Final Project of Calculator

Name of the Team: Group Array

Team Members: Jackie Fine

Katherine Garcia

Saransh Ketulkumar Shah

Vaishnavi Dasari

Bangling Yin

Our final project is the Calculator program. The Java code we provided constitutes a fully functional calculator application that leverages various core programming concepts, including multiple classes, methods, and user interaction. Comprising three essential classes - `UserInterface`, `Calculations`, and `Calculator` - the program employs an effective user interface and a calculation engine to perform arithmetic operations and advanced mathematical functions.

The 'Calculator' class serves as the main entry point, orchestrating the instantiation of the graphical user interface ('UserInterface') and the calculation logic ('Calculations'). This class enables the calculator to be displayed on-screen and handles user interactions, connecting the interface with the underlying calculations.

The 'UserInterface' class provides the graphical layout for the calculator, presenting users with buttons for arithmetic operations, scientific functions, and numerical input. This class facilitates user interaction, translating user actions into method calls to the 'Calculations' class, which performs the actual computations.

The 'Calculations' class is responsible for executing arithmetic calculations, managing mathematical functions, and processing user input. It contains the logic for addition, subtraction, multiplication, division, trigonometric functions, factorials, and other advanced mathematical operations.

The program's purpose is to offer users a comprehensive calculator tool capable of handling various mathematical operations. It's equipped with a user-friendly interface, enabling users to perform simple arithmetic calculations as well as more complex mathematical functions. Notations and comments within the code elucidate complex logic or functionalist, enhancing readability and understanding for future development and maintenance .

Instructions on how to use the calculator:

1. Initialize the Calculator by creating a new instance variable
2. Perform Basic Operations
    a. Addition (+): Press the "+" button.
    b. Subtraction (-): Press the "-" button.
    c. Multiplication (*): Press the "*" button.

    d. Division (/): Press the "/" button.
3. Explore Advanced Operations or Functions
    a. Trigonometric Functions: Use buttons like "sin," "cos," and "tan."
    b. Factorial (n!): Press the "n!" button.
    c. Exponentiation (x^y): Press the "x^y" button.
    d. Square Root (√): Press the "√" button.
4. To Clear the Display, press ‘C’
5. View Information about the Calculator Author & Version
6. Close the calculator window to exit the application

Code: Java Calculator

Calculator Class

```
/**
 * This is the main class for the calculator. This class will focus on
 * creating the calculator that shows on the screen. This calculator will
 * work with the calculator with GUI and calculator engine.
 *
 *
 * @author (Katherine Garcia) & Team Array
 * @version December 16th, 2023)
 */
public class Calculator
{
   // calls the gui from interface class
   private UserInterface gui;
   //calls the engine from calculations class
   private Calculations engine;

   /**
    * Shows the new calculator
    */
   public Calculator()
   {
      // initializes instance variables of engine and gui
      engine = new Calculations();
      gui = new UserInterface(engine);
   }

   /**
    * The method show() will make the calculator visible on the screen.
    *
    *
    * @return    the calculator will show if the boolean is true.
    */
   public void show()
   {
      //the gui will show if it is true
      gui.setVisible(true);
   }
}
```

UserInterface Class

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

/**
 * This the user interface of the calculator. The user will be able to
 * interact with the calculator. This class is responsible for putting up the
 * calculator on the screen. It will refer to the calculations engine to
 * complete the task.
 *
 * @author: Jackie Fine & Team Array
 * @version 2023.12.16
 */
public class UserInterface
{
   private Calculations calc;
   private boolean showingAuthor;
   private JFrame frame;
   private JTextField displayAnswer;
   private JLabel status;
   /**
    * Create a user interface.
    * @param engine The calculator engine.
    */
   public UserInterface(Calculations engine)
   {
      calc = engine;
      showingAuthor = true;
      makeFrame();
      frame.setVisible(true);
   }
   /**
    * Set the visibility of the interface.
    * @param visible true if the interface is to be made visible, false otherwise.
    */
   public void setVisible(boolean visible)
   {
      frame.setVisible(visible);
```

```
}
/**
 * Make the frame for the user interface.
 */
private void makeFrame()
{
    frame = new JFrame(calc.getTitle());

