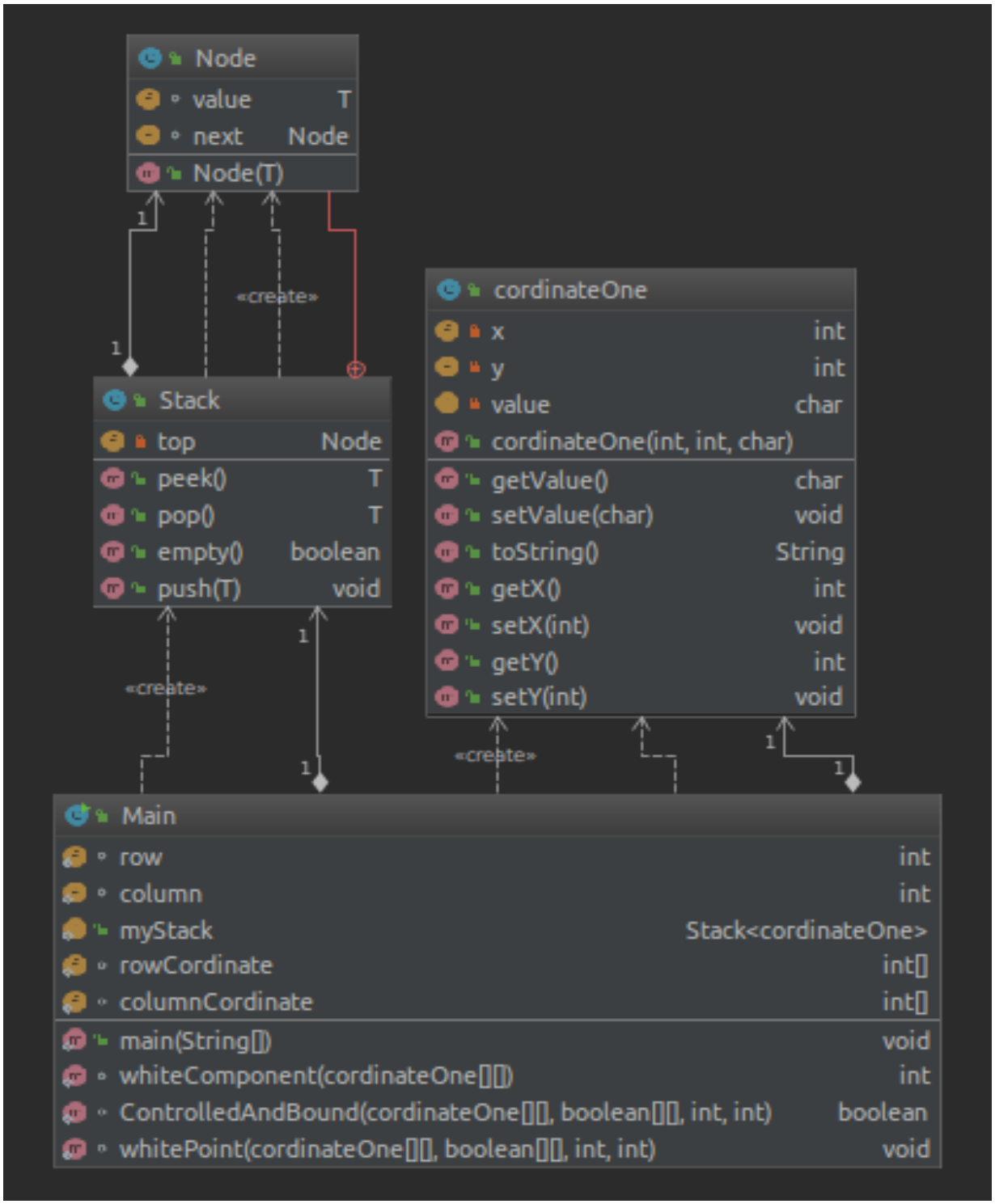
+ Doesnt need a specific of hardware just a computer work.

+ Just take executable file for directly execute program. Then , it works efficently.

2 METHOD

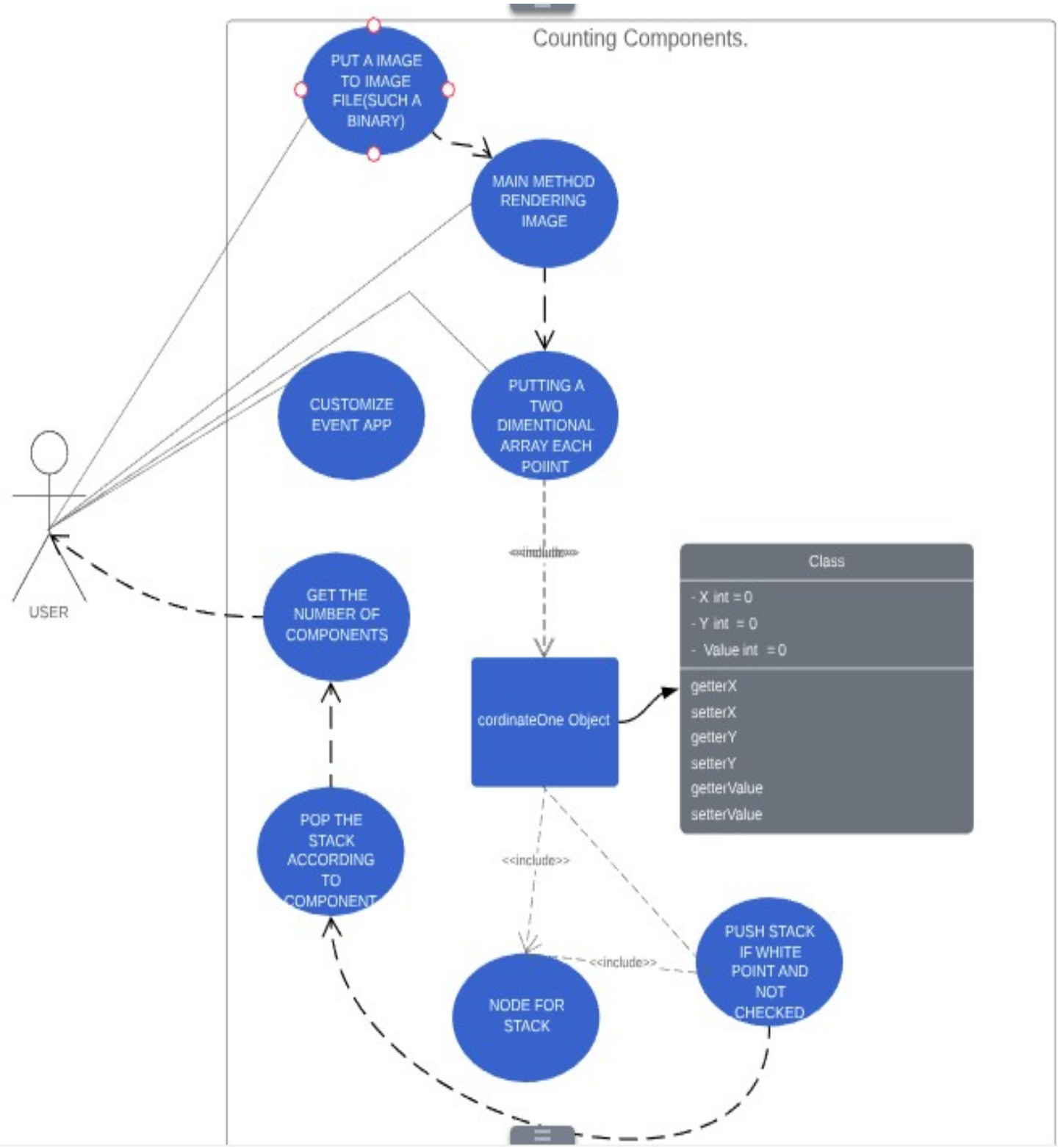
2.1 Class Diagrams

NODE : INNER CLASS IN STACK THAT CREATED A NODE
STACK : CAN BE CREATED MANY IN MAIN
CORDINATEONE : CAN BE CREATED MANY IN MAIN .
MAIN : IF CREATE A STACK , IT HAS TO CREATE A NODE , CORDINATE RELATIONSHIP.



