ASSIGNMENT - 2

NAME : Eshana Singh
ROLL : 2328093
SUB : OOPJ

1.Write a java program to design a scientific calculator using swing and event handling.
CODE:

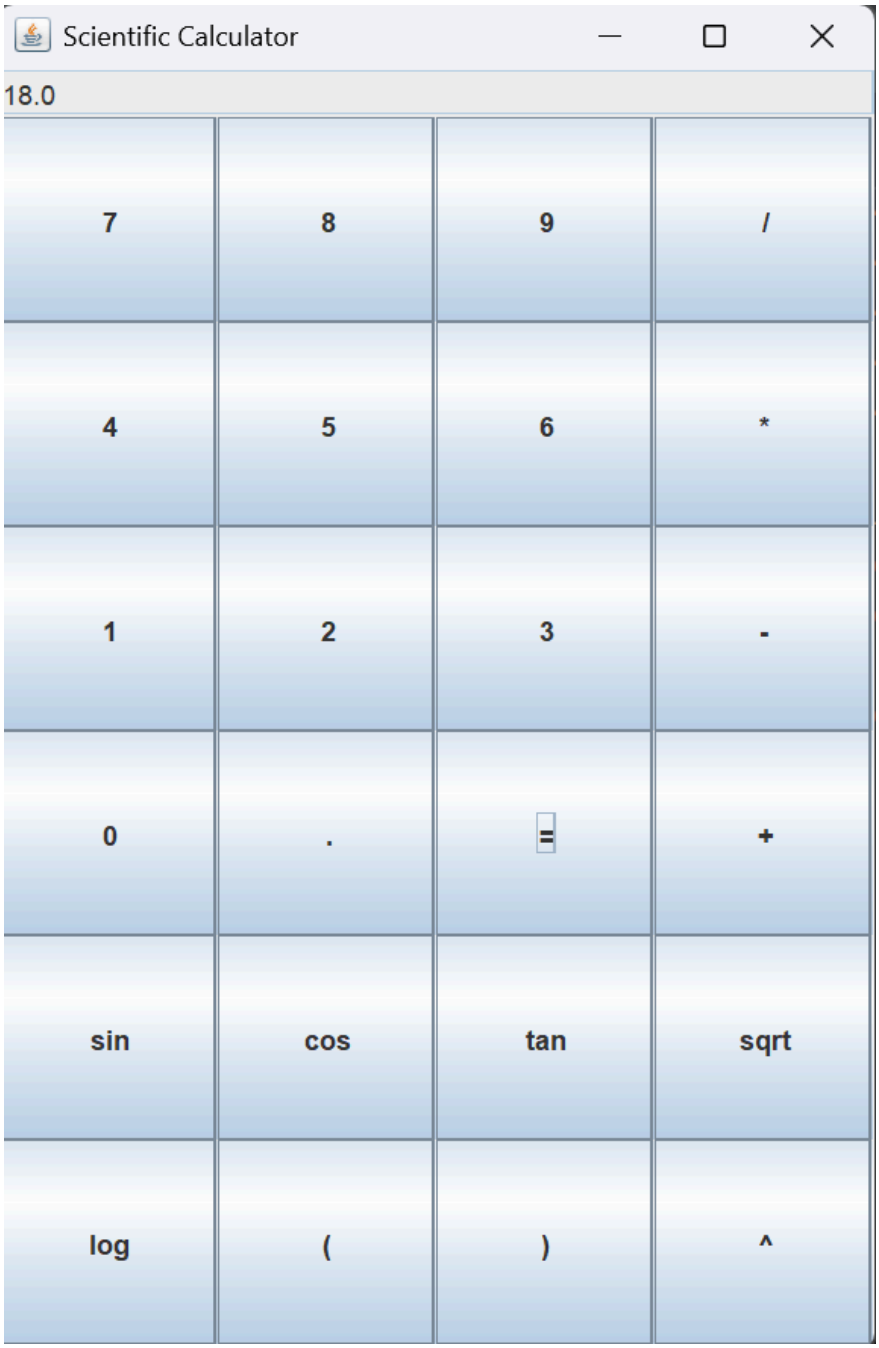
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Stack;

public class ScientificCalculator extends JFrame implements ActionListener {
   private JTextField display;
   private StringBuilder expression;

   public ScientificCalculator() {
       expression = new StringBuilder();
       setTitle("Scientific Calculator");
       setSize(400, 600);
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       setLayout(new BorderLayout());

       display = new JTextField();
       display.setEditable(false);
       add(display, BorderLayout.NORTH);

       JPanel buttonPanel = new JPanel();
       buttonPanel.setLayout(new GridLayout(6, 4));

       String[] buttons = {
               "7", "8", "9", "/",
               "4", "5", "6", "*",
               "1", "2", "3", "-",
               "0", ".", "=", "+",
               "sin", "cos", "tan", "sqrt",
               "log", "(", ")", "^"
       };

       for (String text : buttons) {
           JButton button = new JButton(text);
           button.addActionListener(this);
           buttonPanel.add(button);
```

