U18CO018
Shubham Shekhaliya
Software tools - 4
Lab Assignment-4
Topic: GUI with Java

Write a java event handling program to create a scientific calculator.

NOTES :

- Calculator can take input from key board as well.
- Add validations
  - e.g. for fractionl number only one dot is allowed.
- Screenshot given below is for reference GUI .

Calculator.java

```
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Stack;

import static java.lang.Math.*;
import static java.lang.Math.pow;

public class Calculator {
    private JFrame f;

    private JButton AsinButton, AcosButton, AtanButton;
    private JButton PlusButton, MinusButton, MulButton, DivButton, Plus_Minus;
    private JButton X2Button, X3Button, XYButton, XInvButton;
    private JButton OpenBracket, CloseBracket;
    private JButton Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Nine
, PiButton;
    private JButton SinButton, CosButton, TanButton;
    private JButton nPrButton, nCrButton, NFactButton;
    private JButton TwoRoot, ThreeRoot, YRoot;
    private JButton DELButton, CLRButton, EXITButton, ANSButton;
    private JButton LogButton, Log10Button;
    private JButton TanhButton, CoshButton, SinhButton;
    private JButton ExpButton;
    private JButton Dot;
    private JTextField text;
    private javax.swing.JPanel JPanel;
    private JTextField AnsText;

    private void addButton(JButton btn,int x1, int y1, int x2, int y2, String
label) {
        btn = new JButton(label);
        btn.setBounds(x1,y1,x2,y2);
        f.add(btn);
    }

    private void addText(JTextField field, int x1, int y1, int x2, int y2) {
        field = new JTextField();
        field.setBounds(x1,y1,x2,y2);
        f.add(field);
    }

    public Calculator() {
        f = new JFrame("Scientific Calculator");

        // 1st Row
        DELButton = new JButton("DEL");
```

```
        DELButton.setBounds(10,105,80,40);
        f.add(DELButton);

        CLRButton = new JButton("CLR");
        CLRButton.setBounds(110,105,80,40);
        f.add(CLRButton);

        ANSButton = new JButton("ANS");
        ANSButton.setBounds(710,105,80,40);
        f.add(ANSButton);

        EXITButton = new JButton("EXIT");
        EXITButton.setBounds(810,105,80,40);
        f.add(EXITButton);

        // 2nd Row
        Seven = new JButton("7");
        Seven.setBounds(10,155,80,40);
        f.add(Seven);

        Eight = new JButton("8");
        Eight.setBounds(110,155,80,40);
        f.add(Eight);

        Nine = new JButton("9");
        Nine.setBounds(210,155,80,40);
        f.add(Nine);

        PlusButton = new JButton("+");
        PlusButton.setBounds(310,155,80,40);
        f.add(PlusButton);

        X2Button = new JButton("X^2");
        X2Button.setBounds(410,155,80,40);
        f.add(X2Button);

        X3Button = new JButton("X^3");
        X3Button.setBounds(510,155,80,40);
        f.add(X3Button);

        XYButton = new JButton("X^Y");
        XYButton.setBounds(610,155,80,40);
        f.add(XYButton);

        XInvButton = new JButton("1/X");
        XInvButton.setBounds(710,155,80,40);
        f.add(XInvButton);
```

```
        OpenBracket = new JButton("(");
        OpenBracket.setBounds(810,155,80,40);
        f.add(OpenBracket);

        // 3rd Row
        Four = new JButton("4");
        Four.setBounds(10,205,80,40);
        f.add(Four);

        Five = new JButton("5");
        Five.setBounds(110,205,80,40);
        f.add(Five);

        Six = new JButton("6");
        Six.setBounds(210,205,80,40);
        f.add(Six);

        MinusButton = new JButton("-");
        MinusButton.setBounds(310,205,80,40);
        f.add(MinusButton);

        TwoRoot = new JButton("2√");
        TwoRoot.setBounds(410,205,80,40);
        f.add(TwoRoot);

        ThreeRoot = new JButton("3√");
        ThreeRoot.setBounds(510,205,80,40);
        f.add(ThreeRoot);

        YRoot = new JButton("y√");
        YRoot.setBounds(610,205,80,40);
        f.add(YRoot);

        NFactButton = new JButton("N!");
        NFactButton.setBounds(710,205,80,40);
        f.add(NFactButton);

        CloseBracket = new JButton(")");
        CloseBracket.setBounds(810,205,80,40);
        f.add(CloseBracket);

        // 4th Row
        One = new JButton("1");
        One.setBounds(10,255,80,40);
        f.add(One);

        Two = new JButton("2");
        Two.setBounds(110,255,80,40);
```

```
        f.add(Two);

        Three = new JButton("3");
        Three.setBounds(210,255,80,40);
        f.add(Three);

        MulButton = new JButton("*");
        MulButton.setBounds(310,255,80,40);
        f.add(MulButton);

        SinButton = new JButton("sin");
        SinButton.setBounds(410,255,80,40);
        f.add(SinButton);

        CosButton = new JButton("cos");
        CosButton.setBounds(510,255,80,40);
        f.add(CosButton);

        TanButton = new JButton("tan");
        TanButton.setBounds(610,255,80,40);
        f.add(TanButton);

        ExpButton = new JButton("exp");
        ExpButton.setBounds(710,255,80,40);
        f.add(ExpButton);

        nPrButton = new JButton("nPr");
        nPrButton.setBounds(810,255,80,40);
        f.add(nPrButton);

        // 5th Row
        Zero = new JButton("0");
        Zero.setBounds(10,305,80,40);
        f.add(Zero);

        Dot = new JButton(".");
        Dot.setBounds(110,305,80,40);
        f.add(Dot);

        Plus_Minus = new JButton("±");
        Plus_Minus.setBounds(210,305,80,40);
        f.add(Plus_Minus);

        DivButton = new JButton("/");
        DivButton.setBounds(310,305,80,40);
        f.add(DivButton);

        AsinButton = new JButton("asin");
```

```
        AsinButton.setBounds(410,305,80,40);
        f.add(AsinButton);

        AcosButton = new JButton("acos");
        AcosButton.setBounds(510,305,80,40);
        f.add(AcosButton);

        AtanButton = new JButton("atan");
        AtanButton.setBounds(610,305,80,40);
        f.add(AtanButton);

        LogButton = new JButton("log");
        LogButton.setBounds(710,305,80,40);
        f.add(LogButton);

        nCrButton = new JButton("nCr");
        nCrButton.setBounds(810,305,80,40);
        f.add(nCrButton);


        // 6th Row
        SinhButton = new JButton("sinh");
        SinhButton.setBounds(410,355,80,40);
        f.add(SinhButton);

        CoshButton = new JButton("cosh");
        CoshButton.setBounds(510,355,80,40);
        f.add(CoshButton);

        TanhButton = new JButton("tanh");
        TanhButton.setBounds(610,355,80,40);
        f.add(TanhButton);

        Log10Button = new JButton("log10");
        Log10Button.setBounds(710,355,80,40);
        f.add(Log10Button);

        PiButton = new JButton("π");
        PiButton.setBounds(810,355,80,40);
        f.add(PiButton);

        text = new JTextField();
        text.setBounds(100,10,700,30);
        f.add(text);

        AnsText = new JTextField();
        AnsText.setBounds(200,60,600,30);
        f.add(AnsText);
```

```
        JPanel = new JPanel();
        f.add(JPanel);

        AnsText.setEditable(false);
        AnsText.setText("0.0");

        //Todo: digits and π
        One.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "1"); }
        });
        Two.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "2"); }
        });
        Three.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "3"); }
        });
        Four.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "4"); }
        });
        Five.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "5"); }
        });
        Six.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "6"); }
        });
        Seven.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "7"); }
        });
        Eight.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "8"); }
        });
        Nine.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "9"); }
        });
        Zero.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "0"); }
        });
        PiButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "π"); }
        });
        // ------------------------------------------------------------

        //Todo: Trigonometry Functions
        SinButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "sin"); }
        });
        CosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "cos"); }
        });
        TanButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "tan"); }
        });
        SinhButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "sinh"); }
        });
        CoshButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "cosh"); }
        });
        TanhButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "tanh"); }
        });
        AsinButton.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "asin"); }
        });
        AcosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "acos"); }
        });
        AtanButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "atan"); }
        });
        // ------------------------------------------------------------


        //Todo: Root, Power, Inverse
        X2Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"^2"); }
        });
        X3Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"^3"); }
        });
        XYButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"^"); }
        });
        XInvButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                double t = Double.parseDouble(AnsText.getText());
                AnsText.setText(String.valueOf(1/t));
            }
        });
        TwoRoot.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"2√"); }
        });
        ThreeRoot.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"3√"); }
```

```
        });
        YRoot.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"√"); }
        });
        ExpButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text()+"exp"); }
        });
        // ---------------------------------------------------------------

        //Todo: Log and Log10
        LogButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "log"); }
        });
        Log10Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "log10"); }
        });
        // ---------------------------------------------------------------


        //Todo: Dot and bracket
        OpenBracket.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "("); }
        });
        CloseBracket.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + ")"); }
        });
        Dot.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "."); }
        });
        // ---------------------------------------------------------------

        //Todo: nPr,nCr,n!
        nPrButton.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "P"); }
        });
        nCrButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "C"); }
        });
        NFactButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "!"); }
        });
        // ------------------------------------------------------------

        //Todo: Operators
        PlusButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "+"); }
        });
        MinusButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "-"); }
        });
        MulButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "*"); }
        });
        DivButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) { text.setText(text.get
Text() + "/"); }
        });
        Plus_Minus.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
//                text.setText(text.getText() + "±");
                AnsText.setText(String.valueOf(Double.parseDouble(AnsText.getT
ext())*(-1)));
            }
        });
        // ------------------------------------------------------------

        //Todo: Other Buttons
        DELButton.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                if(text.getText().length()>0) {
                    text.setText(text.getText().substring(0, text.getText().le
ngth() - 1));
                }
            }
        });
        CLRButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("");
            }
        });
        EXITButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });



        ANSButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if(CheckEquation(text.getText())) {
                    try {
                        double Answer = EquationSolver("(" + text.getText() +
")");
                        AnsText.setText(Double.toString(Answer));
                    } catch (Exception exception) {
                        JOptionPane.showMessageDialog(JPanel,"Math proccesing
error occurred");
                    }
                } else {
                    JOptionPane.showMessageDialog(JPanel, "Invalid Input");
                }
            }
        });
        f.getRootPane().setDefaultButton(ANSButton);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(910,450);
        f.setLayout(null);
        f.setVisible(true);
    }

    boolean checkSpace(String eq) {
```

```
        for(int i = 0;i<eq.length();i++) {
            if(eq.charAt(i) == ' ') {
                return  false;
            }
        }
        return true;
    }

    boolean checkParentheses(String eq) {

        Stack<Character> st = new Stack<>();
        for(int i = 0;i<eq.length();i++) {
            if (eq.charAt(i) == '(') {
                st.add('(');
            } else if (eq.charAt(i) == ')') {
                if(st.size() == 0) {
                    return false;
                } else {
                    st.pop();
                }
            }
        }

        return st.size()==0;
    }

    boolean checkDigit(char c) {
        return  (c >= '0' && c<='9') || c=='.';
    }

    boolean checkOperator(char c) {
        return  c=='+' || c=='-
' || c=='*' || c=='/' || c=='^' || c=='!' || c=='√' || c=='C' || c=='P';
    }

    boolean checkFunction(String s) {
        HashSet<String> set = new HashSet<>();
        set.add("sin");set.add("cos");set.add("tan");set.add("exp");
        set.add("asin");set.add("acos");set.add("atan");set.add("log");
        set.add("sinh");set.add("cosh");set.add("tanh");set.add("log10");
        return set.contains(s);
    }

    boolean checkCharacter(char c) {
        return c >= 'a' && c <= 'z';
    }

    boolean CheckEquation(String eq) {
```

```
        // Your code goes here
        // ohk my code goes here
        if(!checkSpace(eq) || !checkParentheses(eq)) return false;

        return true;
    }



    double EquationSolver(String eq) throws Exception {
        double res=0.0;
        // Your code goes here
        // ohk code goes here
        ArrayList<Sci_Calculator.Node> ans = infixToPostfix(eq);
        System.out.println(ans);
        res = solvePostfix(ans);
        return res;
    }

    double solveFunction(String fun, double parameter) {

        switch (fun) {
            case "sin":
                System.out.println("sine function is called");
                return sin(parameter);
            case "cos":
                return cos(parameter);
            case "tan":
                return tan(parameter);
            case "exp":
                return exp(parameter);
            case "asin":
                return asin(parameter);
            case "acos":
                return acos(parameter);
            case "atan":
                return atan(parameter);
            case "log":
                return log(parameter);
            case "sinh":
                return sinh(parameter);
            case "cosh":
                return cosh(parameter);
            case "tanh":
                return tanh(parameter);
            case "log10":
                return log10(parameter);
        }
        return 0.0;
```

```
    }

    int Prec(char ch)
    {
        switch (ch)
        {
            case '+':
            case '-':
                return 1;

            case '*':
            case '/':
                return 2;

            case '^':
            case '√':
                return 3;

            case 'P':
            case 'C':
            case '!':
                return 4;
        }
        return -1;
    }

    ArrayList<Sci_Calculator.Node> infixToPostfix(String exp) throws Exception
{
        String result = "";

        Stack<Character> stack = new Stack<>();
        ArrayList<Sci_Calculator.Node> list = new ArrayList<>();

        for (int i = 0; i<exp.length(); ++i) {
            char c = exp.charAt(i);

            if (checkDigit(c)) {
                StringBuilder sb = new StringBuilder("");
                while(checkDigit(exp.charAt(i))){
                    sb.append(exp.charAt(i));
                    i++;
                }
                double number = Double.parseDouble(String.valueOf(sb));
                i--;

                list.add(new Sci_Calculator.Node("@",number));
                result += String.valueOf(sb);
            }
```

```
            else if (c == '(')
                stack.push(c);
            else if (c == ')') {
                while (!stack.isEmpty() && stack.peek() != '(') {
                    list.add(new Sci_Calculator.Node(String.valueOf(stack.peek
()),0.0));
                    result += stack.pop();
                }
                stack.pop();
            }
            else if (checkOperator(c)) {
                while (!stack.isEmpty() && Prec(c) <= Prec(stack.peek())){
                    list.add(new Sci_Calculator.Node(String.valueOf(stack.peek
()),0.0));
                    result += stack.pop();
                }
                stack.push(c);
            } else if (c=='π') {
                list.add(new Sci_Calculator.Node("@",PI));
                result += 'π';
            } else {
                // means it is function
                StringBuilder fun = new StringBuilder();
                System.out.println("At the start of fun "+ exp.charAt(i));
                while (checkCharacter(exp.charAt(i))) {
                    fun.append(exp.charAt(i));
                    i++;
                }
                if(exp.charAt(i) == '1' && exp.charAt(i+1) =='0') {
                    fun.append("10");
                    i+=2;
                }
                if(exp.charAt(i) != '(') {
                    return null;
                }


                StringBuilder nestedExp = new StringBuilder();
                Stack<Character> temp = new Stack<>();
                temp.add('(');
                nestedExp.append('(');
                while (temp.size() != 0) {
                    i++;
                    if(exp.charAt(i) == '(') {
                        temp.add('(');
                    } else if (exp.charAt(i) == ')') {
                        temp.pop();
                    }
                    nestedExp.append(exp.charAt(i));
```

```
                }

                System.out.println(String.valueOf(nestedExp));
                ArrayList<Sci_Calculator.Node> nested_sol = infixToPostfix(Str
ing.valueOf(nestedExp));
                System.out.println(nested_sol);

                double ans = solvePostfix(nested_sol);
                System.out.println("Solve postfix ans " + ans + " and function
 " + fun);

                ans = solveFunction(String.valueOf(fun),ans);
                System.out.println("And the answer of the function is " + ans)
;
                list.add(new Sci_Calculator.Node("@",ans));

                result+= nested_sol;
            }
        }

        System.out.println(result);

        while (!stack.isEmpty()){
            if(stack.peek() == '(')
                return null;
            result += stack.pop();
        }
        return list;
    }

    long factorial(long p) {
        long ans = 1;
        for(int i = 1;i<=p;i++) {
            ans *= i;
        }
        return ans;
    }

    double solveOperator(double operand1, double operand2, String op) throws E
xception {
        switch (op) {
            case "+":
                return operand1 + operand2;
            case "-":
                return operand1 - operand2;
            case "*":
                return operand1 * operand2;
            case "/":
```