2.2 Use Case Diagrams

All Actions are included in this diagram. User has to be put the binary image in file. Then all Actions are handled.



2.3 Problem Solution Approach

In problem solving , stack is used . Because this problem is close to recursive solution. If you have recursive solution. You can imply with **STACK.**

Solution : If you find a 1 point whose left , right , up and down which are pushing stack , then each 1's of component pop the stack. Each component's 1 is traveled and if A point is traveled point , according to its cordinate of ControlledOnePoint matrix is true (default false).In two dimentional matrix has row and column that means

row * column = Number of Image Point O(n)

MAIN METHOD

Reading the image file. Main Method that is used like driver method. Then main method call whiteComponent method.

WHITE COMPONENT

```
static int whiteComponent(cordinateOne Image[][])
{

    boolean ControlledOnePoint[][] = new boolean[row][column];
    int whiteAreaNumber = 0;
    for (int i = 0; i < row; ++i)
        for (int j = 0; j < column; ++j)
            if (Image[i][j].getValue()=='1' && !ControlledOnePoint[i][j])
            {
                myStack = new Stack<cordinateOne>();
                whitePoint(Image, ControlledOnePoint, i, j);
                ++whiteAreaNumber;
            }

    return whiteAreaNumber;
}
```

This method take a Image Array that has binary number. Each point will be cheched above two for loop that means is i x j = number of point (n). If any point is 1 and not controlled whitePoint function is called. Complexity O(n)

Controlled and Bound

```
static boolean ControlledAndBound(cordinateOne Image[][],
                                  boolean ControlledOnePoint[][], int tempRow, int tempColumn)
{
    boolean suitOrNot =  (tempRow >= 0) && (tempRow < row) &&
            (tempColumn >= 0) && (tempColumn < column) && !ControlledOnePoint[tempRow][tempColumn] &&
            (Image[tempRow][tempColumn].getValue()=='1');

    return  suitOrNot;
}
```

This function checks cordinate of a point that is 1 , checked or not , being in border of matrix or not. Complexity --> O(6)

whitePoint

```
static void whitePoint(cordinateOne Image[][], boolean ControlledOnePoint[][], int currentRow, int currentCol)
{
    for (int k = 0; k < 4; ++k) {
        if (ControlledAndBound(Image, ControlledOnePoint,  tempRow: currentRow + rowCordinate[k],  tempColumn: currentCol + columnCordinate[k])
                &&(!ControlledOnePoint[currentRow + rowCordinate[k]][currentCol + columnCordinate[k]]) ){
            myStack.push(Image[currentRow + rowCordinate[k]][currentCol + columnCordinate[k]]);
        }

        if (k+1 == 4){
            ControlledOnePoint[currentRow][currentCol] = true;
            if (!myStack.empty()){
                cordinateOne temp = myStack.pop();
                currentRow = temp.getX();
                currentCol = temp.getY();
                k = -1;
            }
        }
    }
}
```

Im using STACK data structure in this part. İf a current point is 1 . In this function is checking left , right ,up , down side which are 1 that is pushing STACK. This event is keeping then pop in the stack a point (is 1) all 1 is a one component.