```
        }

        add(buttonPanel, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        if (command.equals("=")) {
            try {
                double result = evaluate(expression.toString());
                display.setText(Double.toString(result));
                expression.setLength(0); // Clear the expression
            } catch (Exception ex) {
                display.setText("Error");
                expression.setLength(0);
            }
        } else if (command.equals("C")) {
            expression.setLength(0);
            display.setText("");
        } else {
            expression.append(command);
            display.setText(expression.toString());
        }
    }

    private double evaluate(String expr) {
        // Basic evaluation logic (you can enhance this)
        Stack<Double> numbers = new Stack<>();
        Stack<Character> operators = new Stack<>();

        for (int i = 0; i < expr.length(); i++) {
            char ch = expr.charAt(i);

            if (Character.isDigit(ch)) {
                StringBuilder sb = new StringBuilder();
                while (i < expr.length() && (Character.isDigit(expr.charAt(i))
|| expr.charAt(i) == '.')) {
                    sb.append(expr.charAt(i++));
                }
                numbers.push(Double.parseDouble(sb.toString()));
                i--; // To counter the last increment of i
            } else if ("+-*/^".indexOf(ch) != -1) {
                while (!operators.isEmpty() && precedence(ch) <=
precedence(operators.peek())) {
                    numbers.push(applyOperation(operators.pop(), numbers.pop(),
numbers.pop()));
                }
                operators.push(ch);
```

```
            } else if (ch == 's') { // sin
                i += 2; // Skip 'in'
                double value = numbers.pop();
                numbers.push(Math.sin(Math.toRadians(value)));
            } else if (ch == 'c') { // cos
                i += 2; // Skip 'os'
                double value = numbers.pop();
                numbers.push(Math.cos(Math.toRadians(value)));
            } else if (ch == 't') { // tan
                i += 2; // Skip 'an'
                double value = numbers.pop();
                numbers.push(Math.tan(Math.toRadians(value)));
            } else if (ch == 's' && expr.startsWith("sqrt", i)) { // sqrt
                i += 3; // Skip 'qrt'
                double value = numbers.pop();
                numbers.push(Math.sqrt(value));
            } else if (ch == 'l' && expr.startsWith("log", i)) { // log
                i += 2; // Skip 'og'
                double value = numbers.pop();
                numbers.push(Math.log10(value));
            }
        }

        while (!operators.isEmpty()) {
            numbers.push(applyOperation(operators.pop(), numbers.pop(),
numbers.pop()));
        }

        return numbers.pop();
    }

    private int precedence(char op) {
        switch (op) {
            case '+':
            case '-':
                return 1;
            case '*':
            case '/':
                return 2;
            case '^':
                return 3;
            default:
                return -1;
        }
    }

    private double applyOperation(char op, double b, double a) {
        switch (op) {
            case '+': return a + b;
```

```
            case '-': return a - b;
            case '*': return a * b;
            case '/':
                if (b == 0) throw new UnsupportedOperationException("Cannot
divide by zero");
                return a / b;
            case '^': return Math.pow(a, b);
            default: return 0;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            ScientificCalculator calculator = new ScientificCalculator();
            calculator.setVisible(true);
        });
    }
}
```

OUTPUT:



2. Write a java program to design a two player tic-tac-toe game using swing and event handing.

CODE:

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

public class tictactoe implements ActionListener {
   private JFrame frame;
```

```
    private JPanel panel;
    private JButton[] buttons = new JButton[9];
    private boolean xTurn = true;

    public tictactoe() {
        frame = new JFrame("Tic-Tac-Toe");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        panel = new JPanel();
        panel.setLayout(new GridLayout(3, 3));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        for (int i = 0; i < 9; i++) {
            buttons[i] = new JButton();
            buttons[i].setFont(new Font("Arial", Font.PLAIN, 60));
            buttons[i].setFocusPainted(false);
            buttons[i].addActionListener(this);
            panel.add(buttons[i]);
        }

        frame.add(panel);
        frame.setSize(300, 300);
        frame.setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JButton clickedButton = (JButton) e.getSource();

        if (!clickedButton.getText().equals("")) {
            return; // Ignore click if button already has a value
        }

        clickedButton.setText(xTurn ? "X" : "O");
        xTurn = !xTurn;

        if (checkForWinner()) {
            JOptionPane.showMessageDialog(frame, (xTurn ? "O" : "X") + "
wins!");
            resetGame();
        } else if (isBoardFull()) {
            JOptionPane.showMessageDialog(frame, "It's a tie!");
            resetGame();
        }
    }

    private boolean checkForWinner() {
        String[][] winningCombinations = {
                {"0", "1", "2"}, {"3", "4", "5"}, {"6", "7", "8"}, // Rows
                {"0", "3", "6"}, {"1", "4", "7"}, {"2", "5", "8"}, // Columns
```

```
                {"0", "4", "8"}, {"2", "4", "6"} // Diagonals
        };

        for (String[] combination : winningCombinations) {
            if (!buttons[Integer.parseInt(combination[0])].getText().equals("")
&&

buttons[Integer.parseInt(combination[0])].getText().equals(buttons[Integer.pars
eInt(combination[1])].getText()) &&

buttons[Integer.parseInt(combination[1])].getText().equals(buttons[Integer.pars
eInt(combination[2])].getText())) {
                return true;
            }
        }
        return false;
    }

    private boolean isBoardFull() {
        for (JButton button : buttons) {
            if (button.getText().equals("")) {
                return false;
            }
        }
        return true;
    }

    private void resetGame() {
        for (JButton button : buttons) {
            button.setText("");
        }
        xTurn = true;
    }

    public static void main(String[] args) {
        new tictactoe();
    }
}
```

OUTPUT: