PROJECT REPORT

Title: Create a native calculator using Java and Java SWING GUI

Abstract:

This project is about building a simple, native calculator application using Java and Java Swing for the GUI. The main goal is to create a basic yet functional calculator that can perform simple arithmetic operations. The focus is on making a clean and easy-to-use interface. We'll also add some basic error handling to manage things like invalid inputs. This project showcases how Java and Swing can be used to create desktop applications with a user-friendly interface.

Objective:

The objective of this project is to develop a native calculator application using Java and Java Swing for the graphical user interface (GUI). The calculator should perform basic arithmetic operations like addition, subtraction, multiplication, and division. This project will demonstrate the practical use of Java and Swing in building simple, effective desktop applications.

Java is a widely-used programming language known for its portability, robustness, and versatility. Swing, a part of Java's standard library, offers a rich set of GUI components that can be used to create sophisticated interfaces. By using these tools, this project aims to develop a calculator that can perform basic arithmetic operations, such as addition, subtraction, multiplication, and division.

Introduction:

The development of native calculator using Java and Java Swing provides a basic knowledge and covers the topics from basics in java such as creating classes to intermediate topics like exception handling and usage of libraries/toolkits such as swing and awt to create simple desktop applications.

This project involves building a simple calculator using Java and Java Swing for the graphical user interface. The goal is to create a simple calculator that performs basic arithmetic operations by covering the topics such as classes, functions, inheritance and swing concepts such as actionListeners, GUI elements etc.,

Additionally, the project emphasizes creating a user-friendly interface, ensuring that the calculator is intuitive and easy to use. It also includes implementing basic error handling to manage invalid inputs and division by zero scenarios, enhancing the application's reliability and robustness. This project serves as a practical exercise in applying Java programming skills and understanding the fundamentals of building desktop applications, providing a solid foundation for future learning and more complex software development projects.

Methodology:

Methodology in a project describes the approach and processes used to achieve the project's objectives. It outlines the steps taken, tools used, and techniques applied throughout the development process.

The development of this project involves several key steps and methodologies to ensure a functional calculator. Starting with the development environment, this project was developed using the programming language Java 22.0.1 and Java Swing is the toolkit used for the Graphical User Interface (GUI). The development environment also included an Integrated Development Environment (IDE). In this case I used Visual Studio Code (a.k.a. VS Code) as the IDE to develop this project.

Initial stage of this project is planning the User Interface (a.k.a. UI) Design. The development environment used for UI designing is FIGMA which is an online platform used to create interface designs. We categorize the objects such as Frame, Buttons, Text Field, Labels from the UI designing phase. Later we went to coding the UI design using Java Swing by using functions like JFrame, JPanel, JButton, JTextField. This executes the front-end of the program by displaying the buttons and the panel similar to the design we come up in the FIGMA.

Later we added actionListeners to the buttons so the buttons can create interaction and receive input from the user. ActionListeners are part of the awt package which supports well with the Swing since Swing considered as the upgraded version of awt applets. Later we collected the events which are the user inputs when they interacted with the front-end of the calculator by overriding the function called actionPerformed which carries the parameter ActionEvent. With the help of actionPerformed function we able to receive the input based on which button is the user pressing and display that input on the text area so the user can see the input he had given. This helps integrate the back-end with the front-end design.

We added some basic functional keys such as CLR (sym_CLR) to clear the screen and DEL (sym_DEL) to delete the last input given by the user and "." (sym_DEC) to calculate the floating-point variables which gives the decimal values apart from the buttons that perform mathematical operations such as "+" (sym_SUM), "-" (sym_SUB), "*" (sym_MUL), "/" (sym_DIV). We used naming conventions for the variables to identify the different buttons in this case: all the number buttons follow the naming convention of "Num_" which indicates that button receives numbers as input whereas the buttons that perform miscellaneous functions or arithmetic functions followed by the naming convention of "sym_" which indicates that these buttons perform system related functions such as clearing screen, deleting the functions, performing arithmetic operations.

The code for the calculator application is designed using Java and Swing, focusing on creating a functional and interactive desktop calculator. It begins by importing necessary Swing and event-handling libraries and declaring the Calculator class, which implements the ActionListener interface to manage user input and button actions. Key instance variables include input and result for storing numerical values and cal for tracking the current operation. The class also defines several Swing components, such as JFrame, JLabel, JTextField, and JButton, used to construct the calculator's interface.