    JPanel contentPane = (JPanel)frame.getContentPane();
    contentPane.setLayout(new BorderLayout(75, 75));
    contentPane.setBorder(new EmptyBorder( 50, 50, 50, 50));



    displayAnswer = new JTextField();
    contentPane.add(displayAnswer, BorderLayout.NORTH);



    JPanel buttonPanel = new JPanel(new GridLayout(8, 5));
        addButton(buttonPanel, "+", () -> calc.plus());
        addButton(buttonPanel, "-", () -> calc.minus());
        addButton(buttonPanel, "*", () -> calc.multiply());
        addButton(buttonPanel, "/'", () -> calc.divide());
        addButton(buttonPanel, "C", () -> calc.clear());

        addButton(buttonPanel, "%", () -> calc.percent());
        addButton(buttonPanel, "sin", () -> calc.sin());
        addButton(buttonPanel, "cos", () -> calc.cos());
        addButton(buttonPanel, "tan", () -> calc.tan ());
        addButton(buttonPanel, "nCr", () -> calc.ncr());

        addButton(buttonPanel, "n!", () -> calc.nabs());
        addButton(buttonPanel, "cosec", () -> calc.cosec());
        addButton(buttonPanel, "sec", () -> calc.sec());
        addButton(buttonPanel, "cot", () -> calc.cot());
        addButton(buttonPanel, "nPr", () -> calc.npr());

        addButton(buttonPanel, "√", () -> calc.sqrt());
        addButton(buttonPanel, "x^2", () -> calc.xsqrd());
        addButton(buttonPanel, "x^3", () -> calc.xcube());
        addButton(buttonPanel, "x^y", () -> calc.expon());
```

```
        addButton(buttonPanel, "x^-1", () -> calc.minone());

        addButton(buttonPanel, "e", () -> calc.e());
        addButton(buttonPanel, "e^x", () -> calc.etox());
        addButton(buttonPanel, "log", () -> calc.log());
        addButton(buttonPanel, "ln", () -> calc.ln());
        addButton(buttonPanel, "|x|", () -> calc.absolx());

        addButton(buttonPanel, "π", () -> calc.pi());
        addButton(buttonPanel, "nπ", () -> calc.npi());
        addButton(buttonPanel, "2^x", () -> calc.twotox());
        addNumberButton(buttonPanel, 0);
        addNumberButton(buttonPanel, 1);

        addNumberButton(buttonPanel, 2);
        addNumberButton(buttonPanel, 3);
        addNumberButton(buttonPanel, 4);
        addNumberButton(buttonPanel, 5);
        addNumberButton(buttonPanel, 6);

        addNumberButton(buttonPanel, 7);
        addNumberButton(buttonPanel, 8);
        addNumberButton(buttonPanel, 9);
        addButton(buttonPanel, "Negate", () -> calc.negate());
        addButton(buttonPanel, "=", () -> calc.calculateResult());

    contentPane.add(buttonPanel, BorderLayout.CENTER);

    status = new JLabel(calc.getAuthor());
    contentPane.add(status, BorderLayout.SOUTH);

    frame.pack();
}
/**
 * Add a button to the button panel.
 * @param panel The panel to receive the button.
 * @param buttonText The text for the button.
 * @param action Action to be taken by the button.
 */
private void addButton(Container panel, String buttonText, ButtonAction action)
```

```
{
    JButton button = new JButton(buttonText);
    button.addActionListener(e -> { action.act(); redisplay(); });
    panel.add(button);
}
/**
 * Add a number button to the button panel.
 * @param panel The panel to receive the button.
 * @param digit The single digit on the button.
 */
private void addNumberButton(Container panel, int digit)
{
    addButton(panel, "" + digit, () -> calc.numberPressed(digit));
}
/**
 * Update the interface display to show the current value of the
 * calculator.
 */
private void redisplay() {
    displayAnswer.setText("" + calc.getDisplayValue());
}
/**
 * Toggle the info display in the calculator's status area between the
 * author and version information.
 */
private void showInfo()
{
    if(showingAuthor)
        status.setText(calc.getVersion());
    else
        status.setText(calc.getAuthor());
    showingAuthor = !showingAuthor;
}
/**
 * Functional interface for button actions.
 */
@FunctionalInterface
private interface ButtonAction
{
    /**
```

```
     * Act on a button press.
     */
    public void act();
  }
}
```

Calculations Class

```
/**
 * This is the calculations part of the calculator. This class will focus on
 * calculating all the functions and operations.
 *
 * @author (Saransh Ketulkumar Shah ) & Team Array
 * @version December 16th, 2023)
 */

public class Calculations  {
  private boolean buildingDisplayValue;
  private boolean haveLeftOperand;
  private char lastOperator;
  private double displayValue;
  private double leftOperand;
  private boolean inDegrees;
  public Calculations() {
    clear(); //this opens the calculator with an empty screen
  }
  public double getDisplayValue() {
    return displayValue; //this displays the value of the button pressed
  }
  public void numberPressed(int number) { //displays the number pressed
    if (buildingDisplayValue) {
      displayValue = displayValue * 10 + number;
    } else {
      displayValue = number;
      buildingDisplayValue = true;
    }
  }
  public void plus() { //adds numbers
    applyOperator('+');
  }
  public void minus() { //subtracts numbers
```

```
    applyOperator('-');
}
public void multiply() { //multiplies numbers
     applyOperator('*');
}
public void divide() { //divides numbers
    applyOperator('/');
}
public void percent() { //shows number in percents
    if (buildingDisplayValue) {
        displayValue = leftOperand * displayValue / 100.0;
    }
}
public void sin() { //takes the sin of a number
    displayValue = inDegrees ? Math.sin(Math.toRadians(displayValue)) : Math.sin(displayValue);
}
public void cos() { // takes the cos of a number
   displayValue = inDegrees ? Math.cos(Math.toRadians(displayValue)) : Math.cos(displayValue);
}
public void tan() { //takes the tan of a number
  displayValue = inDegrees ? Math.tan(Math.toRadians(displayValue)) : Math.tan(displayValue);
}
public void cosec() { //takes the cosec of a number
    double sinValue = sinValue(displayValue);
    if (sinValue != 0) {
        displayValue = 1 / sinValue;
    } else {
        keySequenceError(); //shows error if invalid number
    }
}
public void sec() { //takes the sec of a number
    double cosValue = cosValue(displayValue);
    if (cosValue != 0) {
        displayValue = 1 / cosValue;
    } else {
        keySequenceError();
    }
}
public void cot() { //takes the cot of a number
    double tanValue = tanValue(displayValue);
```

```
    if (tanValue != 0) {
        displayValue = 1 / tanValue;
    } else {
        keySequenceError();
    }
}
public void nabs() { //takes into account factorials
    if (buildingDisplayValue) {
        int n = (int) displayValue;
        if (n >= 0) {
            displayValue = factorial(n);
        } else {
            keySequenceError();
        }
    }
}
public void npr() { //takes into account factorials
    if (buildingDisplayValue) {
        int n = (int) displayValue;
        int r = (int) leftOperand;
        if (n >= r && n >= 0 && r >= 0) {
            displayValue = factorial(n) / factorial(n - r);
        } else {
            keySequenceError();
        }
    }
}
public void ncr() { //takes into account factorials
    if (buildingDisplayValue) {
        int n = (int) displayValue;
        int r = (int) leftOperand;
        if (n >= r && n >= 0 && r >= 0) {
            displayValue = factorial(n) / (factorial(r) * factorial(n - r));
        } else {
            keySequenceError();
        }
    }
}
public void sqrt() { //takes the square root of a number
    if (buildingDisplayValue) {
```

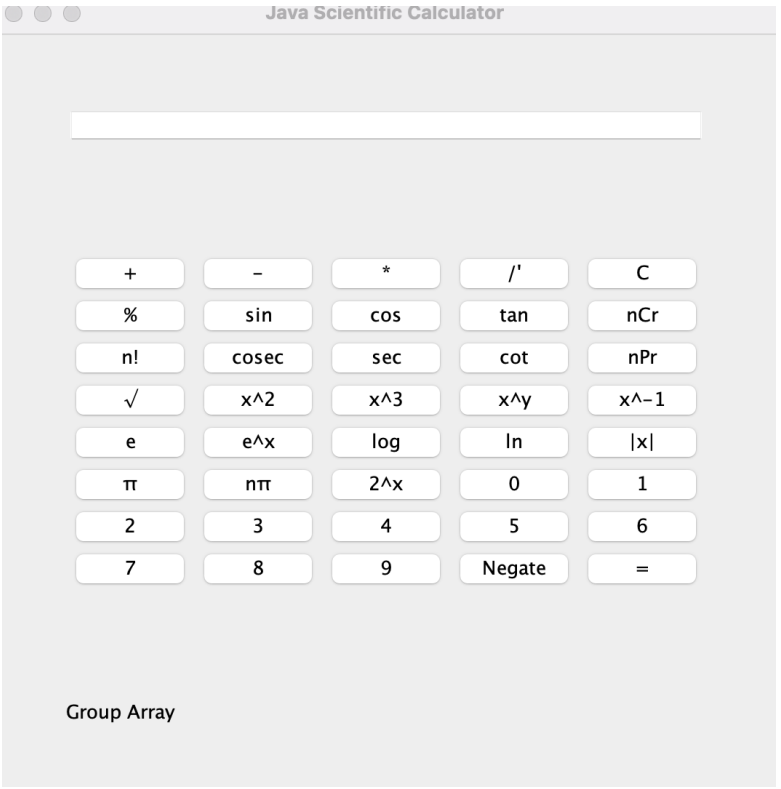
```
        if (displayValue >= 0) {
            displayValue = Math.sqrt(displayValue);
        } else {
            keySequenceError();
        }
    }
}
public void xsqrd() { //takes the number squared
     displayValue = Math.pow(displayValue, 2);
}
public void xcube() { //takes the number cubed
     displayValue = Math.pow(displayValue, 3);
}
public void expon() { //takes the exponent of a number
    displayValue = Math.pow(leftOperand, displayValue);
    haveLeftOperand = true;
    leftOperand = displayValue;
}
public void minone() { //takes the x minus 1 of a number
    if (displayValue != 0) {
        displayValue = 1 / displayValue;
    } else {
        keySequenceError();
    }
}
public void e() { //takes the natural e of a number
     displayValue = Math.E;
}
public void etox() { //takes the e raised to x of a number
      displayValue = Math.exp(displayValue);
}
public void log() { //takes the log of a number
    if (displayValue > 0) {
        displayValue = Math.log10(displayValue);
    } else {
        keySequenceError();
    }
}
public void ln() { //takes the natural log of a number
    if (displayValue > 0) {
```

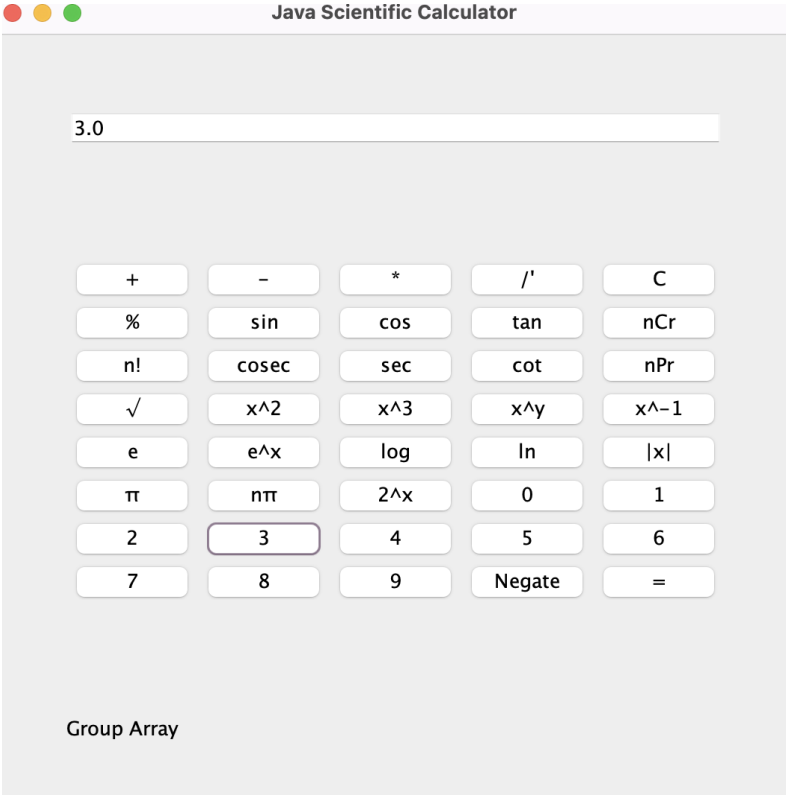
```
        displayValue = Math.log(displayValue);
    } else {
        keySequenceError();
    }
}
public void absolx() { // takes the absolute value of a number
       displayValue = Math.abs(displayValue);
}
public void pi() { //calculates pi
   displayValue = Math.PI;
}
 public void npi() { //calculate the number times pi
    if (buildingDisplayValue) {
        displayValue *= Math.PI;
    }
}
public void twotox() { //calculates 2 to the x
    if (buildingDisplayValue) {
        displayValue = Math.pow(2, displayValue);
    }
}
public void tentox() { // calculate 10 to the x
    if (buildingDisplayValue) {
        displayValue = Math.pow(10, displayValue);
    }
}
public boolean isInDegrees() {
     return inDegrees;
}
public void calculateResult() { //calculates the results
    if (!haveLeftOperand || lastOperator == '?') {
        return; // Do nothing if there's no operation to perform
    }
    switch (lastOperator) {
        case '+':
            displayValue = leftOperand + displayValue;
            break;
        case '-':
            displayValue = leftOperand - displayValue;
            break;
```

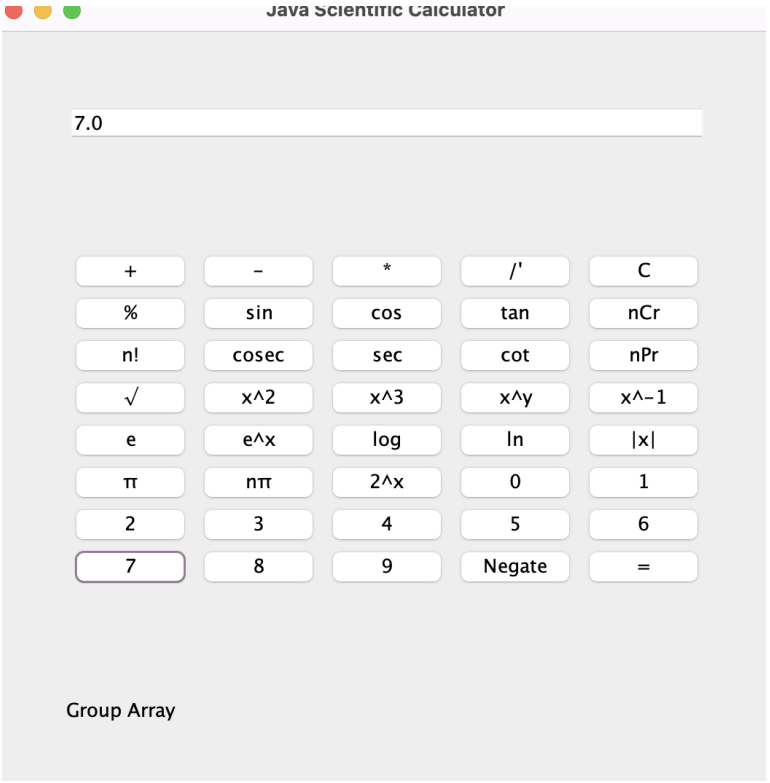
```
        case '*':
            displayValue = leftOperand * displayValue;
            break;
        case '/':
            if (displayValue != 0) {
                displayValue = leftOperand / displayValue;
            } else {
                keySequenceError();
            }
            break;
        // ... other cases as necessary ...
    }
    leftOperand = 0;
    haveLeftOperand = false;
    lastOperator = '?';
}
 public void negate() { //turns the number to a negative
      displayValue = -displayValue;
}
private void applyOperator(char operator) {
    if (!buildingDisplayValue && !(haveLeftOperand && lastOperator == '?')) {
        keySequenceError();
        return;
    }

    if (lastOperator != '?') {
        calculateResult();
    } else {
        haveLeftOperand = true;
        leftOperand = displayValue;
    }
    lastOperator = operator;
    buildingDisplayValue = false;
}
private void keySequenceError() {
    System.out.println("A key sequence error has occurred.");
    clear();
}
public void clear() { //when C is pressed, it clears the screen
    lastOperator = '?';
```

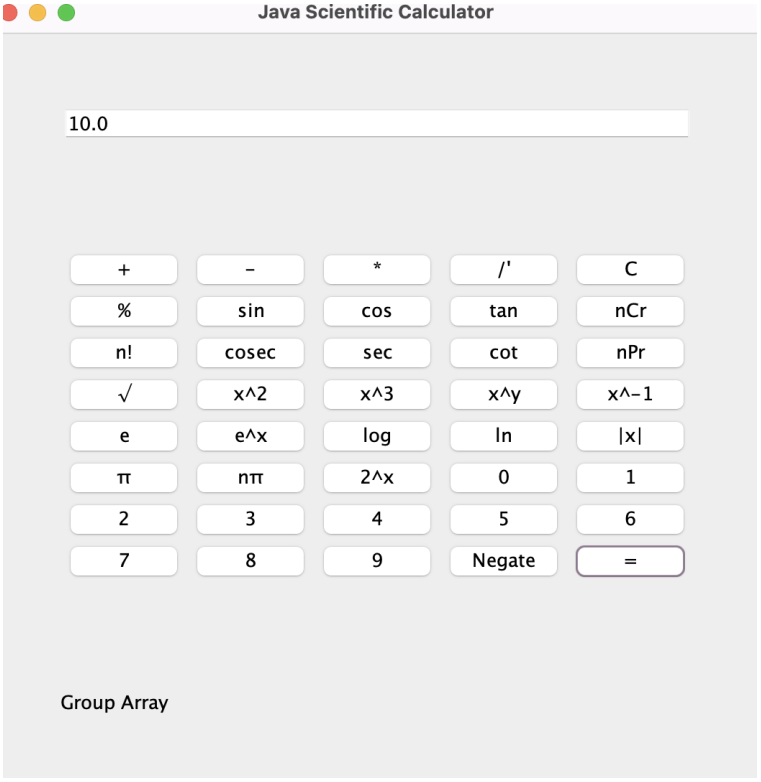
```
        haveLeftOperand = false;
        buildingDisplayValue = false;
        displayValue = 0;
    }
    public String getTitle() { //displays title of the calculator
      return "Java Scientific Calculator";
    }
    public String getAuthor() { //displays the author of the calculator
      return "Group Array";  // Our team name (Jackie's idea)
    }
    public String getVersion() { //displays the version of the calculator
       return "Version 1.0";
    }
    public void setDegrees(boolean inDegrees) {
        this.inDegrees = inDegrees;
    }
    private double sinValue(double value) {
        return inDegrees ? Math.sin(Math.toRadians(value)) : Math.sin(value);
    }
    private double cosValue(double value) {
        return inDegrees ? Math.cos(Math.toRadians(value)) : Math.cos(value);
    }
    private double tanValue(double value) {
        return inDegrees ? Math.tan(Math.toRadians(value)) : Math.tan(value);
    }
    private int factorial(int n) {
        int result = 1;
        for (int i = 2; i <= n; i++) {
            result *= i;
        }
        return result;
    }
}
```

Code Output:


Java Scientific Calculator
+ - * /' C
% sin cos tan nCr
n! cosec sec cot nPr
√ x^2 x^3 x^y x^-1
e e^x log ln |x|
π nπ 2^x 0 1
2 3 4 5 6
7 8 9 Negate =
Group Array


Java Scientific Calculator
3.0
+ - * /' C
% sin cos tan nCr
n! cosec sec cot nPr
√ x^2 x^3 x^y x^-1
e e^x log ln |x|
π nπ 2^x 0 1
2 3 4 5 6
7 8 9 Negate =
Group Array


Java Scientific Calculator
7.0
+
-
*
/'
C
%
sin
cos
tan
nCr
n!
cosec
sec
cot
nPr
√
x^2
x^3
x^y
x^-1
e
e^x
log
ln
|x|
π
nπ
2^x
0
1
2
3
4
5
6
7
8
9
Negate
=
Group Array


Java Scientific Calculator
10.0
+
-
*
/'
C
%
sin
cos
tan
nCr
n!
cosec
sec
cot
nPr
√
x^2
x^3
x^y
x^-1
e
e^x
log
ln
|x|
π
nπ
2^x
0
1
2
3
4
5
6
7
8
9
Negate
=
Group Array