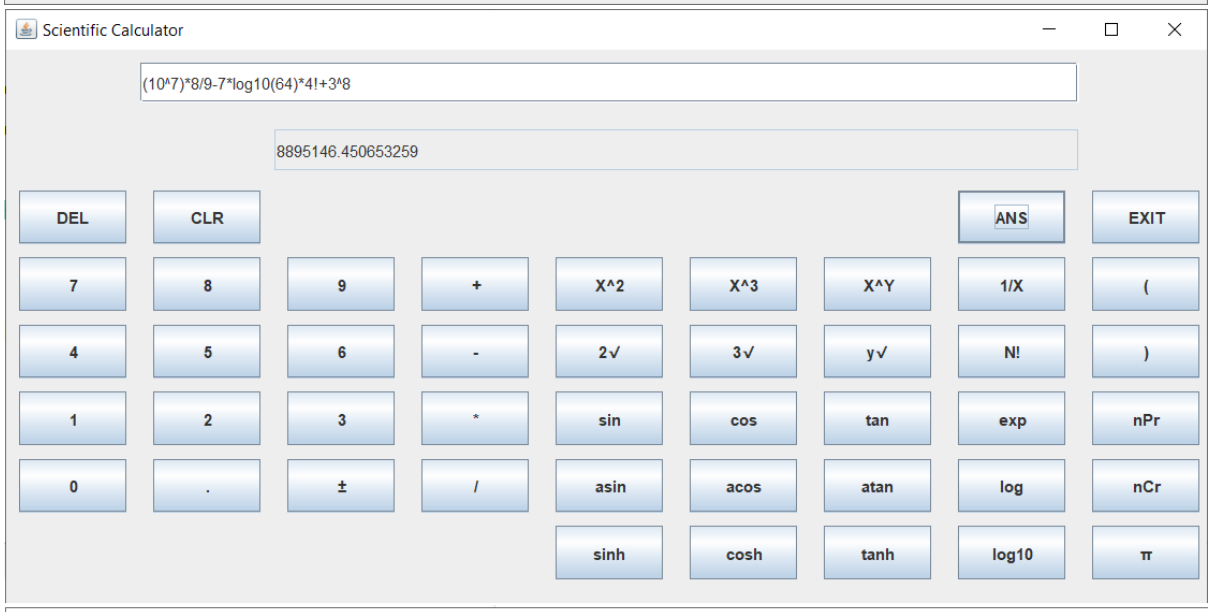
```
                return operand1 / operand2;
            case "^":
                return pow(operand1,operand2);
            case "√":
                return Math.pow(operand2,(1/operand1));
            case "P":
                return permutation(operand1, operand2);
            case "C":
                return combination(operand1, operand2);
        }
        return -1;
    }

    double permutation(double operand1,double operand2) throws Exception{

        if(operand1 != (long)operand1 || operand2 != (long)operand2) {
            throw new Exception();
        }

        long op1 = (long)operand1;
        long op2 = (long)operand2;

        long de = factorial(op1);
        long nu = factorial(op2);

        return (double) de/nu;
    }

    double combination(double operand1, double operand2) throws Exception {
        if(operand1 != (long)operand1 || operand2 != (long)operand2) {
            throw new Exception();
        }

        long op1 = (long)operand1;
        long op2 = (long)operand2;

        double de = permutation(operand1,operand2);
        long nu = factorial(op1 - op2);

        return de/nu;
    }

    double solvePostfix(ArrayList<Sci_Calculator.Node> list) throws Exception
{
        Stack<Sci_Calculator.Node> stack = new Stack<>();

        for(int i = 0;i<list.size();i++ ) {
            System.out.print("stack ");
```

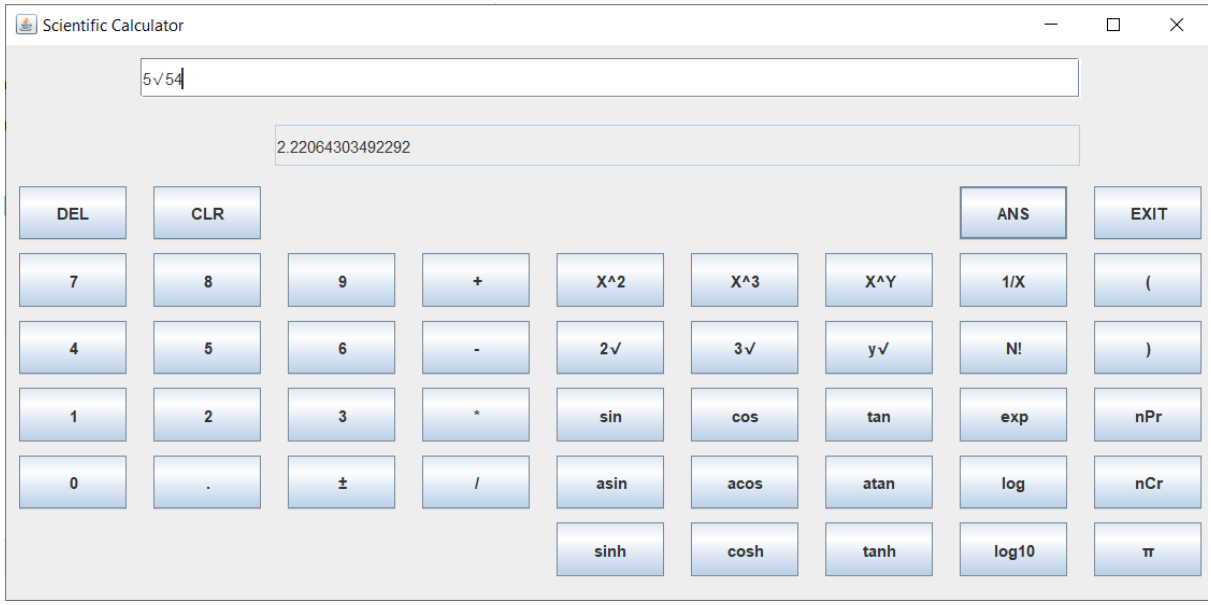
```
            System.out.println(stack);

            if(list.get(i).op.equals("@")) {
                stack.add(list.get(i));
            } else if(list.get(i).op.equals("!")) {
                Sci_Calculator.Node n = stack.pop();

                System.out.println("node is " + n);

                if (n.number != (long)n.number) {
                    throw new Exception();
                }

                long p = (long) n.number;

                System.out.println(p);
                p = factorial(p);
                System.out.println("factorial of p");
                stack.add(new Sci_Calculator.Node("@",p));
            } else {

                String op =  list.get(i).op;
                double operand2 = stack.pop().number;
                double operand1 = stack.pop().number;

                System.out.println("Before solve operator");
                double ans = solveOperator(operand1,operand2,op);
                System.out.println(ans);
                stack.add(new Sci_Calculator.Node("@",ans));
            }

        }

        System.out.println(stack);
        if (stack.size() != 1) {
            throw new Exception();
        }
        System.out.println("sine worked fine");
        return stack.peek().number;
    }

    static class Node {
        String op;
        double number;

        public Node(String op, double number) {
            this.op = op;
            this.number = number;
```