The constructor initializes the user interface by calling methods to set up the frame (CreateInterface()), arrange components (InterfaceComponenets()), and add event listeners (AddInterfaceEventListener()). The CreateInterface() method configures the main frame's properties, including size and layout. The InterfaceComponenets() method defines the layout and positioning of various buttons and fields. The AddInterfaceEventListener() method attaches action listeners to each button to handle user interactions. Finally, the actionPerformed() method processes these interactions, performs the appropriate calculations based on the selected operation, and updates the display accordingly. This structure allows for a responsive and functional calculator application.

Code:

```
package App;


import javax.swing.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


class Calculator implements ActionListener{


    double input, result;

    String cal;


    /* Interface Components ------------------------------------------------------------------- */

        JFrame frame;

        JLabel label = new JLabel("");


        JTextField textView = new JTextField();


        // First Row

        JButton sym_CLR = new JButton("CLR");

        JButton sym_DEL = new JButton("DEL");

        JButton sym_MUL = new JButton("X");

        JButton sym_DIV = new JButton("/");
        // Second Row
        JButton Num_Seven = new JButton("7");
        JButton Num_Eight = new JButton("8");
        JButton Num_Nine = new JButton("9");
        JButton sym_SUB = new JButton("-");

        // Third Row
        JButton Num_Four = new JButton("4");
        JButton Num_Five = new JButton("5");
        JButton Num_Six = new JButton("6");
        JButton sym_SUM = new JButton("+");

        // Fourth Row
        JButton Num_One = new JButton("1");
        JButton Num_Two = new JButton("2");
        JButton Num_Three = new JButton("3");
        JButton sym_EQUAL = new JButton("=");

        // Fifth Row
        JButton Num_Zero = new JButton("0");
        JButton sym_DEC = new JButton(".");

    /* ------------------------------------------------------------------------------------------ */
```

```
Calculator(){
    CreateInterface();
    InterfaceComponenets();
    AddInterfaceEventListener();
}

public void CreateInterface() {
    // Basic Swing Layout
    frame = new JFrame("Calculator");

    frame.getContentPane().setLayout(null);
    frame.setLocationRelativeTo(null);
    frame.setResizable(false);
    frame.setSize(305, 400);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}

public void InterfaceComponenets() {
    /* Output Row ----------------------------------------------------------------------------- */
        label.setBounds(180, 0, 100, 30);
        frame.add(label);
        textView.setBounds(10, 40, 270, 60);
        textView.setEditable(false);
        textView.setHorizontalAlignment(SwingConstants.RIGHT);
        frame.add(textView);
    /* ---------------------------------------------------------------------------------------------- */
    /* First Row ----------------------------------------------------------------------------------- */
        sym_CLR.setBounds(10, 110, 60, 40);
        frame.add(sym_CLR);

        sym_DEL.setBounds(80, 110, 60, 40);
        frame.add(sym_DEL);

        sym_MUL.setBounds(150, 110, 60, 40);
        frame.add(sym_MUL);

        sym_DIV.setBounds(220, 110, 60, 40);
        frame.add(sym_DIV);
    /* ---------------------------------------------------------------------------------------------- */
    /* Second Row ---------------------------------------------------------------------------------- */
        Num_Seven.setBounds(10, 160, 60, 40);
        frame.add(Num_Seven);

        Num_Eight.setBounds(80, 160, 60, 40);
        frame.add(Num_Eight);
        Num_Nine.setBounds(150, 160, 60, 40);
        frame.add(Num_Nine);

        sym_SUB.setBounds(220, 160, 60, 40);
        frame.add(sym_SUB);
    /* ---------------------------------------------------------------------------------- */
    /* Third Row ------------------------------------------------------------------------- */
        Num_Four.setBounds(10, 210, 60, 40);
        frame.add(Num_Four);

        Num_Five.setBounds(80, 210, 60, 40);
        frame.add(Num_Five);

        Num_Six.setBounds(150, 210, 60, 40);
        frame.add(Num_Six);

        sym_SUM.setBounds(220, 210, 60, 40);
        frame.add(sym_SUM);
    /* ---------------------------------------------------------------------------------- */
    /* Fourth Row ------------------------------------------------------------------------ */
        Num_One.setBounds(10, 260, 60, 40);
        frame.add(Num_One);
```

```
      Num_Two.setBounds(80, 260, 60, 40);
      frame.add(Num_Two);

      Num_Three.setBounds(150, 260, 60, 40);
      frame.add(Num_Three);

      sym_EQUAL.setBounds(220, 260, 60, 90);
      frame.add(sym_EQUAL);
   /* -------------------------------------------------------------------------------- */
   /* Fifth Row ---------------------------------------------------------------------- */
      Num_Zero.setBounds(10, 310, 130, 40);
      frame.add(Num_Zero);

      sym_DEC.setBounds(150, 310, 60, 40);
      frame.add(sym_DEC);
   /* -------------------------------------------------------------------------------- */
}

public void AddInterfaceEventListener() {
   // 1st Row
   sym_CLR.addActionListener(this);
   sym_DEL.addActionListener(this);
   sym_MUL.addActionListener(this);
   sym_DIV.addActionListener(this);
```