Stack Usage Complexity :

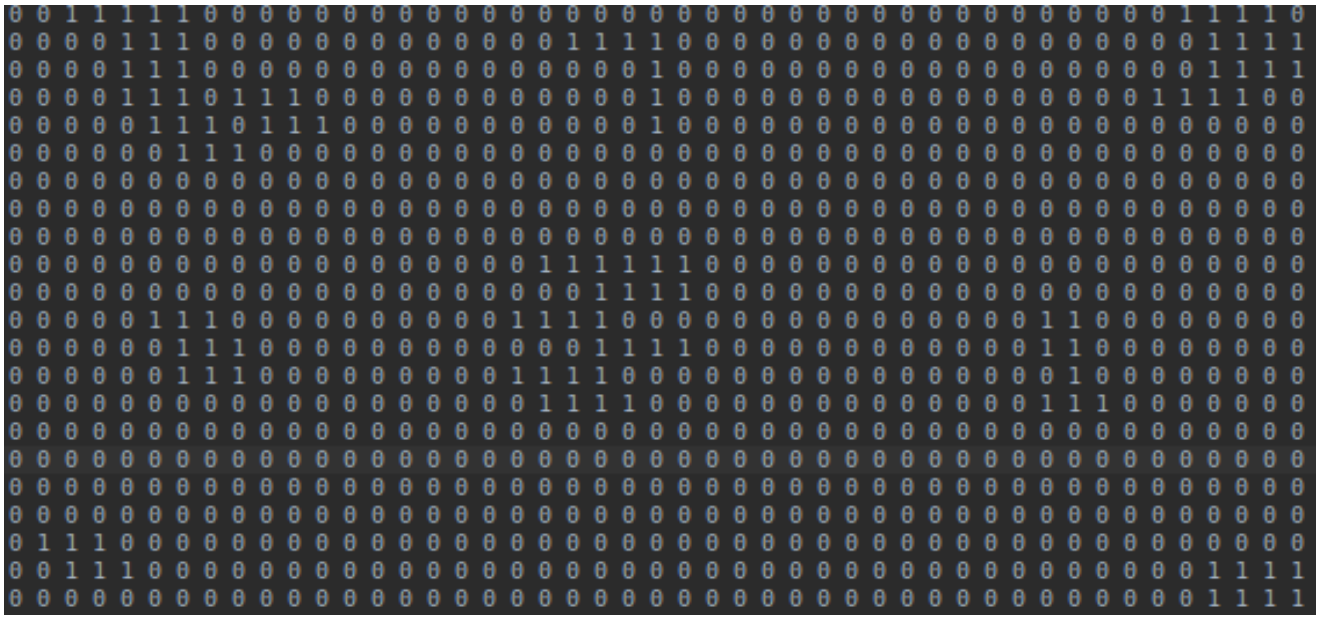
IF all point of image are 1 complexity O(n) otherwise less than O(n).

STACK

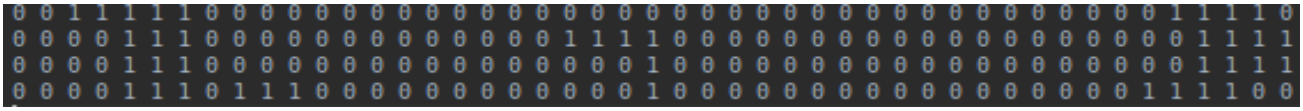
Stack has 4 method empty , push , pop , peek . I created like a linkist just keeping top point of stack then top each point has a next point. In All method is checked top node of stack. All Method Complexity is **O(1)**

3 RESULT

3.1 Test Cases



Above Binary Image. Test 1



Above Binary Image. Test 2

3.2 Running Results

Test 1

```
Number of White Components : 9
```

Test 2

```
Number of White Components : 4
```

----------PART – 2----------

4 INTRODUCTION

4.1 Problem Definition

There are 9 problems below to solve and algorithm that is thought.

1- All equation (infix expression) how to convert to postfix expression

2- Implementing my own cos , sin , abs methods.

3- How to difference betwen precedence parenthesis and trigonometry method parenthesis.

4- Evaluating postfix expression' value.

5- Where Stack is used with evaluating postfix expression' value.

6- When evaluating trigonometry methods , postfix or infix.

7- For String equation , doing a string compute methods.

8- Where variable are keeping (Data Structure)

9- What variables should have attributes.

4.2 System Requirements

+ Total of Algoirthm Complexity is work with O(n) because we are traveling one time each keword (')' , variable , '+' etc.)

+ All Operating Systems handle this program.

+ Doesnt need a lot of memory acually if have alot of data , memory usage will raise linearly.

+ Pogram can work 128KB of memory (That can be change your data.)

+ This program doest make properly in smartphone it can be work in computer.

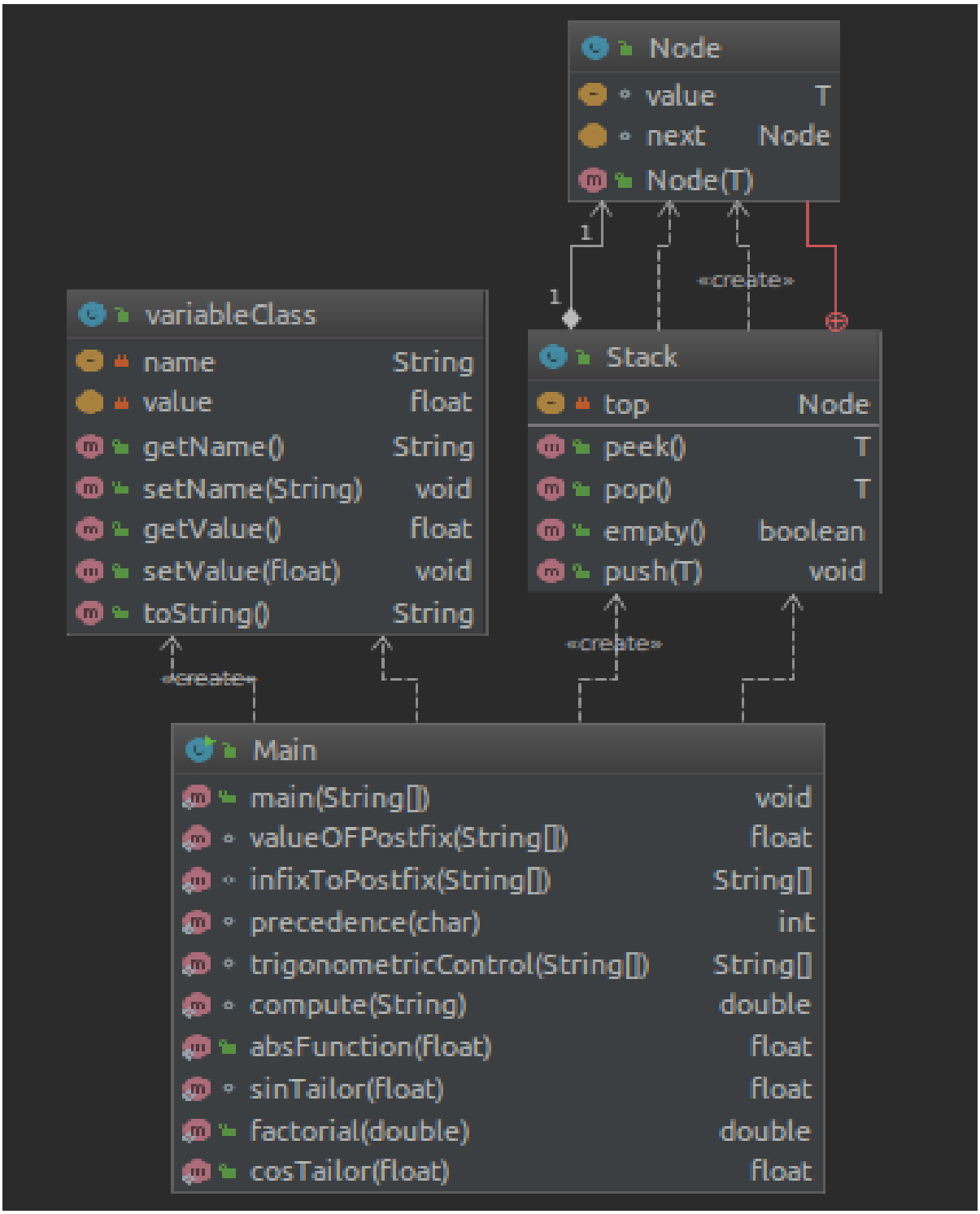
+ Doesnt need a specific of hardware just a computer work.

+ Just take executable file for directly execute program. Then , it works efficently.

5 METHOD

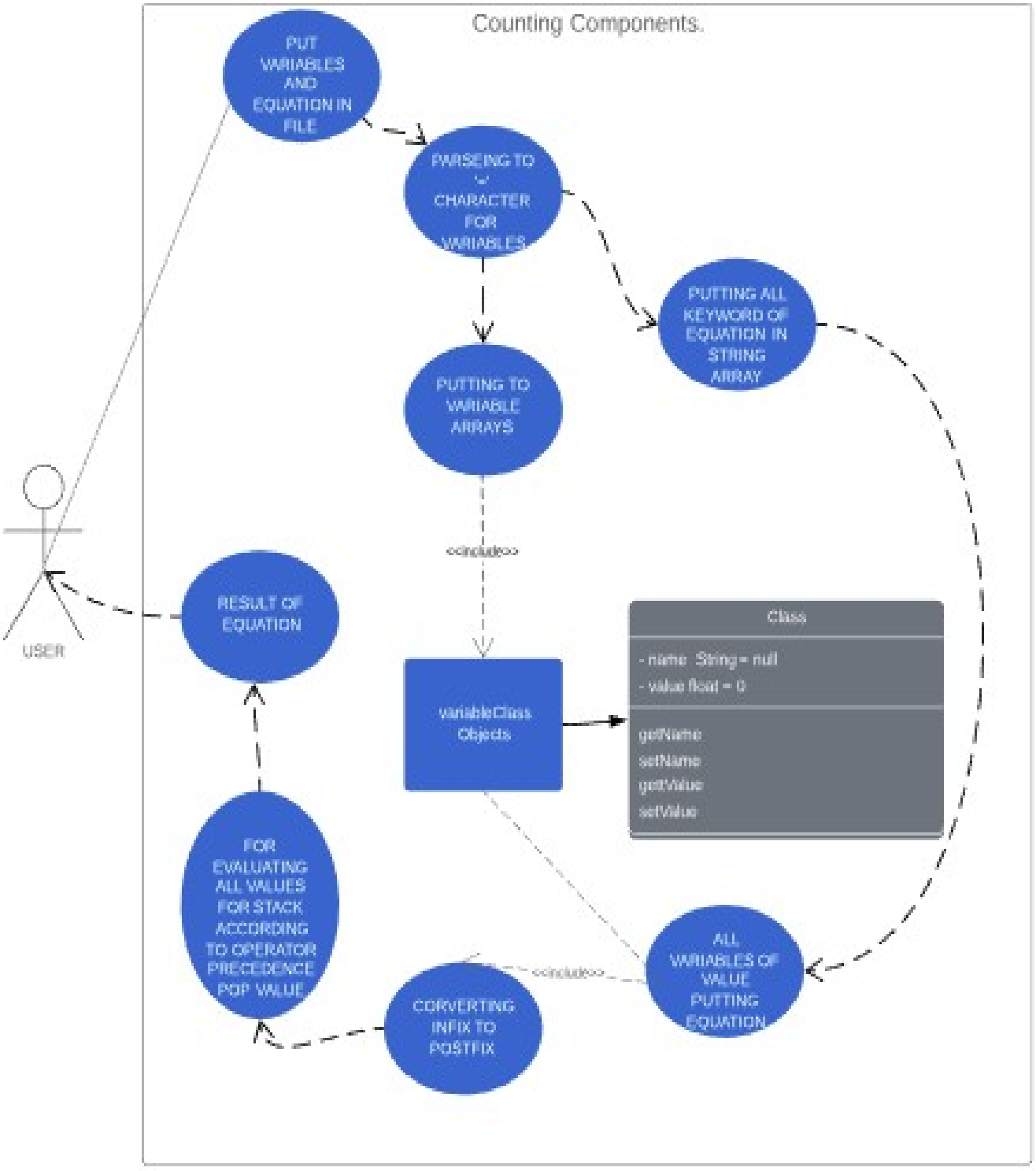
5.1 Class Diagrams

NODE : INNER CLASS IN STACK THAT CREATED A NODE
STACK : CAN BE CREATED MANY IN MAIN
VARIABLECLASS : CAN BE CREATED MANY IN MAIN WE ARE CHECKING
ACCORDIN TO ' = ' CHARACTER.
IT KEEPS VARIABLE NAME AND VALUE.
MAIN : MAIN METHOD WHERE READING FILE THEN PUTTING VARIABLE ARRAY AND KEYWORD ARRAY



5.2 Use Case Diagrams

All Actions are included in this diagram. User has to be put the “VARIABLES ”and “EQUATION” properly in file. Then all Actions are handled.



5.3 Problem Solution Approach

+ Firstly, StringTokenizer is helping to parse variables according to ‘ = ‘ character . Left side is variable of name , rightside is variable of value . Then variableClass array is created that keeping all variable.

+ Then if any equation is there (file) . It is separated according to ‘(‘ , ‘)’ , ‘+’ , ‘ - ‘ , ‘*’ , ‘ / ‘ Then , all keyword is putting String array. (Ex Keyword : “y” , “-15.8” , “cos” etc.)

+ All variable value are changed instead of variable name.

+ All trigonometrix function is calculated. Result is putting instead of trigonometric string.

+ Equation is coverted according to precedence of operator. Operators is pushed a STACK precedence is bigger then stack putting the new equation. Actuall 3 state for operator.

1) If any operand add to postfix equation

2)If operator bigger precedence in stack . Bigger operator is put in postfix equation.

3)If operator equal precedence in stack . In of stack operator is put in postfix equation.

+ Evaluating of postfix value has two step . STACK is used for operand

1) If any operand push the stack.

2) If any operator 2 pop from the stack , so you get 2 operand that is evaluated then push again in to STACK.

MAIN METHOD

Reading the file for Equation and Variable :

Firstly, StringTokenizer is helping to parse variables according to ‘ = ‘ character . Left side is variable of name , rightside is variable of value . Then variableClass array is created that keeping all variable.

Then if any equation is there (file) . It is separated according to ‘(‘ , ‘)’ , ‘+’ , ‘ - ‘ , ‘*’ , ‘ / ‘ Then , all keyword is putting String array. (Ex Keyword : “y” , “-15.8” , “cos” etc.)

TRIGONOMETRICCONTROL METHOD O(N)

```
for (int l = 0; l<infixStringArray.length;l++){
```

Just a one loop (for)O(n) It is kept until keyword of lenght

if there is any trigonometriccontrol. Inside of trigonometric method is taken a String and calculated then all evaluated equation that is sent to my own cos , sin or ,abs methods.

Then result putting in the array again.

SinTailor - cosTailor - abs methods are implemented .

INFIXTOPOSTFIX METHOD O(N)

All converted process continues accoding to keyword of size that means is O(n).

Equation is coverted according to precedence of operator. Operators is pushed a STACK precedence is bigger then stack putting the new equation. Actuall 3 state for operator.

1) If any operand add to postfix equation

2)If operator bigger precedence in stack . Bigger operator is put in postfix equation.

3)If operator equal precedence in stack . In of stack operator is put in postfix equation.

VALUEOFPOSTFIX METHOD O(N)

I calculate 11 step I got the help in book.

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

6 RESULT

6.1 Test Cases

Test 1

```
w=5
x=6

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

Test 2

```
y=3
z=16

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

Test 3

```
y=3
z=16

( y - z )
```

6.2 Running Results

Test 1

```
Result : -692.14923
```

Test 2

```
Result : 168.54286
```

Test 3

```
Result : -13.0
```

- Main titles -> 16pt , 2 line break
- Subtitles -> 14pt, 1.5 line break
- Paragraph -> 12pt, 1.5 line break