ONCAMPUS
Achieve more.

Coursework Submission Sheet

Complete your details and read the information below. Please attach this sheet to your online submission to confirm that you understand and accept the **ON**CAMPUS academic regulations regarding coursework.

Student Full Name:	Nada Eraihane Merzoug
Student ID Number: (e.g. 12345)	27358
Date of Birth: (DD/MM/YYYY)	11/08/2004
Programme: (e.g. UFP)	IY1
Programme Start Date: (e.g. September 2020)	January 2024

Please confirm your understanding that:

1. At **ON**CAMPUS, we recognise that English may not be your first language. Our assessments are designed to allow you to achieve while continuing to develop your academic English skills. We expect all submitted work to be your own words (apart from in-text quotations), written in a style that reflects your English language level.

2. At any time, you may be asked to attend an interview with at least two members of academic staff to discuss the content of your submitted coursework. It is likely that one interviewer will be the Centre Head or Deputy Centre Head. Interviews may be requested as part of a random check, or if there is suspicion that an academic offence has been committed. We expect all students to be able to discuss and/or explain the main ideas and vocabulary used in submitted coursework.

3. It is your responsibility to read the programme handbook, including the policy on late submission, and information on available support if you are finding your studies difficult.

4. If you are unsure about any part of your coursework, you must ask your tutors for help before the submission deadline.

By attaching this sheet, you confirm that that you understand and accept all of the information above. You understand that all submitted coursework must include this coursework submission sheet, and that submissions without the sheet included will be rejected.

Table of Contents

Figures list:

I. Analysing Requirements:

1. The Purpose:

The purpose of this project is to develop a graphical user interface for a special calculator using Java. The GUI is supposed to covert between floating-point numbers and their binary representations that follows the IEEE 754 floating-point standard for double precision. This application is designed for accurate conversion and require a precision in floating-point arithmetic, such as those in MIPS architecture.

2. The Objectives:

a) Develop A User Friendly Gui: create a graphical interface that is easy to navigate.

b) Accurate Conversion: have a reliable code that is able to convert decimal floating-points to their binary representation and vice versa using the IEEE 754 standard

c) Precision: double precision is required for a more accurate result each time

d) Implement Essential Calculator Features: have different input buttons like numbers, and input fields and operational buttons.

e) Enable User Input Features: the user can input either using the keyboard or the buttons prese t in the GUI

f) Display Conversion Results: show the conversion results in real life for the user to view them and verify their input.

g) Have A Code That Follows Best Practice: comments, clean clear code, use object-oriented programming principles.

3. The Significance of The Project:

This project is an example of applications of object-oriented programming in solving real-world problems, it also helps understand how floating-point conversion using IEEE 754 standards work. The calculator is helpful to anyone that needs precision in binary to decimal conversion and vice versa. This project also involves different stages of software development(requirement analysis, design, implementation, testing and documentation) which a good way to understand software development effectively.

II. The Design Specification for The Application:

1. The Constructs & GUI Design:

a) Inputs:

Decimal field: this field will allow the user to input the decimal value they want to make into a binary number, the field allows the input of both decimal integers and floating points. The JTextField from swing is used to create it and it is places at (50, 20) and its size is (390, 50).

Binary field: this field will allow the user to input the binary value they want to make into a decimal number, the field allows the input of both binary integers and floating points using IEEEE 754 standards. The JTextField from swing is used to create it and it is places at (50, 80) and its size is (390, 50).

Two fields are used to make it easier to know what the decimal value and binary values are at the same time and avoid any confusion. For example, if 10 is entered in the decimal field it is processed as a decimal and not confused for a binary value by both the computer and the user.

Buttons: there are different buttons in this Gui that perform specific functions like conversion or clearing the input output fields or moving right or left or deletion entering numbers displaying sign bit or deciding the decimal places. The buttons are placed this way in the 1st row there is the “to decimal”, ”to binary”, ”clear” buttons with the first two being double width. In the 2nd row there is the “7”, ”8”, ”9”, ”decimal places” buttons with the last one being double width. In the 3rd row there is “4”, ”5”, ”6”, ”left”, ”right” buttons with all of them being regular size. The 4th row there is “1”, ”2”, ”3”, ”enter ” buttons with the last one being double width. In the 5th and last row there is “0”, ”.”, ”-”, ”S”, ”delete” buttons with all of them being regular size. The buttons are place at (50, 150) and between each row and column the spacing is (10) the size of a regular button is 70 for width and 50 for height and for the double width buttons it is 70*2+10 which make them have 150 for width and keep 50 as the height.

b) Processes:

Buttons action:

Clear: this button clears both input fields (binary and decimal)

To binary: this button makes decimal values entered in the decimal field into binary numbers

To decimal: this button makes binary values entered in the binary field into decimal numbers

Enter: in the code implemented this button doesn’t have a function since the “to binary”, ”to decimal” buttons are sufficient to do the actions demanded.

Decimal places: this button askes the user how many decimal places for the floating-point conversion. In default it is set to 6 places for double precision (3 is singular precision) and it can be increased or decreased

Left: this button moves the cursor to the left in the active text field for editing

Right: this button moves the cursor to the right in the active text field for editing

S: this button displays the sign bit of the binary number, if it is 0 it will display 0 (+ve) and if it is 1 it will display 1 (-ve).

Delete: this button deletes the character that is at the current cursor position in the active field

Numbers: the number button when pressed enter a number in the active text field.

-: the minus sign button is used to enter negative values in the decimal text filed

.: the point button is used to enter floats in the decimal text field.

Focus handling: this is used to manage the currently active text field. It sets the active field to the one that gain focus and moves the cursor to the end of the field for extra input using the focusGained(FocusEvent e) method. And using the focusLost(FocusEvent e) methos it loses focus and doesn’t use the currently unused text field.

Conversion methods: in this calculator there are two conversion methods. The first one is the convertToBinary() method, this one converts the decimal values entered in the decimal text field into binary string using IEEE 754 format for floating points numbers. It does it by checking if the input contains a point to determine if it is a float, then uses the Double.parseDouble for floats or uses Long.parseLong for integers, the value calculated is then outputted in the binary field. On the other hand, the convertToDecimal() method is used to convert binary values in the binary text field to a decimal value. First it checks if the number of bits entered is 64 if so it will consider it as a float and compute it as such. It uses Long.parseLong for integers and binaryStringToDouble for float points conversion. The result of the conversion will be outputted in the decimal text field.

Utility methods: these are methods called when one of the button is pressed. First of all, the doubleToBinaryString(double value) method is used to convert a double to an IEEE 754 binary string. While binaryStringToDouble(String binary) is there to converts an IEEE 754 binary string to a double. In addition, the setDecimalPlaces() method prompts the user to set the number of decimal places for floating-point conversion. Beside the moveCursorLeft() moves the cursor one position to the left in the active text field. On the other hand, the moveCursorRight() method aims to move the cursor one position to the right in the active text field. Also, the displaySignBit() is used to displays the sign bit of the binary number. At last, the deleteCharacter() method deletes the character at the current cursor position in the active text field.

c) Outputs:

Binary field output: it displays the binary representation of the decimal input when the button “to binary” is pressed.

Decimal field output: it displays the decimal representation of the binary input when the button “to decimal ” is pressed.

Sign bit display: this message box displays the sign bit of the binary number when s is pressed 0(+ve) for positive values and 1(-ve) for negative ones.

Errors: there are error messages that are outputted if the values entered are not right like entering a decimal value in the binary field or entering letters instead of numbers…

2. The Storyboard:

The GUI consists of a special calculator that converts floating points numbers into binary string and vice versa using IEEE 754 floating-point standard for double precision. It is created using Java and has a user-friendly graphic user interface the makes the conversion process easier.

The main frame is named “IEEE 754 converter” it is (510x500px) in size and the background is light Gray. There are two input fields the decimal one and the binary one. The decimal text field is a JTextField positioned at (50, 20) and has the size (390, 50). The font is set to Arial plain and the font size is 12pt. this field is for the user to enter decimal values or to output the converted decimal values. On the other hand, the binary text field has the same type (JTextField) and is positioned at (50, 80) and its size is (390, 50). The font is set at Arial plain, 12pt. and this field is used to input binary strings by user or to output the result of the conversion.

The buttons are arranged this way:

Row 1:

to decimal: (50, 150) – Orange (double-sized)

to binary: (180, 150) – Orange (double-sized)

clear: (310, 150) - Magenta

Row 2:

7: (50, 210) - Blue

8: (120, 210) - Blue

9: (190, 210) - Blue

decimal places: (260, 210) - Gray (double-sized)

Row 3:

4: (50, 270) - Blue

5: (120, 270) - Blue

6: (190, 270) - Blue

left: (260, 270) - Gray

right: (340, 270) - Gray

Row 4:

1: (50, 330) - Blue

2: (120, 330) - Blue

3: (190, 330) - Blue

enter: (260, 330) - Gray (double-sized)

Row 5:

0: (50, 390) - Blue

.: (120, 390) - Blue

-: (190, 390) - Blue

s: (260, 390) - Gray

delete: (340, 390) – Gray

Regular size is 70 for width and 50 for height and double button is 70*2+10 which is 150 for width and 50 for height.

The “to decimal” button converts binary input to decimal and displays in the decimal field.

While the “to binary” is there to convert decimal input to binary and displays in the binary field. The “Clear” clears both the decimal and binary fields. And the “0-9, ., -“: allow users to input numbers and floats and a negative sign into the active field. The “Decimal places” button prompts the user to set the number of decimal places for floating-point conversion in default it is set to 6 places. Also, the “left”, and “right” buttons move the cursor left or right in the active field. Additionally, the “s” one displays the sign bit of the binary number.
Finally, the “delete” button is used to Delete the character at the current cursor position in the active field.

<u>The results expected:</u>

This Gui is expected to convert decimal floats into a binary string using IEEE 754 floating-point double precision, for example when 77.4 is entered in the decimal text field it is expected to obtain 0100000001010011010110011001100110011001100110011001100110011010 in the binary text field and in this case when the “s” button is pressed it is expected to get 0(+ve). On the other hand, if -35.9 is entered in the decimal text field again it is the Gui is supposed to output 1100000001000001111100110011001100110011001100110011001100110011 and when “S” button is pressed it should output 1(-ve). Additionally, if the scenario is reversed and 0100000001010011010110011001100110011001100110011001100110011010 is entered in the binary text field and the “to decimal ” button is pressed in the decimal field 77.4 should appear. And same thing for 1100000001000001111100110011001100110011001100110011001100110011, 35.9 should appear.