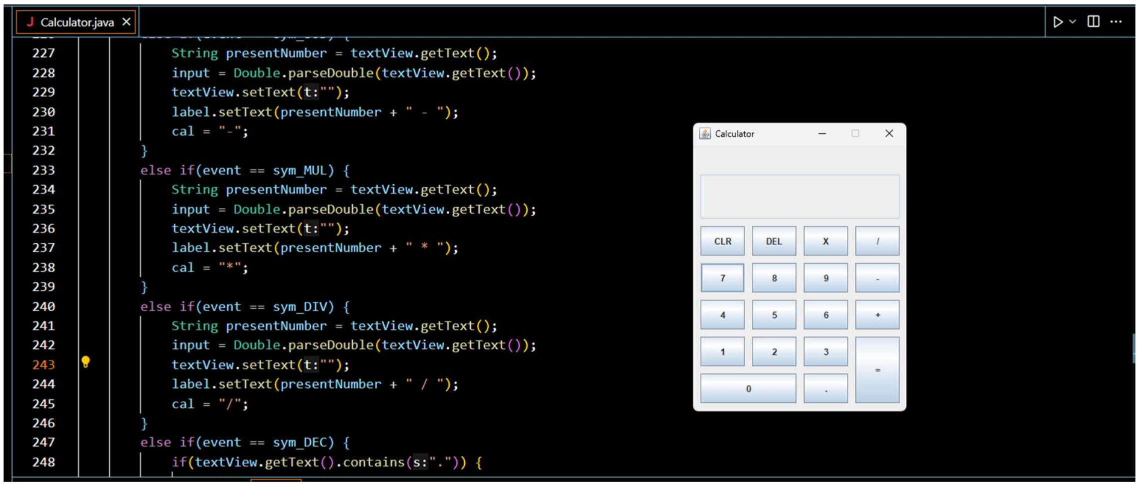
```
    // 2nd Row
    Num_Seven.addActionListener(this);
    Num_Eight.addActionListener(this);
    Num_Nine.addActionListener(this);
    sym_SUB.addActionListener(this);
    // 3rd Row
    Num_Four.addActionListener(this);
    Num_Five.addActionListener(this);
    Num_Six.addActionListener(this);
    sym_SUM.addActionListener(this);
    // 4th Row
    Num_One.addActionListener(this);
    Num_Two.addActionListener(this);
    Num_Three.addActionListener(this);
    sym_EQUAL.addActionListener(this);
    // 5th Row
    Num_Zero.addActionListener(this);
    sym_DEC.addActionListener(this);
}

@Override
public void actionPerformed(ActionEvent e) {
    Object event = e.getSource();
```

```
if(event == Num_One) {
    textView.setText(textView.getText() + "1");
}
else if(event == Num_Two) {
    textView.setText(textView.getText() + "2");
}
else if(event == Num_Three) {
    textView.setText(textView.getText() + "3");
}
else if(event == Num_Four) {
    textView.setText(textView.getText() + "4");
}
else if(event == Num_Five) {
    textView.setText(textView.getText() + "5");
}
else if(event == Num_Six) {
    textView.setText(textView.getText() + "6");
}
else if(event == Num_Seven) {
    textView.setText(textView.getText() + "7");
}
else if(event == Num_Eight) {
    textView.setText(textView.getText() + "8");
}
```

```
else if(event == Num_Nine) {
    textView.setText(textView.getText() + "9");
}
else if(event == Num_Zero) {
    if(textView.getText().equals("0")) {
        return;
    } else {
        textView.setText(textView.getText() + "0");
    }
}
else if(event == sym_CLR) {
    label.setText("");
    textView.setText("");
}
else if(event == sym_DEL) {
    int length = textView.getText().length();
    int num = length-1;
    if(length>0) {
        StringBuilder numString = new StringBuilder(textView.getText());
        numString.deleteCharAt(num);
        textView.setText(numString.toString());
    }
    if(textView.getText().endsWith("")){
        label.setText("");
    }
}
else if(event == sym_SUM) {
    String presentNumber = textView.getText();
    input = Double.parseDouble(textView.getText());
    textView.setText("");
    label.setText(presentNumber + " + ");
    cal = "+";
}
else if(event == sym_SUB) {
    String presentNumber = textView.getText();