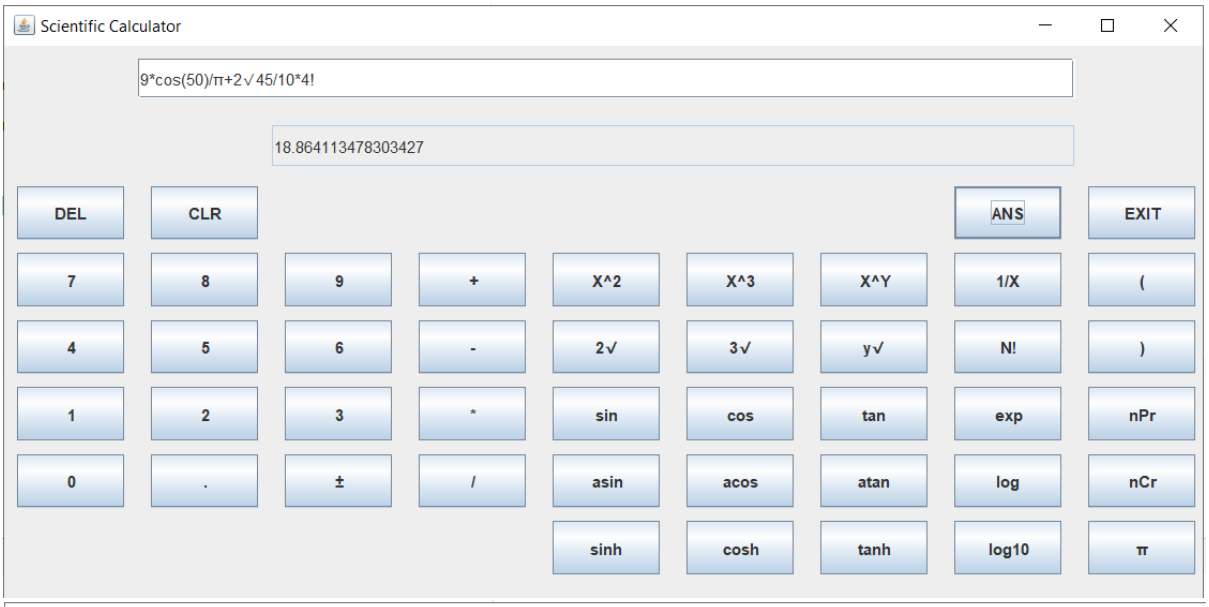
```
        }

        @Override
        public String toString() {
            return "{" +
                    op +
                    ", " + number +
                    '}';
        }
    }

    public static void main(String[] args) {
        Calculator c = new Calculator();
    }
}
```

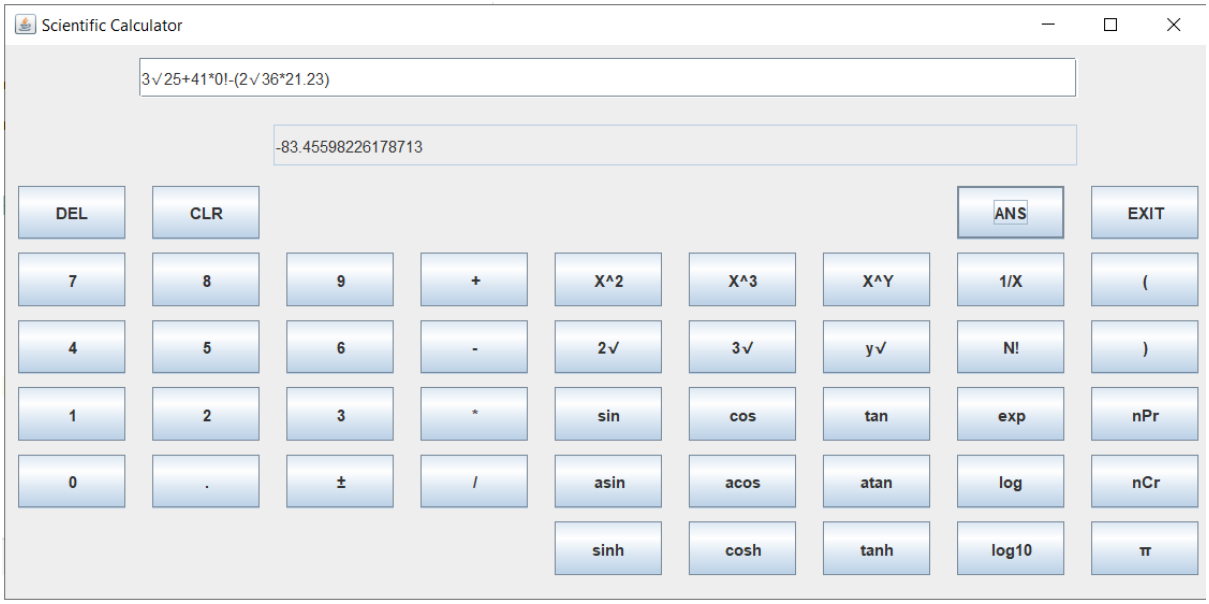
Solution

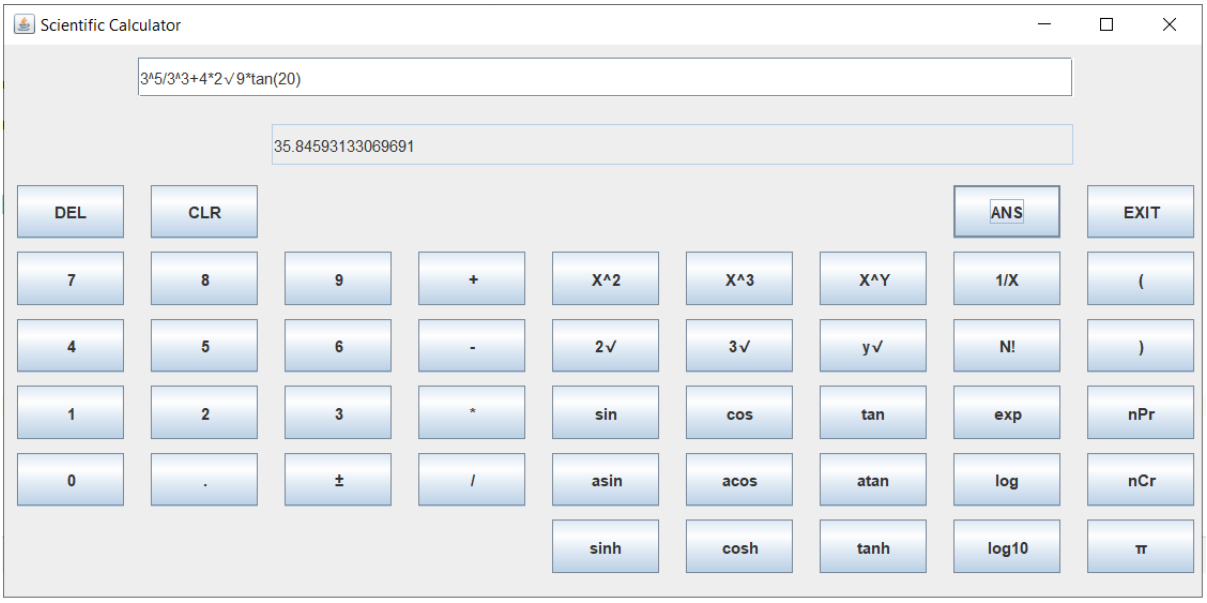

Scientific Calculator
(10^7)*8/9-7*log(64)*4!+3^8
8894751.196530884
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π

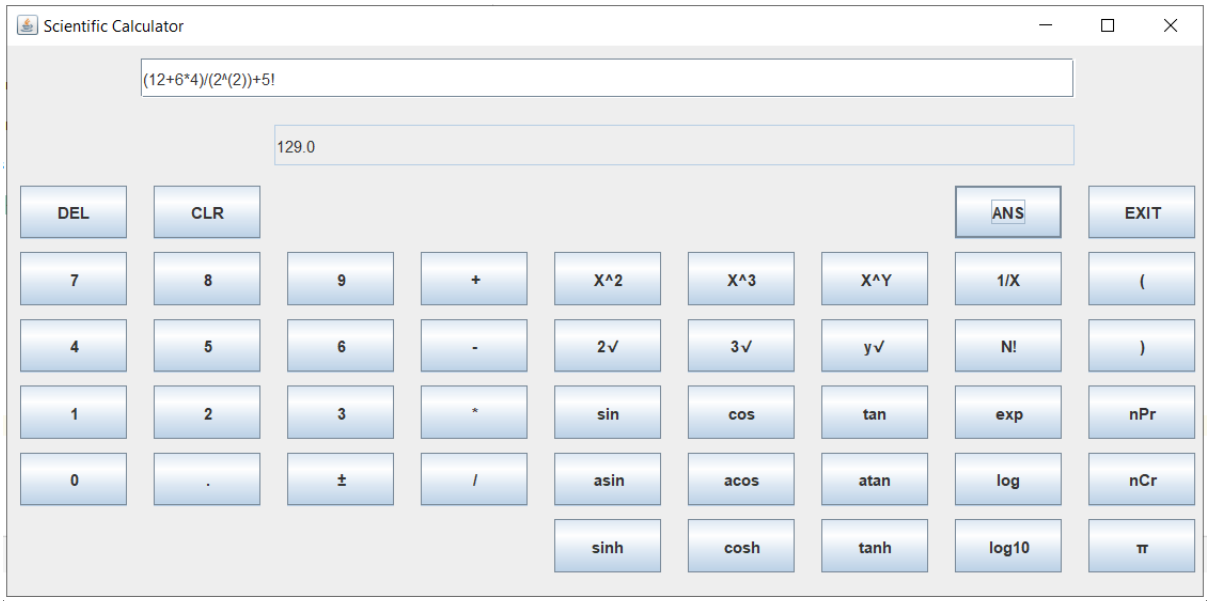

Scientific Calculator
(10^7)*8/9-7*log10(64)*4!+3^8
8895146.450653259
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π

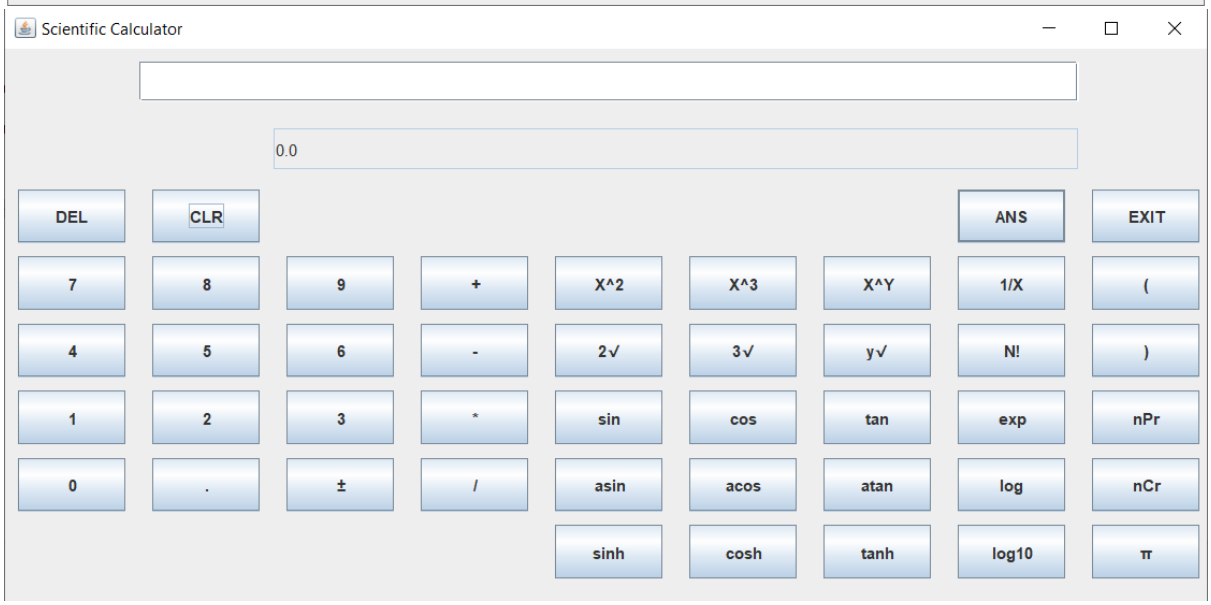

Scientific Calculator
5√54
2.22064303492292
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π


Scientific Calculator
9*cos(50)/π+2√45/10*4!
18.864113478303427
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π


Scientific Calculator
3√25+41*0!-(2√36*21.23)
-83.45598226178713
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π


Scientific Calculator
3^5/3^3+4*2√9*tan(20)
35.84593133069691
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π


Scientific Calculator
(12+6*4)/(2^(2))+5!
129.0
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π


Scientific Calculator
0.0
DEL
CLR
ANS
EXIT
7
8
9
+
X^2
X^3
X^Y
1/X
(
4
5
6
-
2√
3√
y√
N!
)
1
2
3
*
sin
cos
tan
exp
nPr
0
.
±
/
asin
acos
atan
log
nCr
sinh
cosh
tanh
log10
π