    input = Double.parseDouble(textView.getText());
    textView.setText("");
    label.setText(presentNumber + " - ");
    cal = "-";
}
else if(event == sym_MUL) {
    String presentNumber = textView.getText();
    input = Double.parseDouble(textView.getText());
    textView.setText("");
    label.setText(presentNumber + " * ");
    cal = "*";
}
else if(event == sym_DIV) {
    String presentNumber = textView.getText();
    input = Double.parseDouble(textView.getText());
    textView.setText("");
    label.setText(presentNumber + " / ");
    cal = "/";
}
else if(event == sym_DEC) {
    if(textView.getText().contains(".")) {
        return;
    }
    else{
        textView.setText(textView.getText() + ".");
    }
}
else if(event == sym_EQUAL) {
    switch (cal) {
        case "+" :
            result = input + (Double.parseDouble(textView.getText()));

            if(Double.toString(result).endsWith(".0")) {
                textView.setText(Double.toString(result).replace(".0", ""));
            }
            else {
                textView.setText(Double.toString(result));
    }

    label.setText("");
    break;
case "-" :
    result = input - (Double.parseDouble(textView.getText()));

    if(Double.toString(result).endsWith(".0")) {
        textView.setText(Double.toString(result).replace(".0", ""));
    }
    else {
        textView.setText(Double.toString(result));
    }

    label.setText("");
    break;
case "*" :
    result = input * (Double.parseDouble(textView.getText()));

    if(Double.toString(result).endsWith(".0")) {
        textView.setText(Double.toString(result).replace(".0", ""));
    }
    else {
        textView.setText(Double.toString(result));
```

```
                    }

                    label.setText("");
                    break;
                case "/" :
                    result = input / (Double.parseDouble(textView.getText()));

                    if(Double.toString(result).endsWith(".0")) {
                        textView.setText(Double.toString(result).replace(".0", ""));
                    }
                    else {
                        textView.setText(Double.toString(result));
                    }

                    label.setText("");
                    break;
            }
        }
    }

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

Output



```
Calculator.java
            String presentNumber = textView.getText();
            input = Double.parseDouble(textView.getText());
            textView.setText(t:"");
            label.setText(presentNumber + " - ");
            cal = "-";
        }
        else if(event == sym_MUL) {
            String presentNumber = textView.getText();
            input = Double.parseDouble(textView.getText());
            textView.setText(t:"");
            label.setText(presentNumber + " * ");
            cal = "*";
        }
        else if(event == sym_DIV) {
            String presentNumber = textView.getText();
            input = Double.parseDouble(textView.getText());
            textView.setText(t:"");
            label.setText(presentNumber + " / ");
            cal = "/";
        }
        else if(event == sym_DEC) {
            if(textView.getText().contains(s:".")) {
```

Figure 1 Native Calculator build using Java and Java Swing GUI

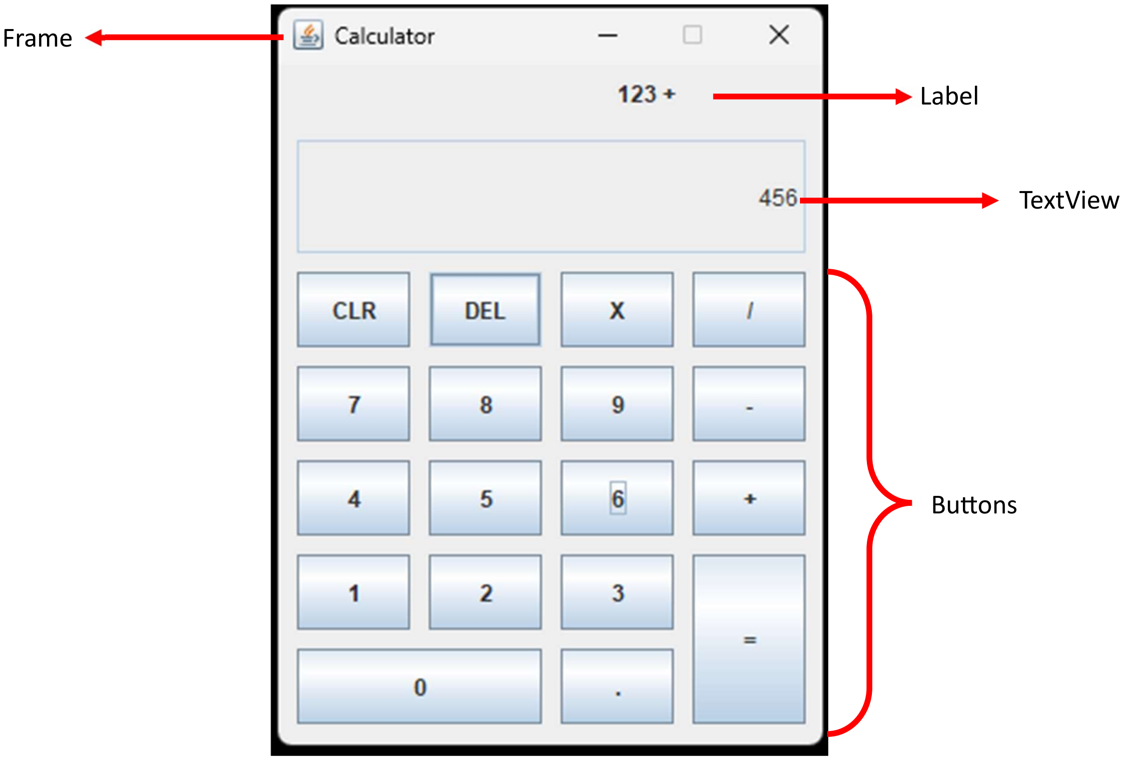


Figure 2 Components Present in the Calculator GUI

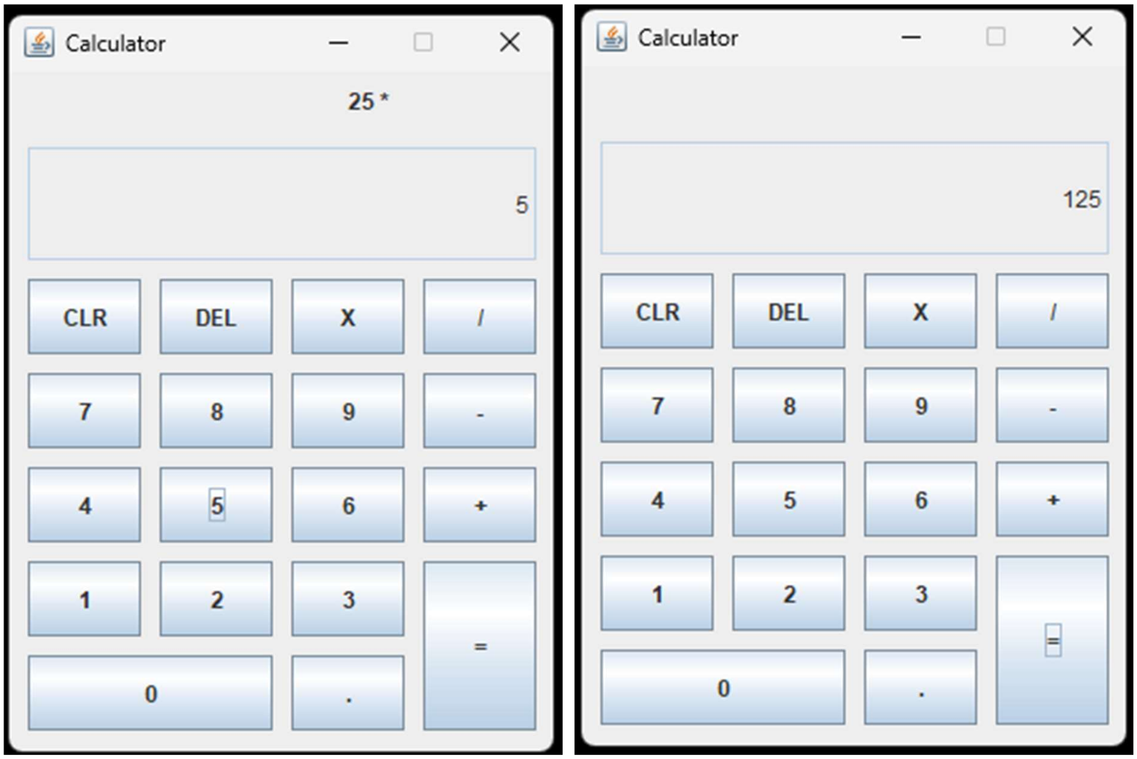


Figure 3 Multiplication Operation Performing by the Calculator by taking Inputs (on Left) and by displaying output (on Right)

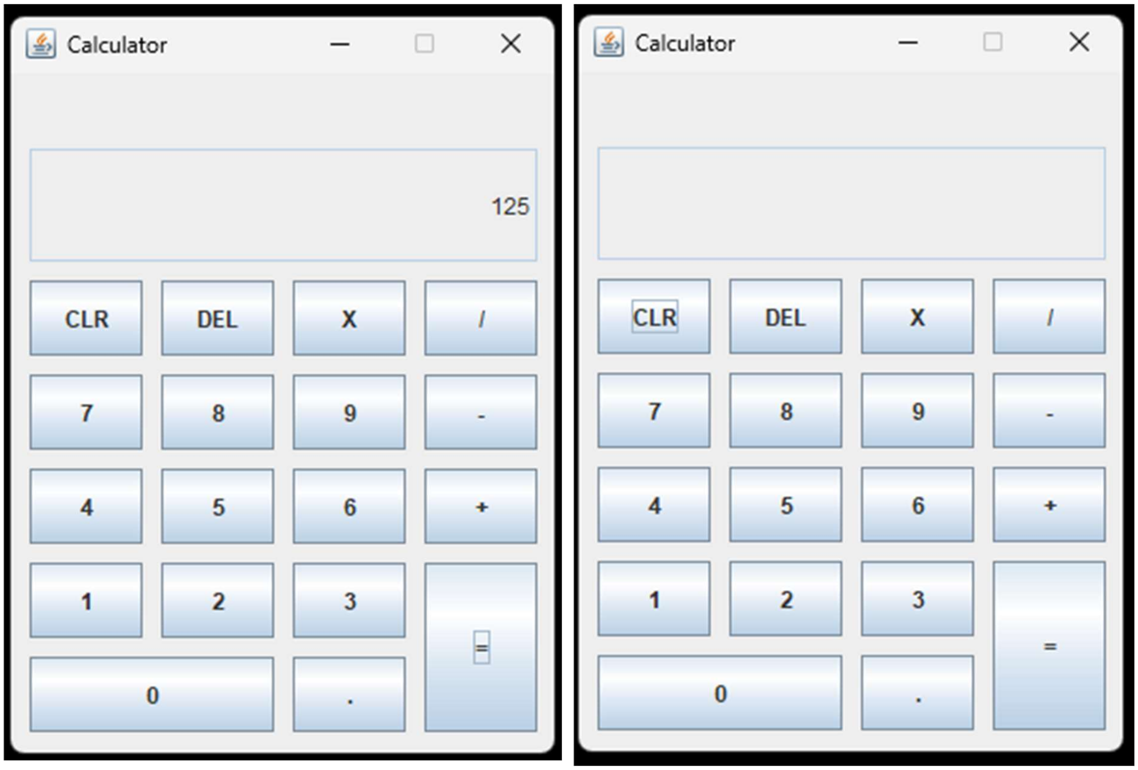


Figure 4 CLR Button clears the textView / Display and label

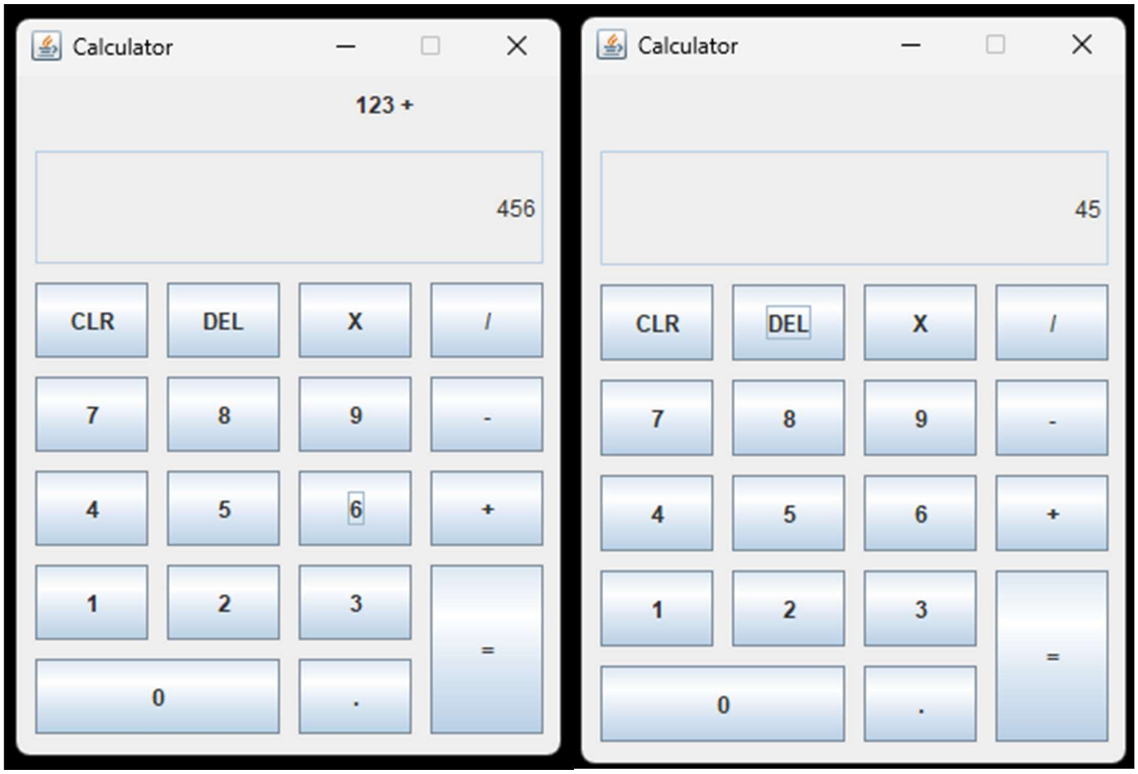


Figure 5: DEL Button deletes the label and the last received input

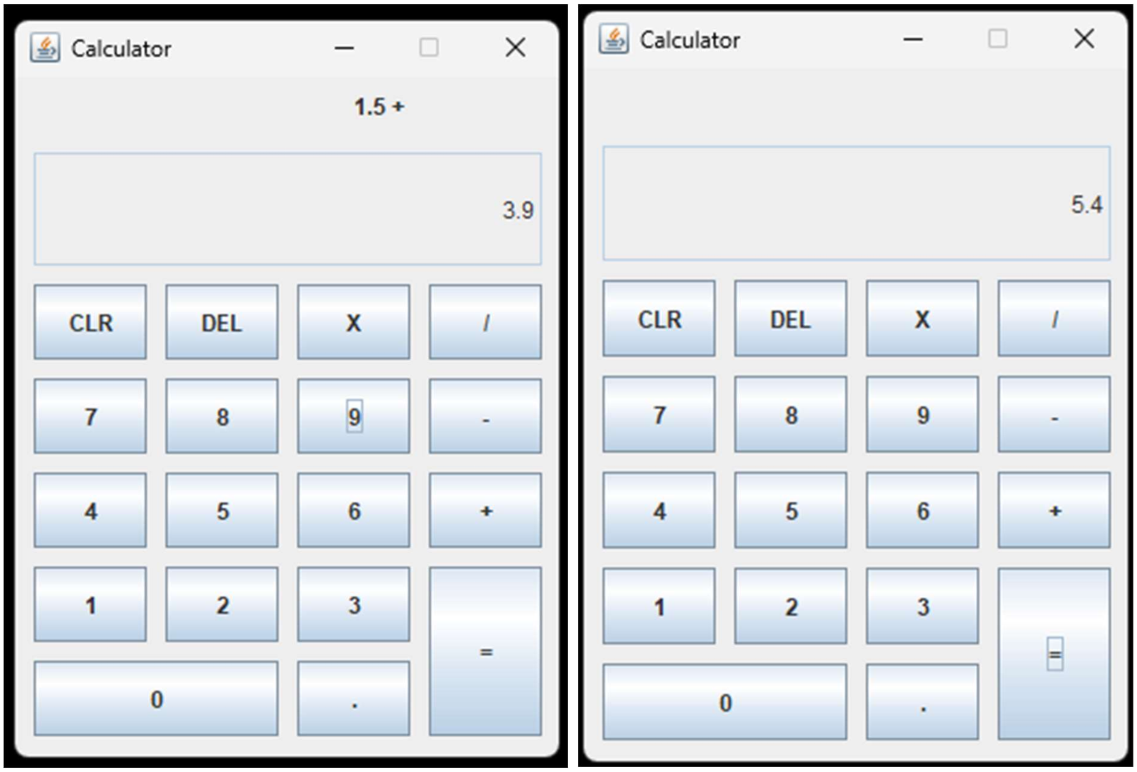
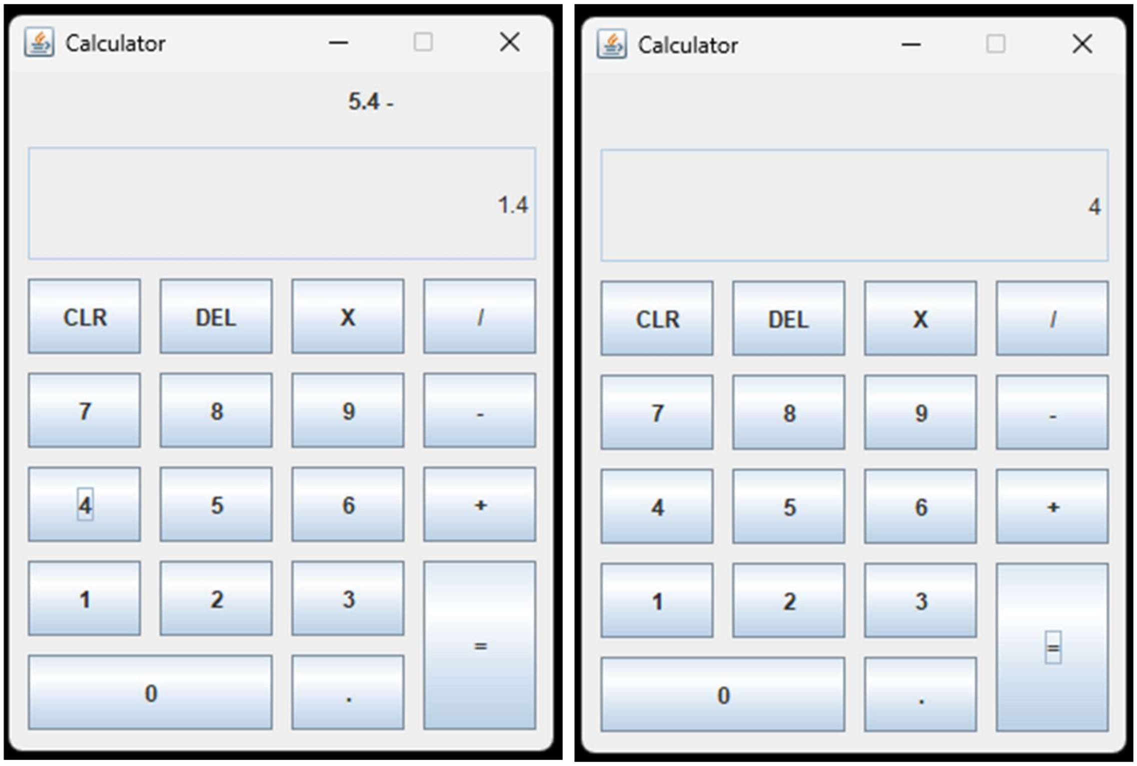


Figure 6 Performing Addition Operation between two Decimal values



```
5.4 -
1.4
```

```
4
```

Figure 7 Subtraction Operation between two Decimal values shows that if the result is in form of ".0" then it will only display the Integer part of the result

Conclusion:

The development of this calculator application using Java and Java Swing has successfully demonstrated the practical application of Java programming concepts and Swing GUI components. The project effectively integrates fundamental programming principles such as class creation, method implementation, and exception handling with more advanced topics like GUI design and event management. The resulting calculator performs basic arithmetic operations, including addition, subtraction, multiplication, and division, with a user-friendly interface that enhances usability.