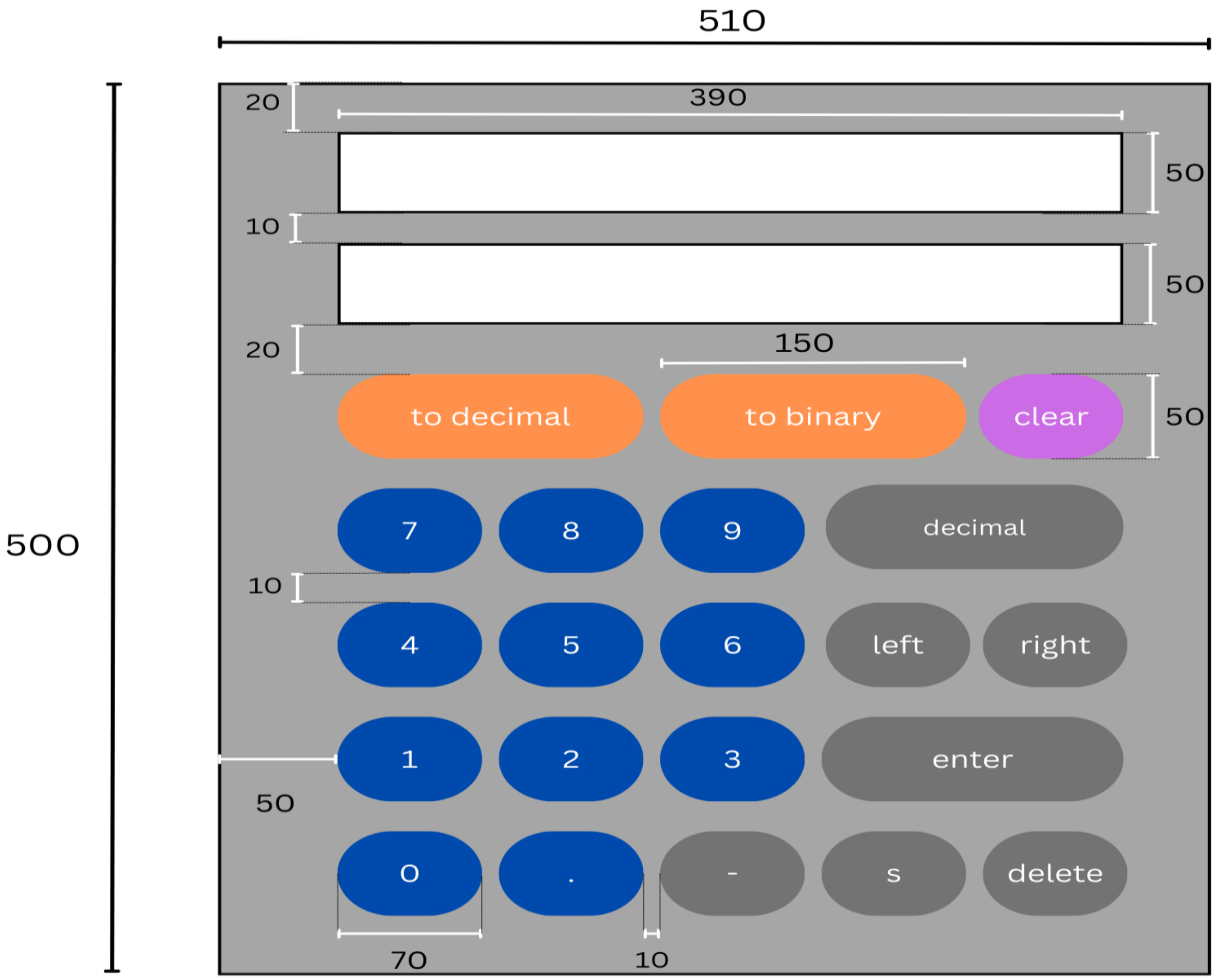


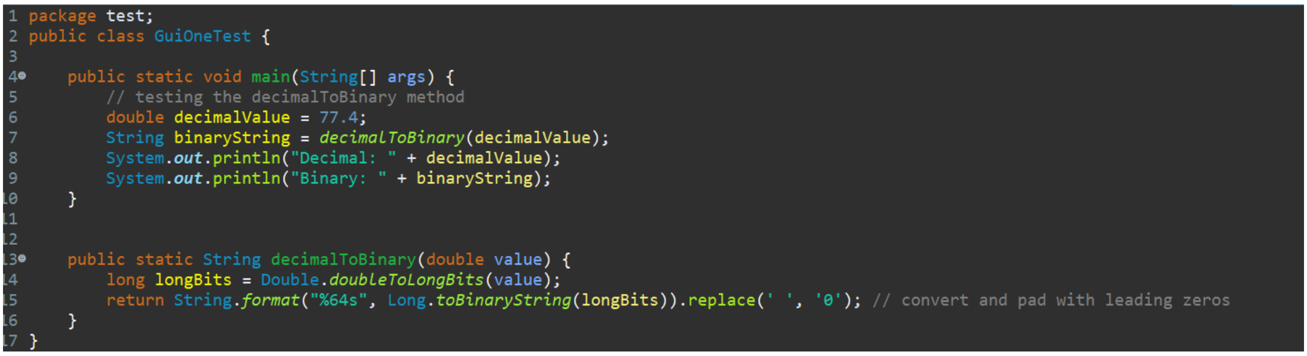
Figure 1 storyboard of the GUI with dimensions

III. Testing & Implementation:

1. Testing Procedures Observed:

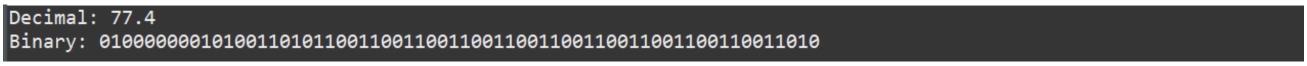
a) Unit Testing:

Testing the decimalToBinary() and binaryToDecimal() methods as separate, self-contained pieces of code



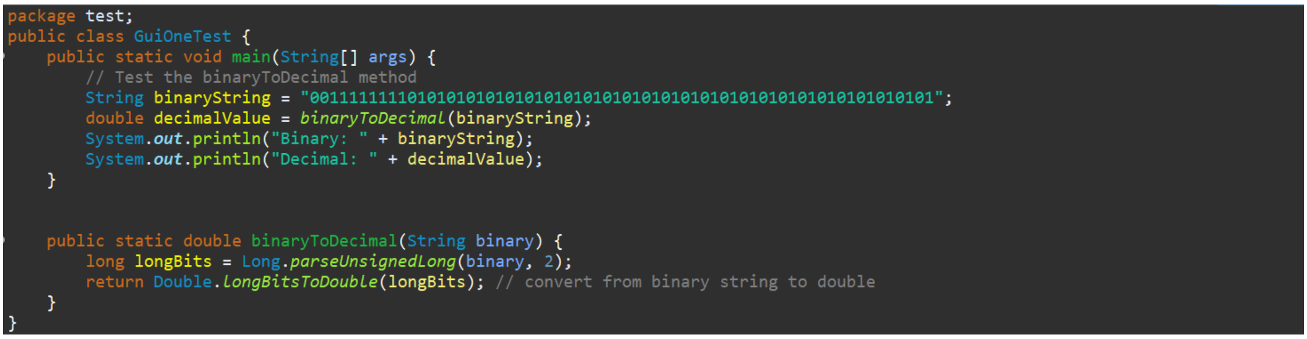
```java
package test;
public class GuiOneTest {

    public static void main(String[] args) {
        // testing the decimalToBinary method
        double decimalValue = 77.4;
        String binaryString = decimalToBinary(decimalValue);
        System.out.println("Decimal: " + decimalValue);
        System.out.println("Binary: " + binaryString);
    }


    public static String decimalToBinary(double value) {
        long longBits = Double.doubleToLongBits(value);
        return String.format("%64s", Long.toBinaryString(longBits)).replace(' ', '0'); // convert and pad with leading zeros
    }
}
```

Figure 2 decimal to binary code testing



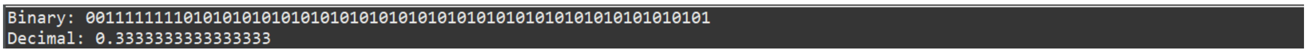
```
Decimal: 77.4
Binary: 0100000001010011010110011001100110011001100110011001100110011010
```

Figure 3 decimal to binary results



```java
package test;
public class GuiOneTest {
    public static void main(String[] args) {
        // Test the binaryToDecimal method
        String binaryString = "0011111111010101010101010101010101010101010101010101010101010101";
        double decimalValue = binaryToDecimal(binaryString);
        System.out.println("Binary: " + binaryString);
        System.out.println("Decimal: " + decimalValue);
    }


    public static double binaryToDecimal(String binary) {
        long longBits = Long.parseUnsignedLong(binary, 2);
        return Double.longBitsToDouble(longBits); // convert from binary string to double
    }
}
```

Figure 4 binary to decimal code testing



```
Binary: 0011111111010101010101010101010101010101010101010101010101010101
Decimal: 0.3333333333333333
```

Figure 5binary to decimal results

b) System Testing:

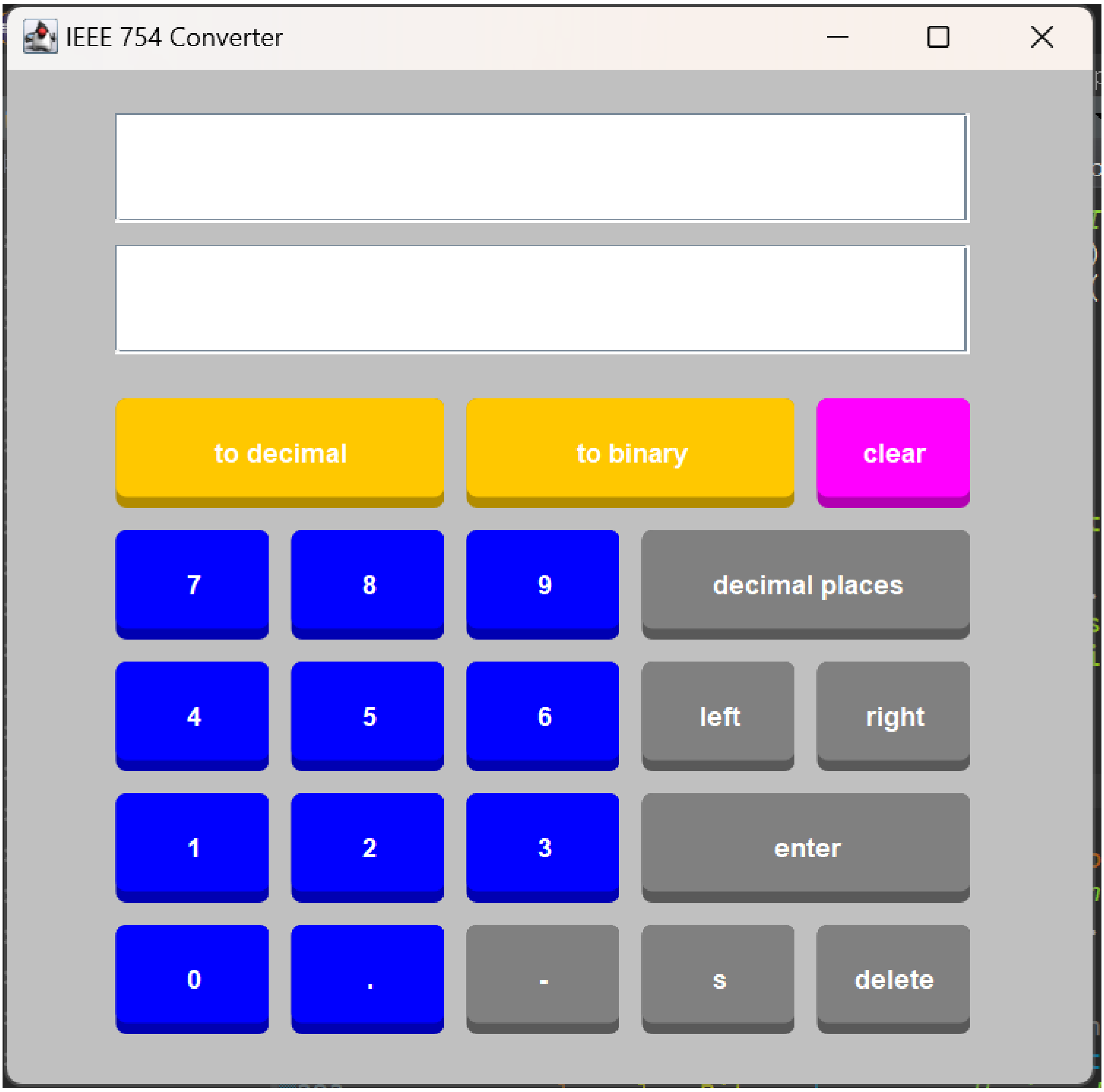
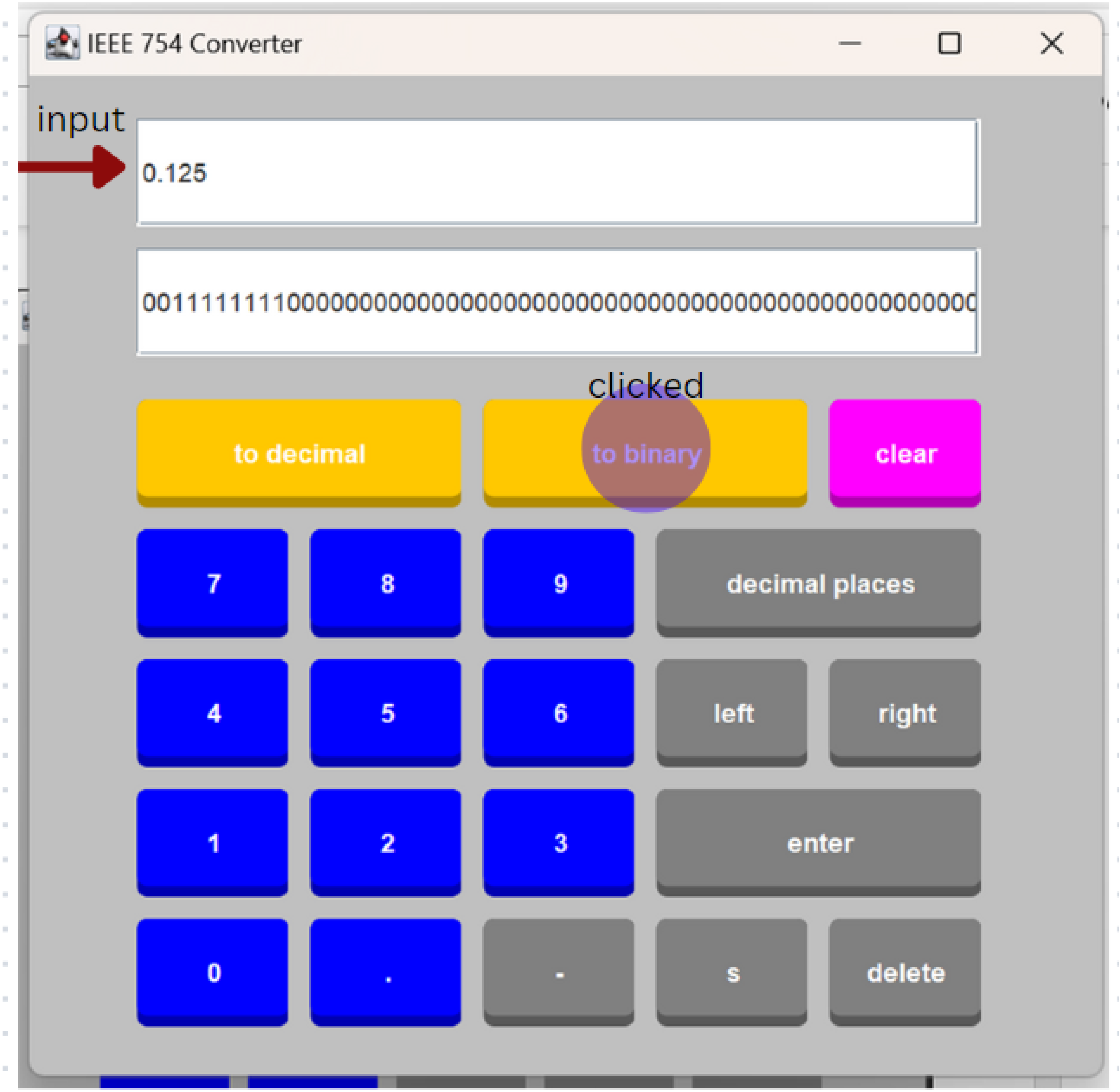


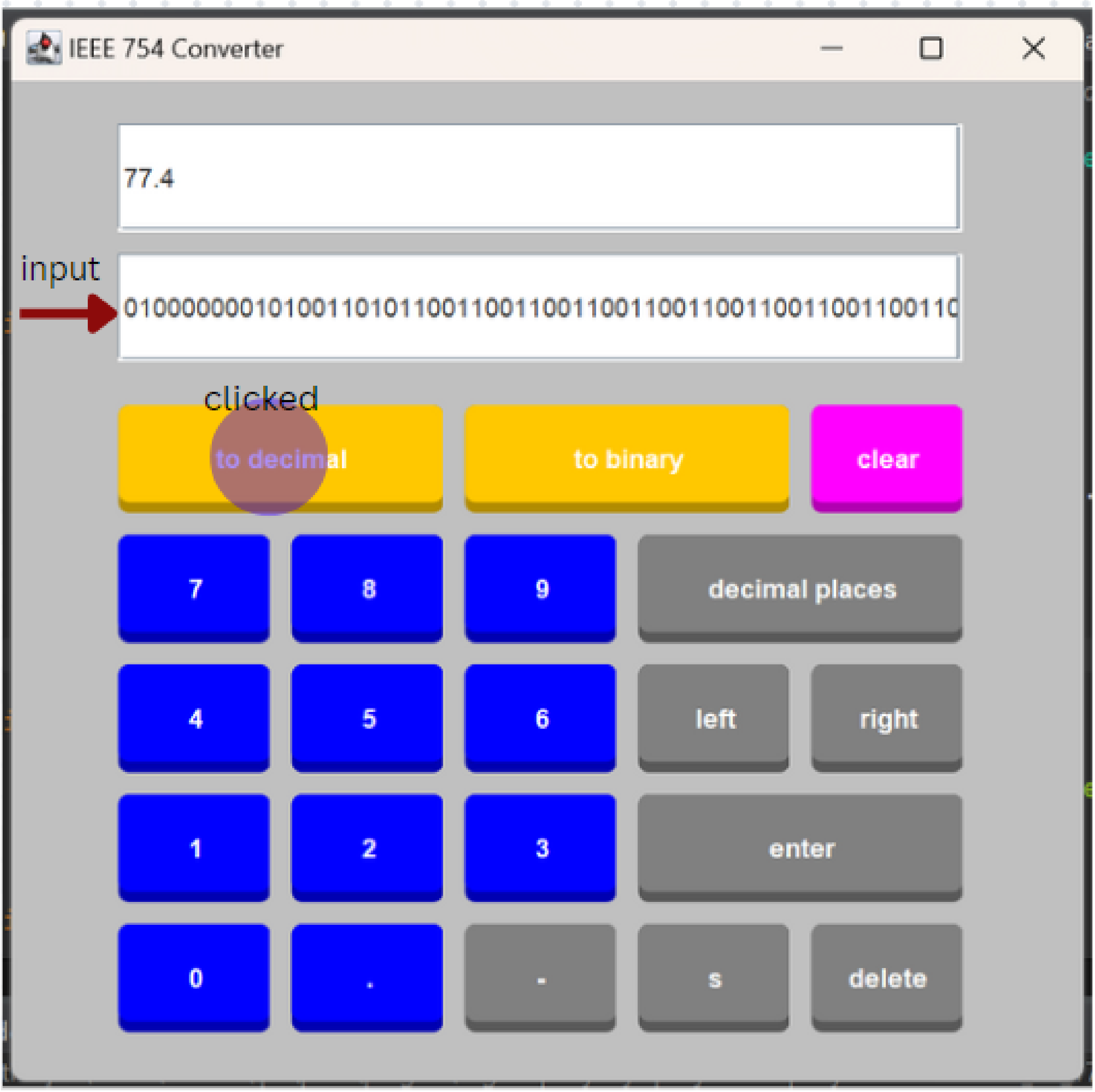
Figure 6 GUI look



0.125

001111111100000000000000000000000000000000000000000000000000

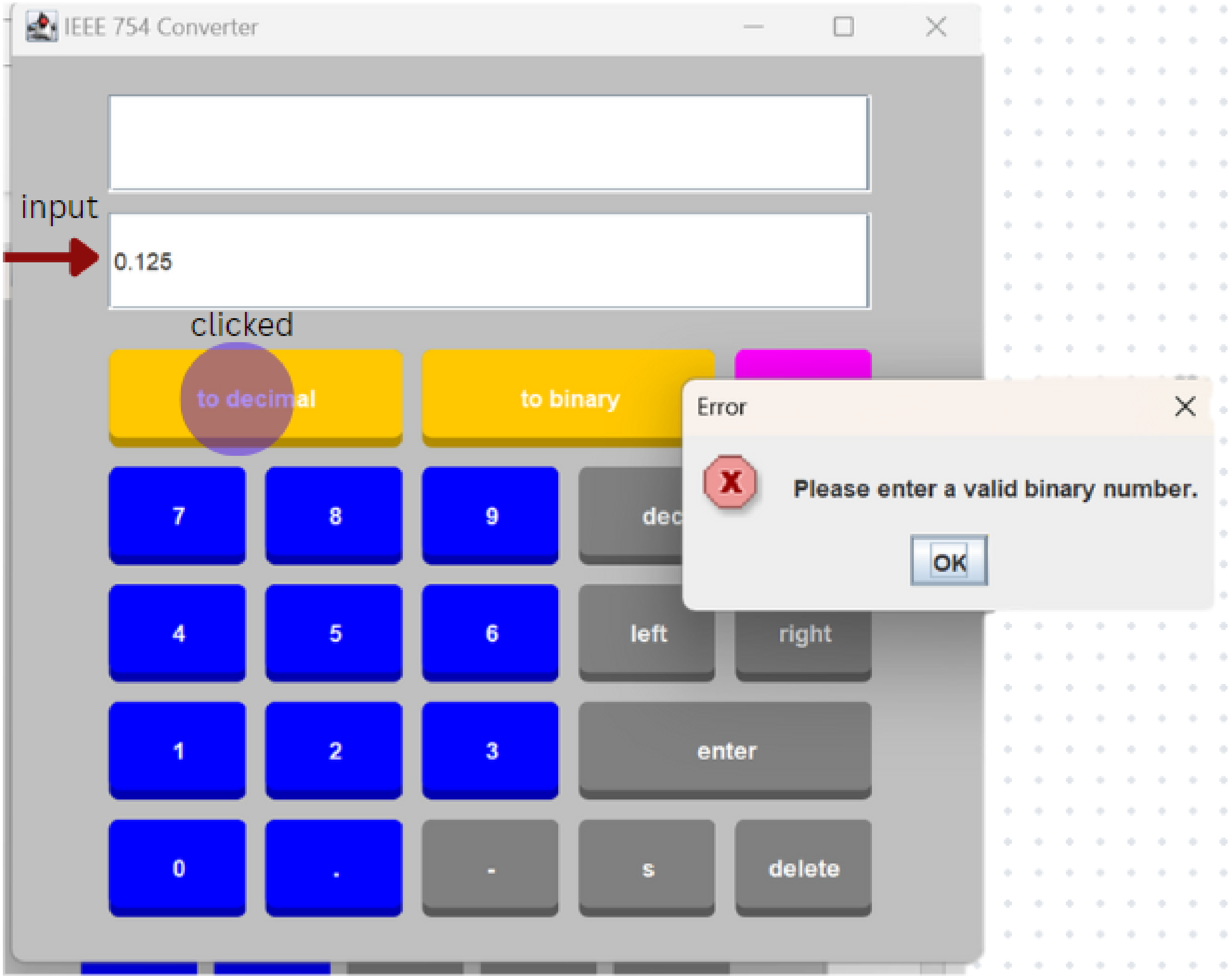
Figure 7 decimal float to binary string conversion



77.4

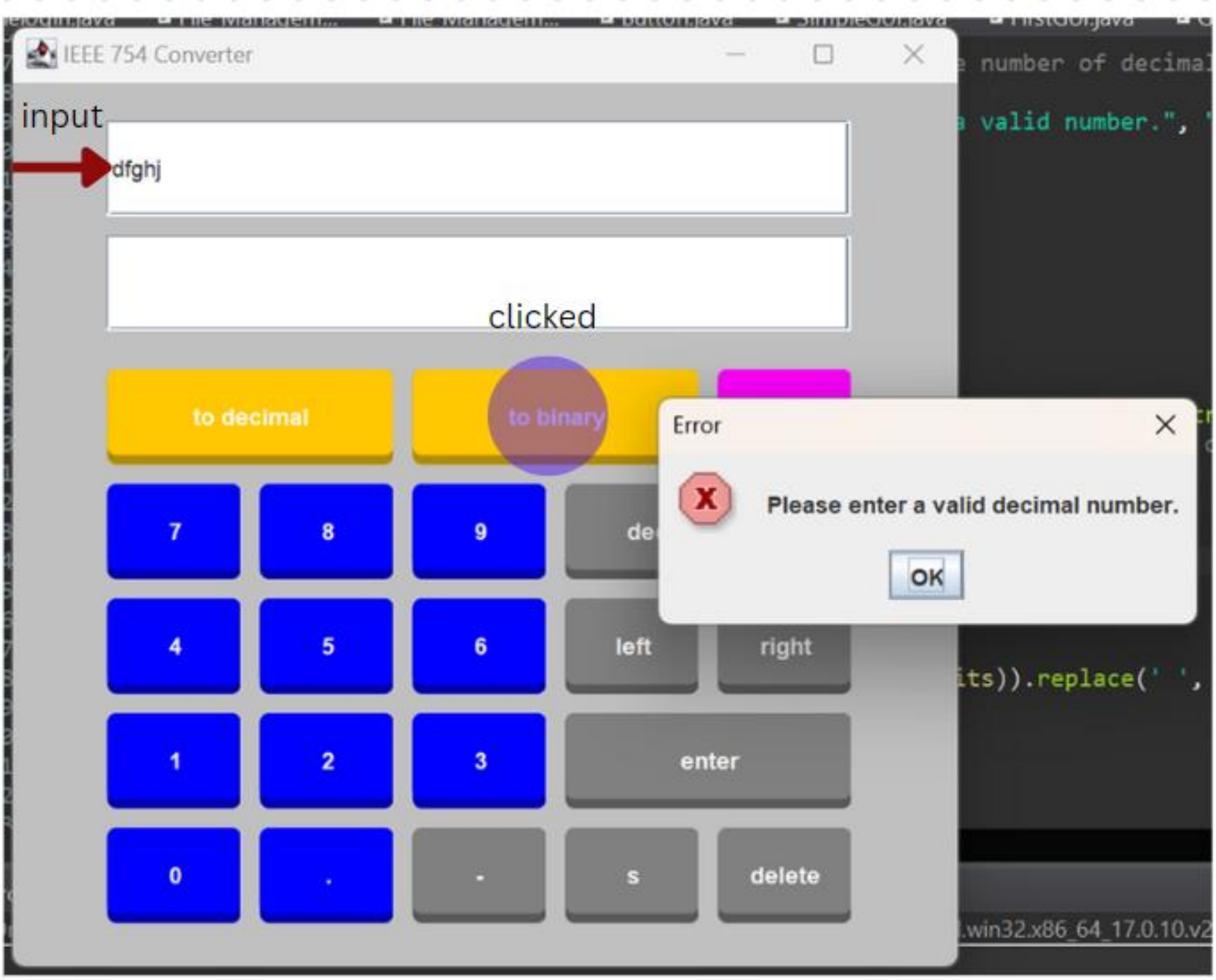
01000000010100110101100110011001100110011001100110011001100110011001100

Figure 8 conversion of a decimal float from a binary string



Please enter a valid binary number.

Figure 9 error message if binary input is incorrect



IEEE 754 Converter

dfghj

Error

Please enter a valid decimal number.

Figure 10 error message if the decimal input is wrong

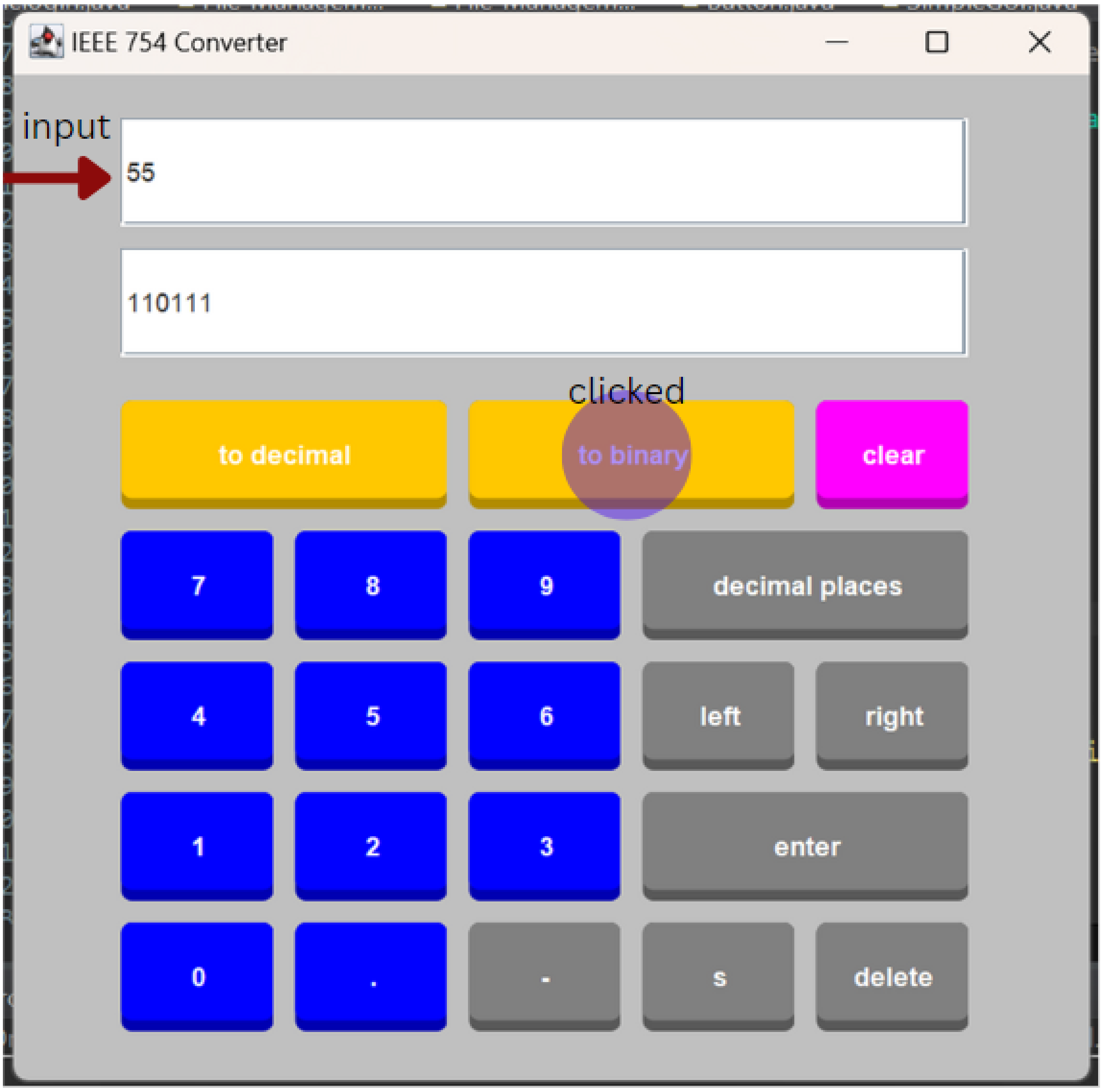
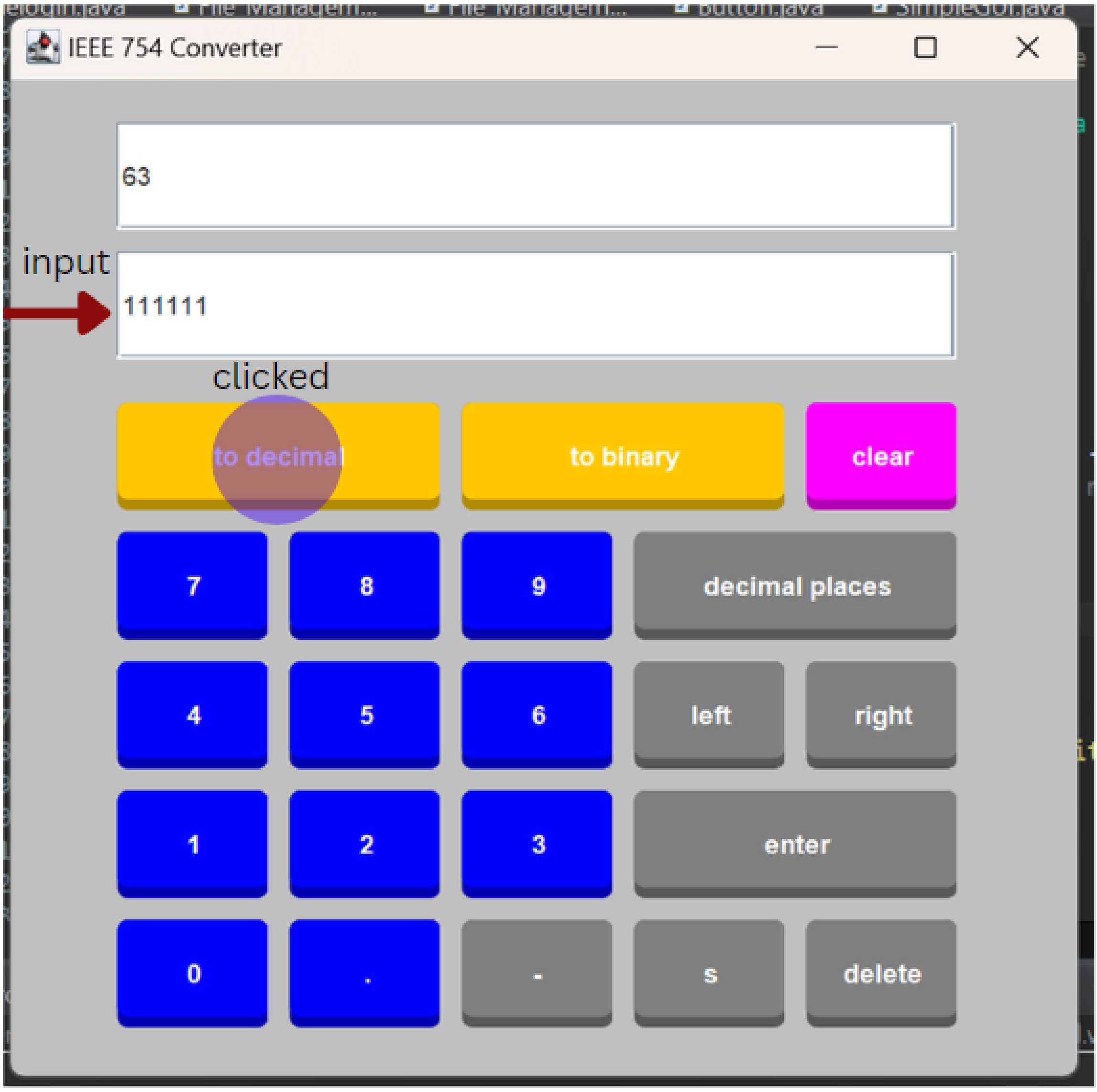


Figure 11 conversion of a decimal integer into a binary float

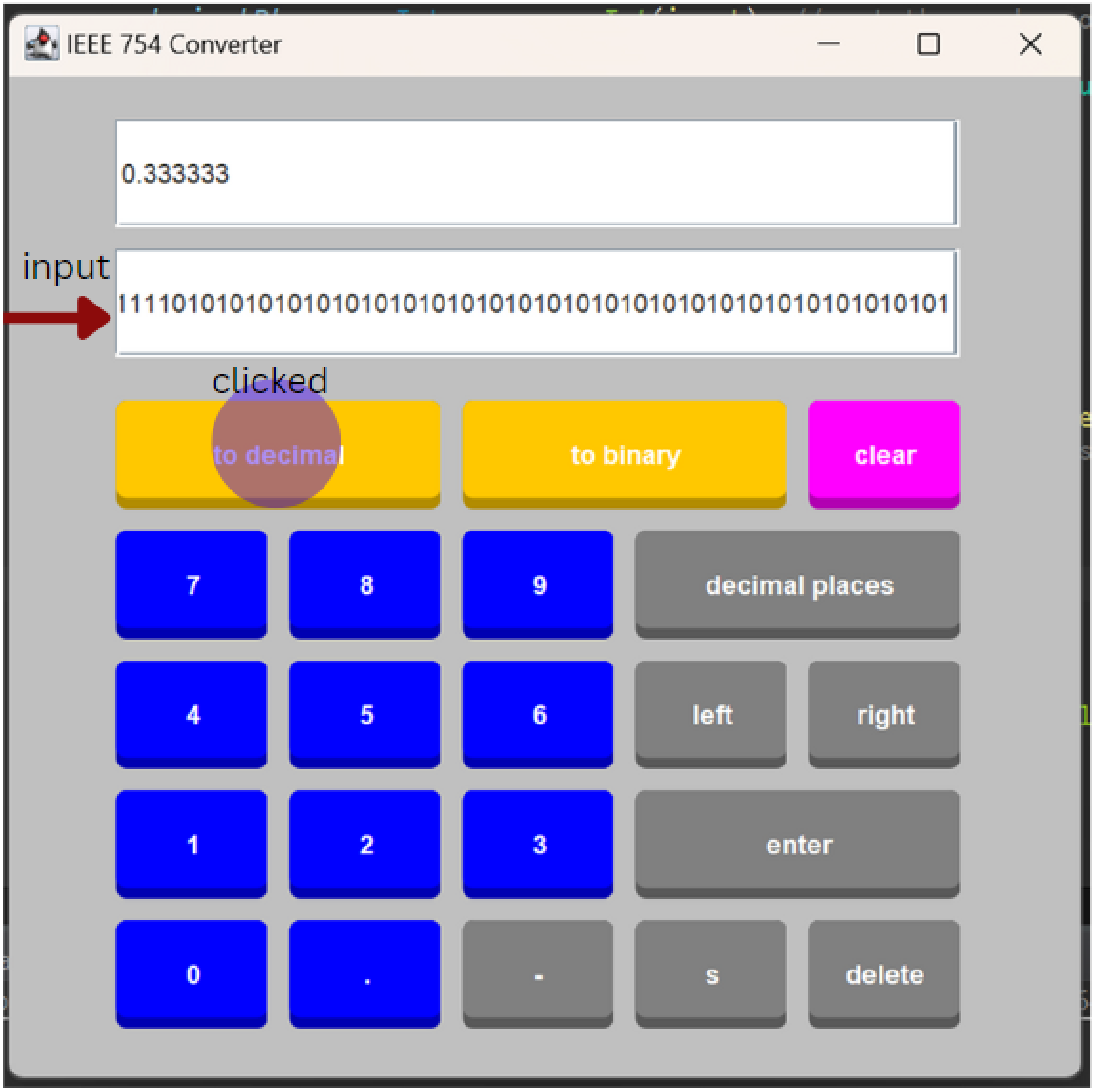


IEEE 754 Converter

63

111111

Figure 12 conversion of a binary string into a decimal integer

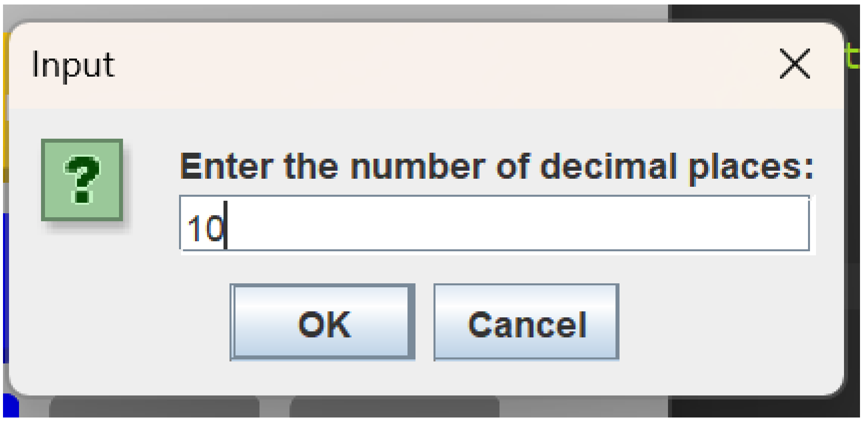


IEEE 754 Converter

0.333333

1111010101010101010101010101010101010101010101010101010101

Figure 13 conversion of a binary string into a decimal float with default decimal places

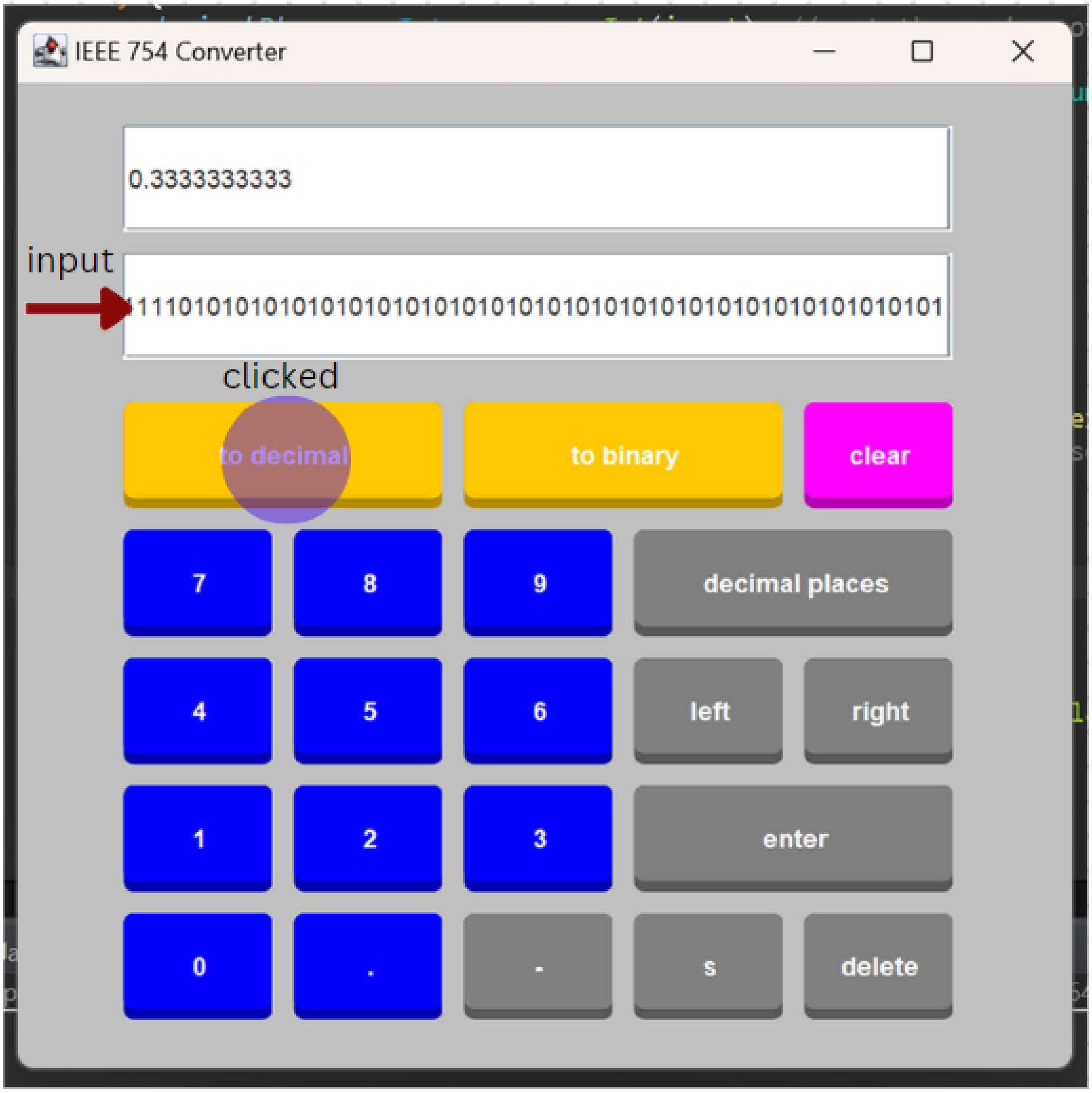


Input

Enter the number of decimal places:

10

Figure 14 setting a different number of decimal places

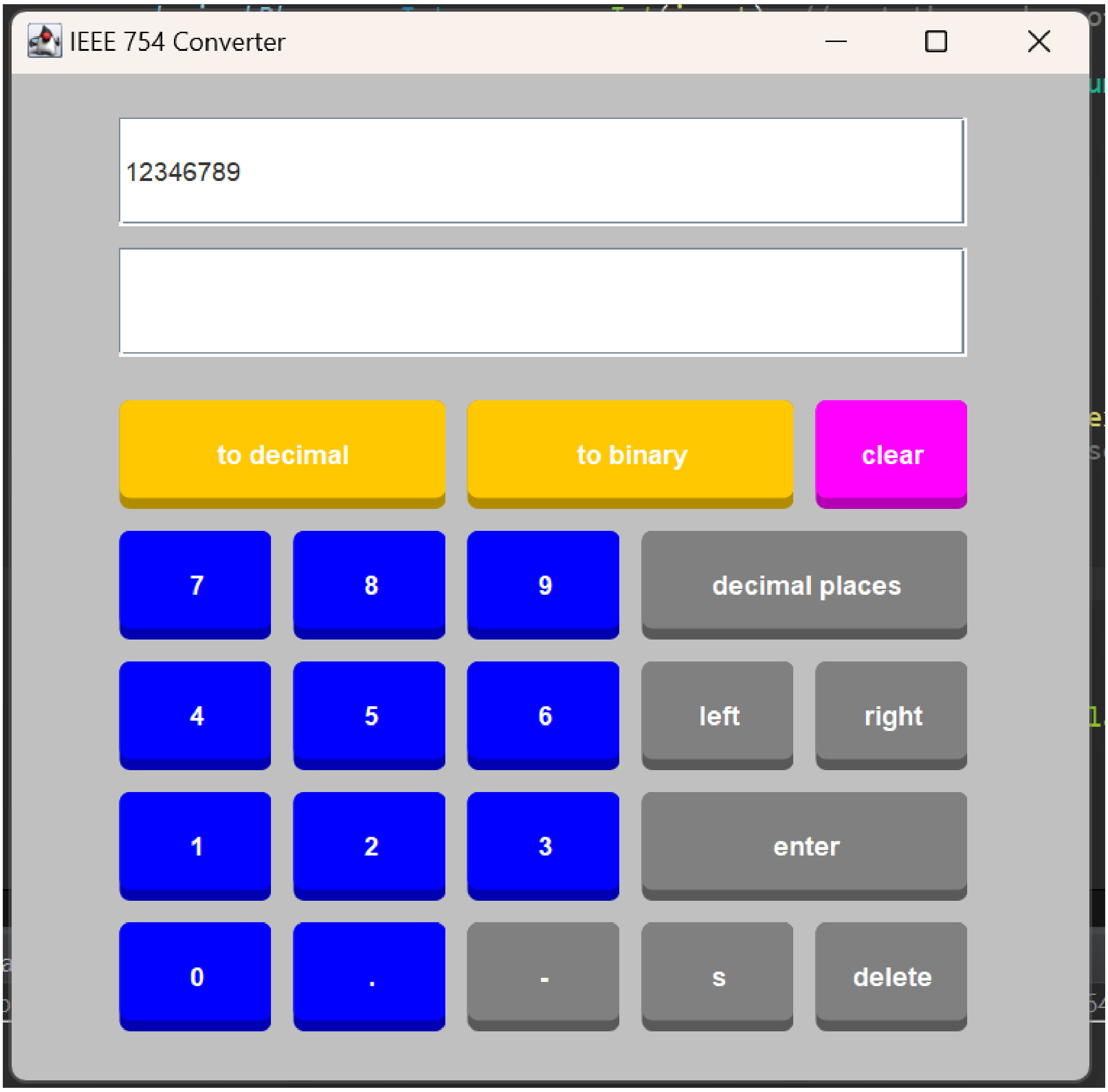


IEEE 754 Converter

0.3333333333

111010101010101010101010101010101010101010101010101010101

Figure 15 conversion of a binary string into a decimal float with 10 decimal places



12346789

Figure 16 wrong input before editing

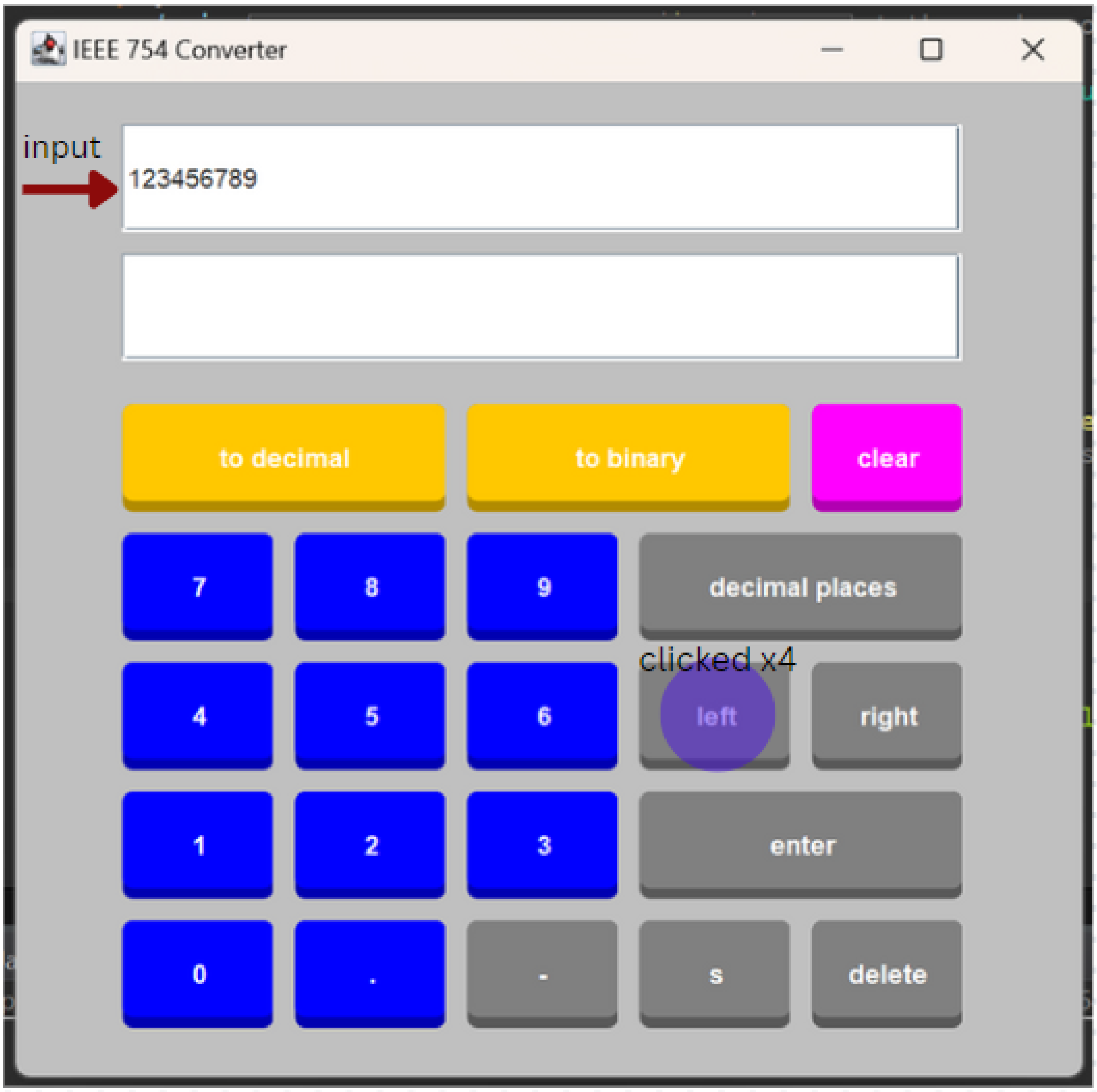
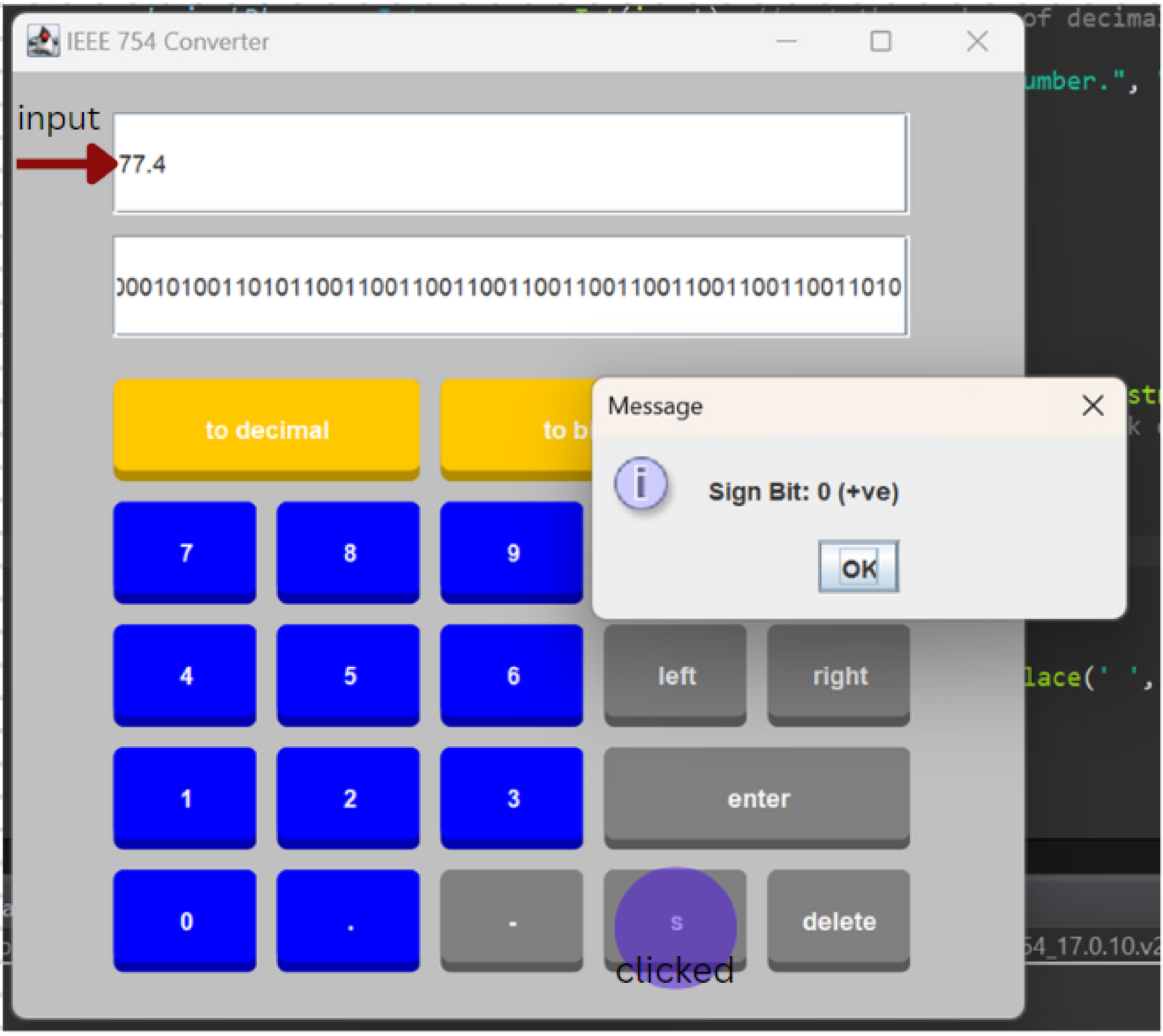


Figure 17 input after editing



input

77.4

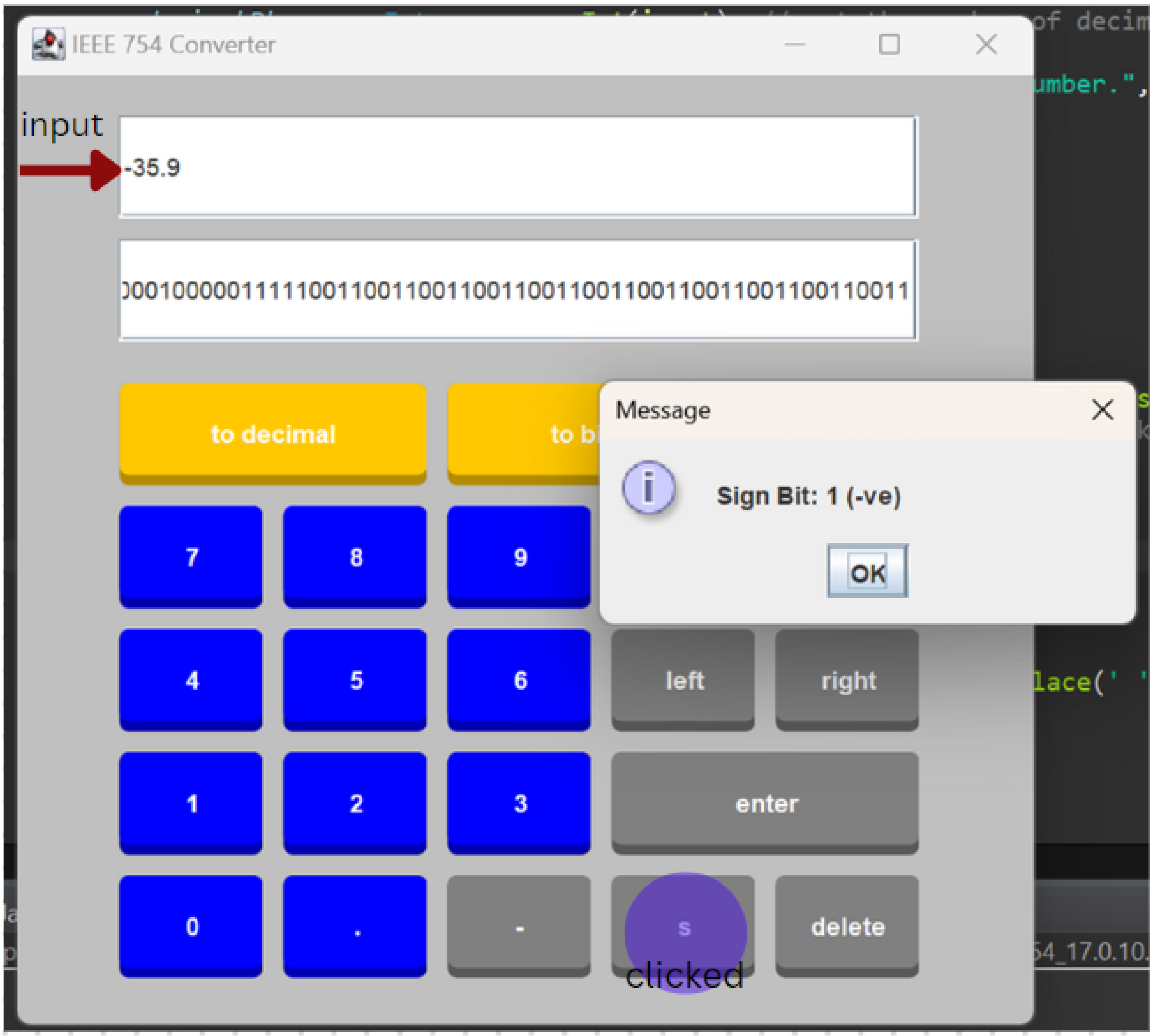
00101001101011001100110011001100110011001100110011001100110011010

Message

Sign Bit: 0 (+ve)

OK

Figure 18 output of the sign bit button when input is positive



-35.9

)0010000011111001100110011001100110011001100110011001100110011

Message

Sign Bit: 1 (-ve)

OK

Figure 19 output of the sign bit button when input is negative

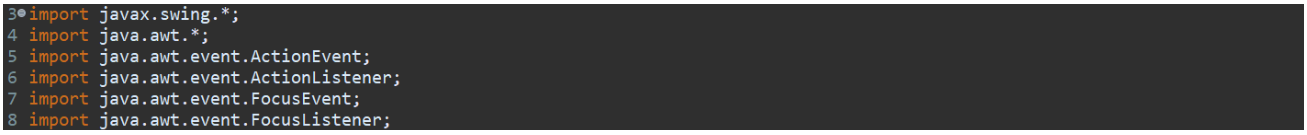
2. The Code Analyses with Screenshots of Results:

First of all, a new java project is created called GuiOne in this case and a package is created named gui.

```
package gui;
```

Figure 20 package creation

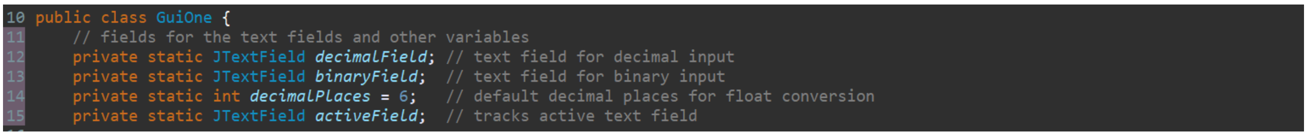
Then all the libraries needed are imported: swing for buttons, text fields, dialog boxes... awt is for fonts, colours, geometrical control of objects, and for rendering algorithms. Awt.event.ActionEvent is to define when an action occurred like clicking a button, awt.event.ActionListener is an interface that receives action events, When the action event occurs, that object's actionPerformed method is invoked. awt.event.FocusEvent is a class that represents a focus event. And awt.event.FocusListener is an interface for receiving keyboard focus events.



```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
```

Figure 21 importing the libraries

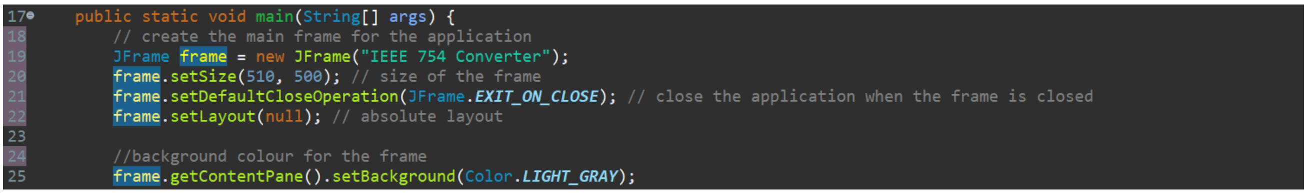
Using JTextField for both decimal text field and the binary one and setting the decimal places at 6 as a default.



```java
public class GuiOne {
    // fields for the text fields and other variables
    private static JTextField decimalField; // text field for decimal input
    private static JTextField binaryField;  // text field for binary input
    private static int decimalPlaces = 6;   // default decimal places for float conversion
    private static JTextField activeField;  // tracks active text field
```

Figure 22 making text fields and setting the default decimal places

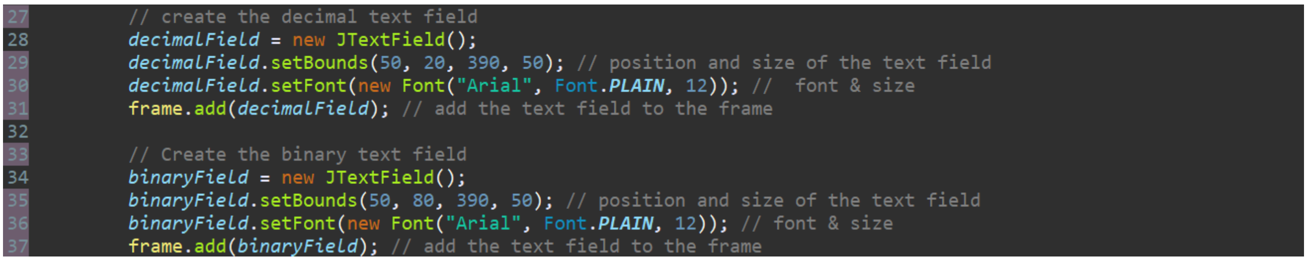
Creating the main frame and naming it, defining the size, and what happens when it is closed. Also giving the background a set colour.



```java
    public static void main(String[] args) {
        // create the main frame for the application
        JFrame frame = new JFrame("IEEE 754 Converter");
        frame.setSize(510, 500); // size of the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // close the application when the frame is closed
        frame.setLayout(null); // absolute layout

        //background colour for the frame
        frame.getContentPane().setBackground(Color.LIGHT_GRAY);
```

Figure 23 creation of the main frame

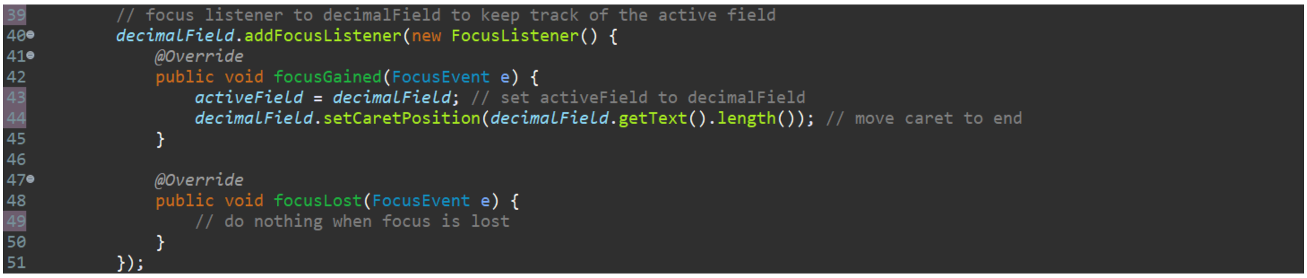
Creating the decimal field and binary one and placing them in the frame and setting their size and font information.



```java
        // create the decimal text field
        decimalField = new JTextField();
        decimalField.setBounds(50, 20, 390, 50); // position and size of the text field
        decimalField.setFont(new Font("Arial", Font.PLAIN, 12)); //  font & size
        frame.add(decimalField); // add the text field to the frame

        // Create the binary text field
        binaryField = new JTextField();
        binaryField.setBounds(50, 80, 390, 50); // position and size of the text field
        binaryField.setFont(new Font("Arial", Font.PLAIN, 12)); // font & size
        frame.add(binaryField); // add the text field to the frame
```

Figure 24 creation of text fields for decimal and binary input and output

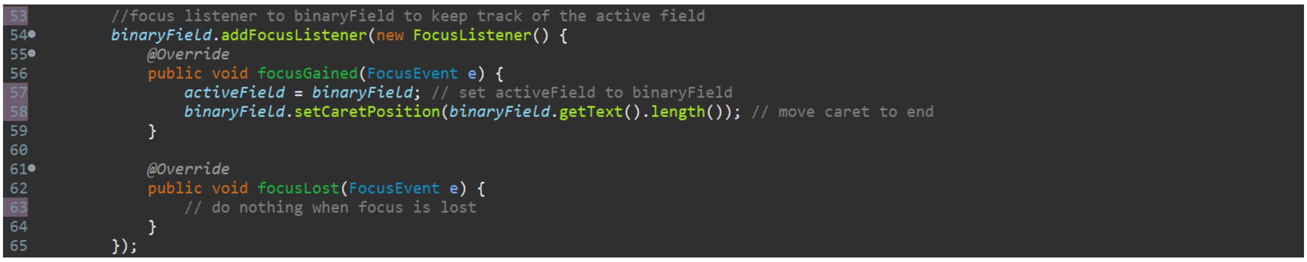
Setting a focus listener to the decimal text field using focusGained(FocusEvent e) method and focusLost(FocusEvent e)



```java
        // focus listener to decimalField to keep track of the active field
        decimalField.addFocusListener(new FocusListener() {
            @Override
            public void focusGained(FocusEvent e) {
                activeField = decimalField; // set activeField to decimalField
                decimalField.setCaretPosition(decimalField.getText().length()); // move caret to end
            }

            @Override
            public void focusLost(FocusEvent e) {
                // do nothing when focus is lost
            }
        });
```

Figure 25 setting a focus listener for the decimal text field

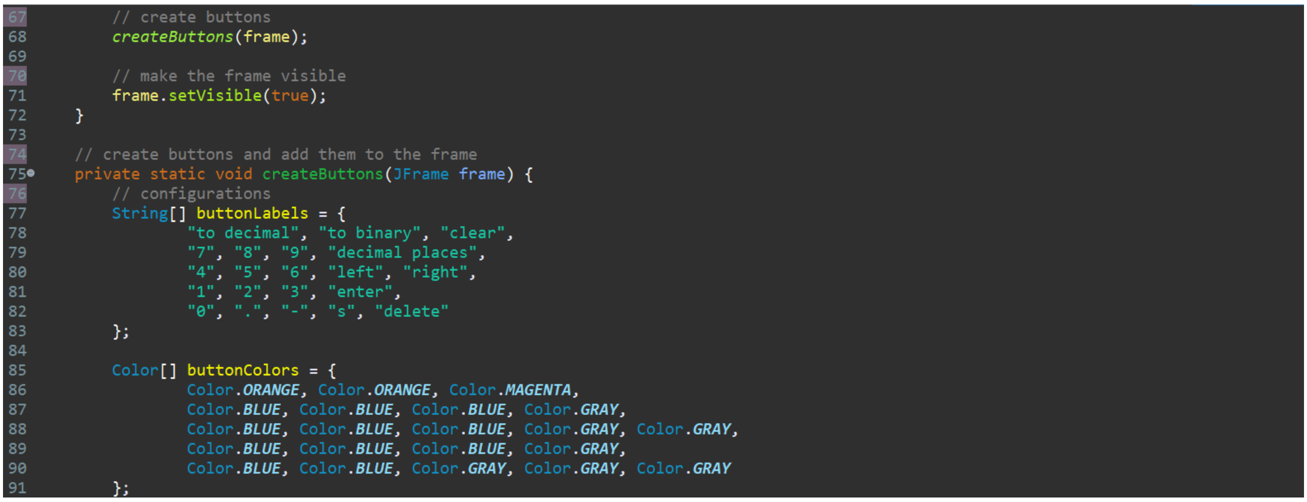
Setting a focus listener to the binary text field using focusGained(FocusEvent e) method and focusLost(FocusEvent e)



```java
        //focus listener to binaryField to keep track of the active field
        binaryField.addFocusListener(new FocusListener() {
            @Override
            public void focusGained(FocusEvent e) {
                activeField = binaryField; // set activeField to binaryField
                binaryField.setCaretPosition(binaryField.getText().length()); // move caret to end
            }

            @Override
            public void focusLost(FocusEvent e) {
                // do nothing when focus is lost
            }
        });
```

Figure 26 setting a focus listener for the binary text field

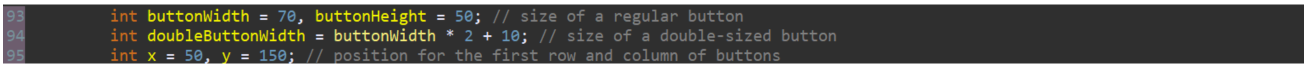
Creating the buttons in the frame, naming them, and setting their colours.



```java
        // create buttons
        createButtons(frame);

        // make the frame visible
        frame.setVisible(true);
    }

    // create buttons and add them to the frame
    private static void createButtons(JFrame frame) {
        // configurations
        String[] buttonLabels = {
                "to decimal", "to binary", "clear",
                "7", "8", "9", "decimal places",
                "4", "5", "6", "left", "right",
                "1", "2", "3", "enter",
                "0", ".", "-", "s", "delete"
        };

        Color[] buttonColors = {
                Color.ORANGE, Color.ORANGE, Color.MAGENTA,
                Color.BLUE, Color.BLUE, Color.BLUE, Color.GRAY,
                Color.BLUE, Color.BLUE, Color.BLUE, Color.GRAY, Color.GRAY,
                Color.BLUE, Color.BLUE, Color.BLUE, Color.GRAY,
                Color.BLUE, Color.BLUE, Color.GRAY, Color.GRAY, Color.GRAY
        };
```

Figure 27 creating buttons and naming them and defining their colours

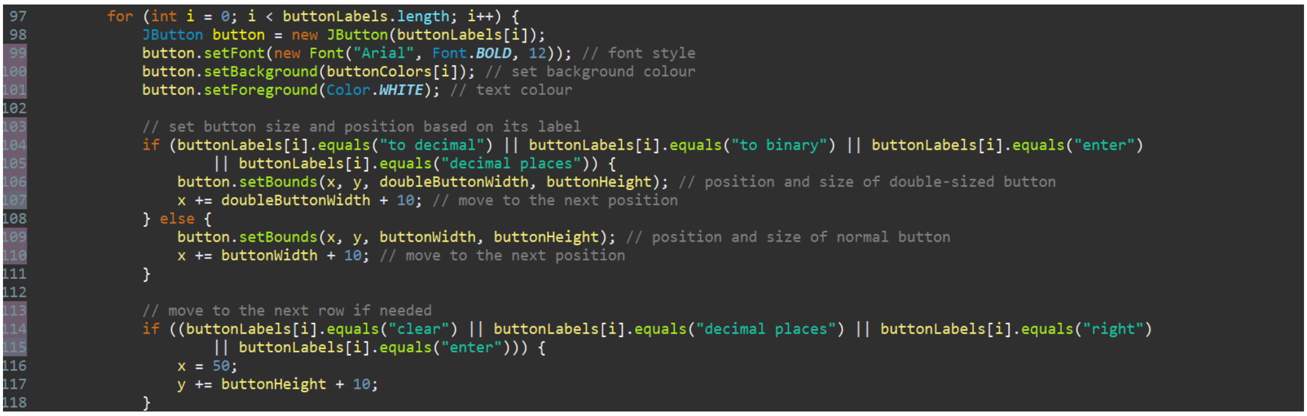
Define the regular size of a button and the double size buttons



```java
        int buttonWidth = 70, buttonHeight = 50; // size of a regular button
        int doubleButtonWidth = buttonWidth * 2 + 10; // size of a double-sized button
        int x = 50, y = 150; // position for the first row and column of buttons
```

Figure 28 setting the regular button size and double size ones

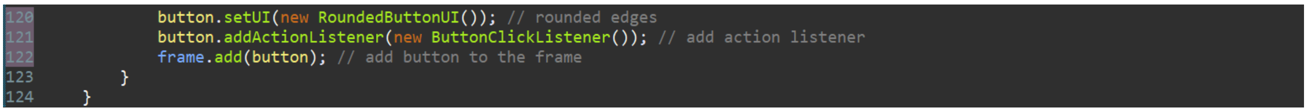
Setting a font for the buttons label and a text colour, and defining what buttons are double sized or regular sized and setting the spacing between buttons vertically and horizontally.



```
        for (int i = 0; i < buttonLabels.length; i++) {
            JButton button = new JButton(buttonLabels[i]);
            button.setFont(new Font("Arial", Font.BOLD, 12)); // font style
            button.setBackground(buttonColors[i]); // set background colour
            button.setForeground(Color.WHITE); // text colour

            // set button size and position based on its label
            if (buttonLabels[i].equals("to decimal") || buttonLabels[i].equals("to binary") || buttonLabels[i].equals("enter")
                    || buttonLabels[i].equals("decimal places")) {
                button.setBounds(x, y, doubleButtonWidth, buttonHeight); // position and size of double-sized button
                x += doubleButtonWidth + 10; // move to the next position
            } else {
                button.setBounds(x, y, buttonWidth, buttonHeight); // position and size of normal button
                x += buttonWidth + 10; // move to the next position
            }

            // move to the next row if needed
            if ((buttonLabels[i].equals("clear") || buttonLabels[i].equals("decimal places") || buttonLabels[i].equals("right")
                    || buttonLabels[i].equals("enter"))) {
                x = 50;
                y += buttonHeight + 10;
            }
```

Figure 29 setting the font of the button and defining the size of the double size buttons and the spacing

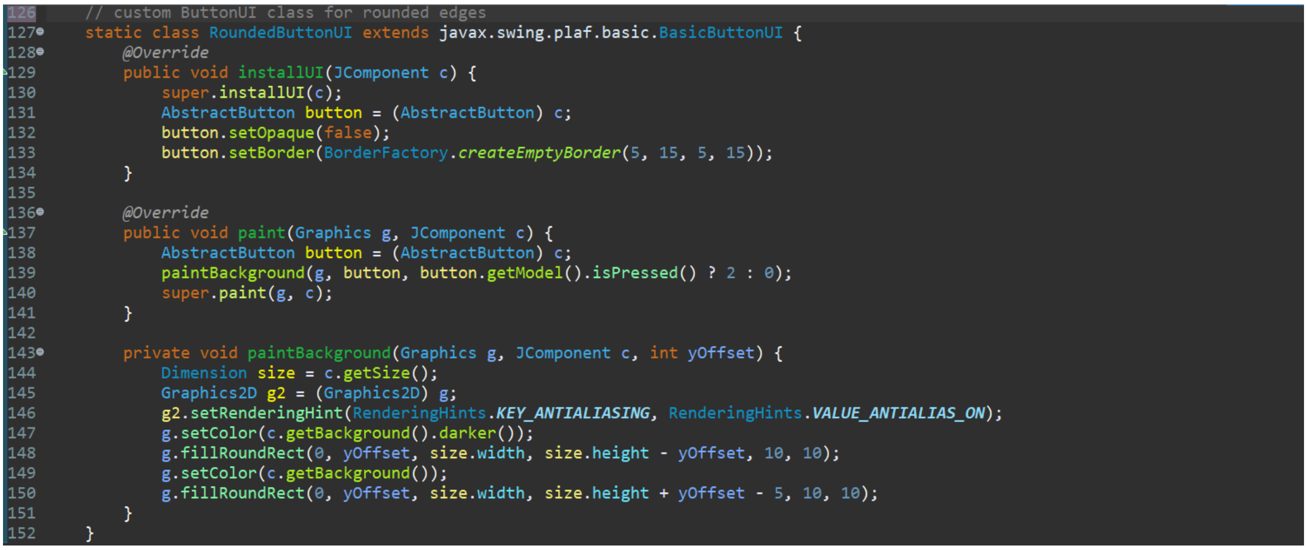
Making the edges round for the buttons and adding them to the frame.



```
            button.setUI(new RoundedButtonUI()); // rounded edges
            button.addActionListener(new ButtonClickListener()); // add action listener
            frame.add(button); // add button to the frame
        }
    }
```

Figure 30 making the buttons round edged

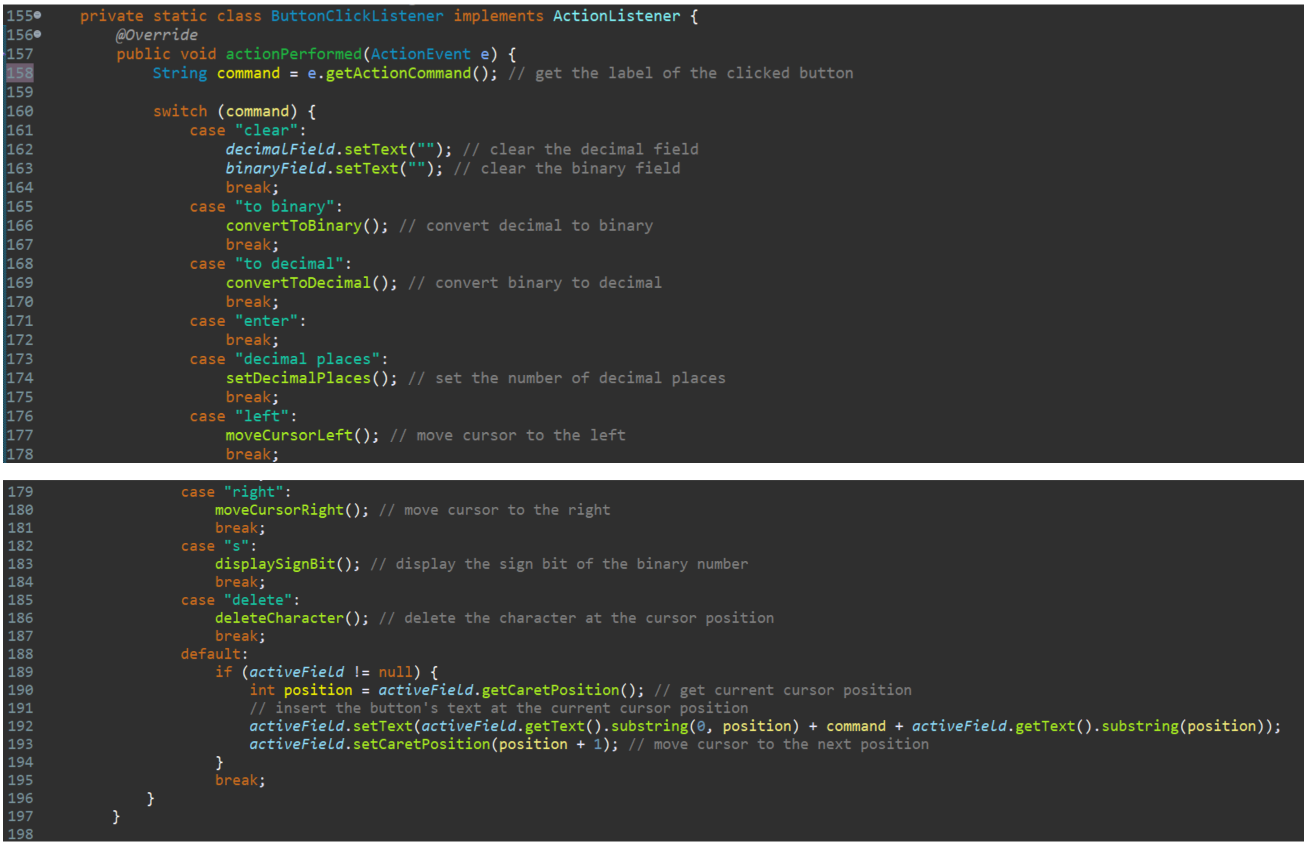
Styling the button to make it look 2D and have round edges and have a shadow.



```
    // custom ButtonUI class for rounded edges
    static class RoundedButtonUI extends javax.swing.plaf.basic.BasicButtonUI {
        @Override
        public void installUI(JComponent c) {
            super.installUI(c);
            AbstractButton button = (AbstractButton) c;
            button.setOpaque(false);
            button.setBorder(BorderFactory.createEmptyBorder(5, 15, 5, 15));
        }

        @Override
        public void paint(Graphics g, JComponent c) {
            AbstractButton button = (AbstractButton) c;
            paintBackground(g, button, button.getModel().isPressed() ? 2 : 0);
            super.paint(g, c);
        }

        private void paintBackground(Graphics g, JComponent c, int yOffset) {
            Dimension size = c.getSize();
            Graphics2D g2 = (Graphics2D) g;
            g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
            g.setColor(c.getBackground().darker());
            g.fillRoundRect(0, yOffset, size.width, size.height - yOffset, 10, 10);
            g.setColor(c.getBackground());
            g.fillRoundRect(0, yOffset, size.width, size.height + yOffset - 5, 10, 10);
        }
    }
```

Figure 31 implementing the round edge buttons

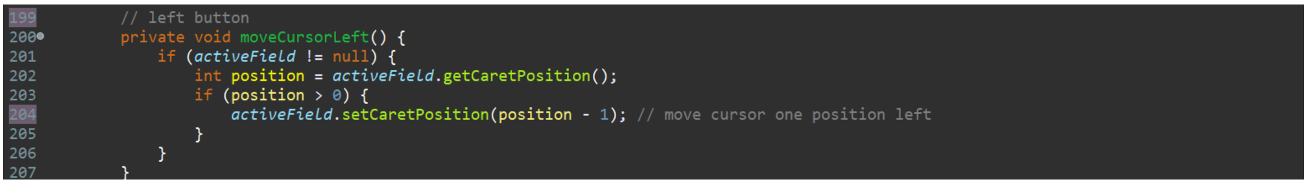
Giving each button a method that is called when the button is clicked.



```java
    private static class ButtonClickListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String command = e.getActionCommand(); // get the label of the clicked button

            switch (command) {
                case "clear":
                    decimalField.setText(""); // clear the decimal field
                    binaryField.setText(""); // clear the binary field
                    break;
                case "to binary":
                    convertToBinary(); // convert decimal to binary
                    break;
                case "to decimal":
                    convertToDecimal(); // convert binary to decimal
                    break;
                case "enter":
                    break;
                case "decimal places":
                    setDecimalPlaces(); // set the number of decimal places
                    break;
                case "left":
                    moveCursorLeft(); // move cursor to the left
                    break;
                case "right":
                    moveCursorRight(); // move cursor to the right
                    break;
                case "s":
                    displaySignBit(); // display the sign bit of the binary number
                    break;
                case "delete":
                    deleteCharacter(); // delete the character at the cursor position
                    break;
                default:
                    if (activeField != null) {
                        int position = activeField.getCaretPosition(); // get current cursor position
                        // insert the button's text at the current cursor position
                        activeField.setText(activeField.getText().substring(0, position) + command + activeField.getText().substring(position));
                        activeField.setCaretPosition(position + 1); // move cursor to the next position
                    }
                    break;
            }
        }

```

Figure 32 giving methods to each buttons

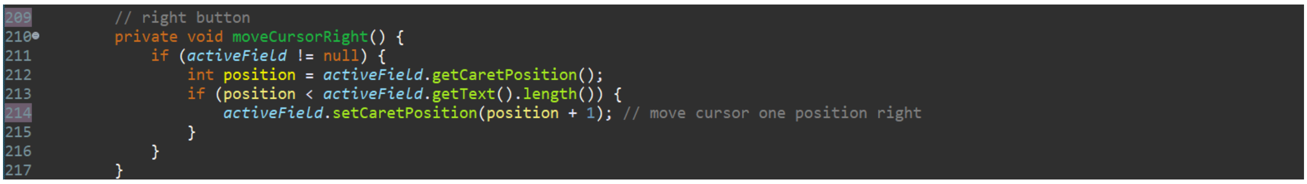
The ‘left’ button when pressed makes the cursor move one position to the left for easy editing.



```java
    // left button
    private void moveCursorLeft() {
        if (activeField != null) {
            int position = activeField.getCaretPosition();
            if (position > 0) {
                activeField.setCaretPosition(position - 1); // move cursor one position left
            }
        }
    }
```

Figure 33 method for the left button

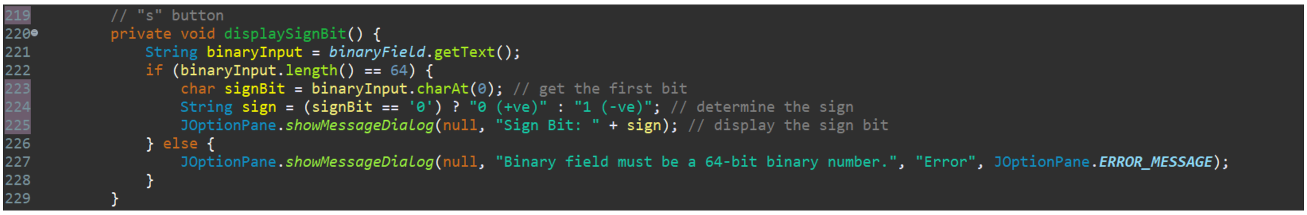
The ‘right’ button when pressed makes the cursor move one position to the left for easy editing.



```java
    // right button
    private void moveCursorRight() {
        if (activeField != null) {
            int position = activeField.getCaretPosition();
            if (position < activeField.getText().length()) {
                activeField.setCaretPosition(position + 1); // move cursor one position right
            }
        }
    }
```

Figure 34 method for the right button

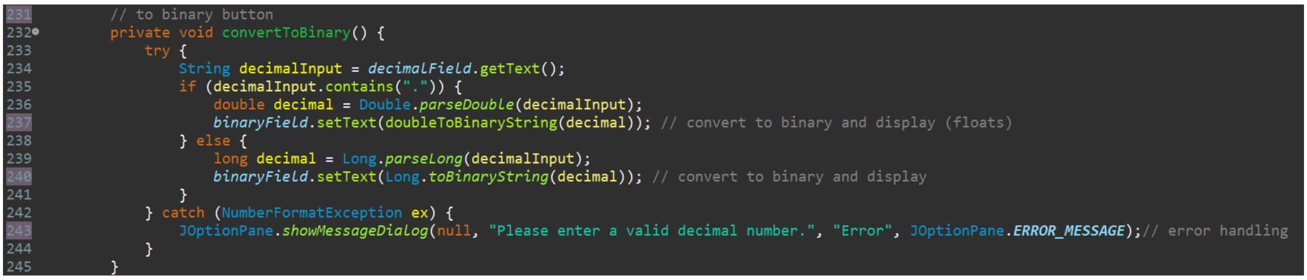
The sign bit buttin when pressed will output a message dialog box that has 0(+ve) if the binary string is positive and will output 1(-ve) if the string is negative.



```java
        // "s" button
        private void displaySignBit() {
            String binaryInput = binaryField.getText();
            if (binaryInput.length() == 64) {
                char signBit = binaryInput.charAt(0); // get the first bit
                String sign = (signBit == '0') ? "0 (+ve)" : "1 (-ve)"; // determine the sign
                JOptionPane.showMessageDialog(null, "Sign Bit: " + sign); // display the sign bit
            } else {
                JOptionPane.showMessageDialog(null, "Binary field must be a 64-bit binary number.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
```

Figure 35 method for the sign bit button

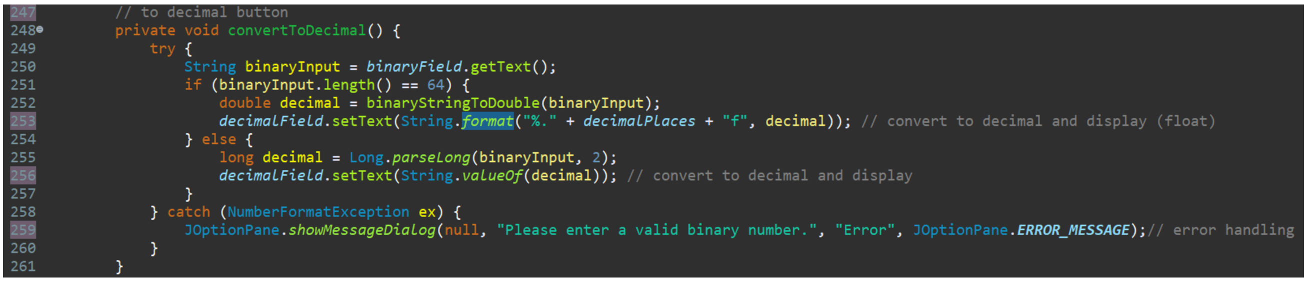
Defines a private method convertToBinary() that converts the decimal input from the user into its binary representation. It retrieves the text input from the decimal text field. If the input contains a decimal point, it is parsed as a double and converted to a binary string using the doubleToBinaryString() method, which handles floating-point numbers. If the input does not contain a decimal point, it is parsed as a long integer and converted to a binary string using Java's built-in Long.toBinaryString() method. The resulting binary string is then displayed in the binary text field. If the input is not a valid number, a Number Format Exception is caught, and an error message is displayed to the user using a JOptionPane.



```java
        // to binary button
        private void convertToBinary() {
            try {
                String decimalInput = decimalField.getText();
                if (decimalInput.contains(".")) {
                    double decimal = Double.parseDouble(decimalInput);
                    binaryField.setText(doubleToBinaryString(decimal)); // convert to binary and display (floats)
                } else {
                    long decimal = Long.parseLong(decimalInput);
                    binaryField.setText(Long.toBinaryString(decimal)); // convert to binary and display
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Please enter a valid decimal number.", "Error", JOptionPane.ERROR_MESSAGE);// error handling
            }
        }
```

Figure 36 method for the “to binary” button

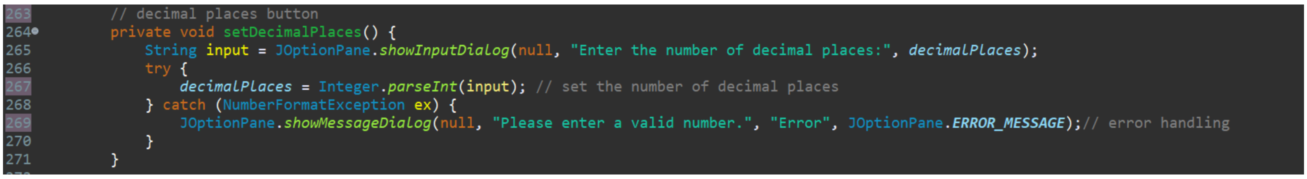
Defines a private method convertToDecimal() that converts the binary input from the user into its decimal representation. It retrieves the text input from the binary text field. If the input contains 64 bits, it is parsed as a binary string and converted to a double using the binaryStringToDouble() method, which handles floating-point numbers. If the input does not contain a 64 bits , it is parsed as a long binary and converted parsed as a long integer with base 2 using Long.parseLong(). The resulting decimal integer is then displayed in the decimal text field. If the input is not a valid binary number, a Number Format Exception is caught, and an error message is displayed to the user using a JOptionPane.



```java
        // to decimal button
        private void convertToDecimal() {
            try {
                String binaryInput = binaryField.getText();
                if (binaryInput.length() == 64) {
                    double decimal = binaryStringToDouble(binaryInput);
                    decimalField.setText(String.format("%." + decimalPlaces + "f", decimal)); // convert to decimal and display (float)
                } else {
                    long decimal = Long.parseLong(binaryInput, 2);
                    decimalField.setText(String.valueOf(decimal)); // convert to decimal and display
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Please enter a valid binary number.", "Error", JOptionPane.ERROR_MESSAGE);// error handling
            }
        }
```

Figure 37 method for the “to decimal” button

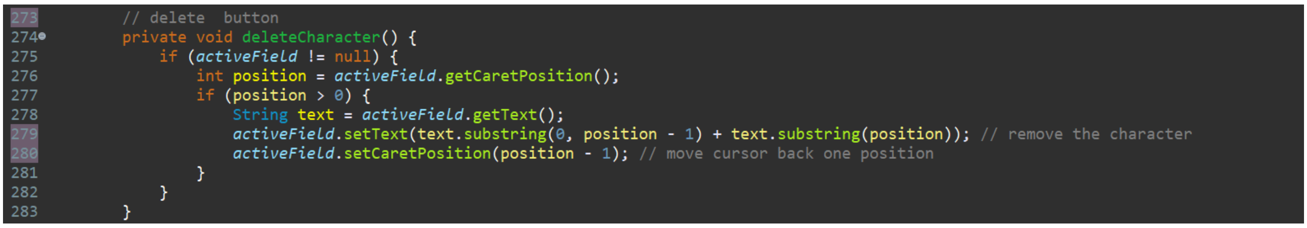
The decimal places button when clicked will output a dialog box that lets you input the number of decimal places you want to be on the screen. It has error handling in case the input is invalid.



```
    // decimal places button
    private void setDecimalPlaces() {
        String input = JOptionPane.showInputDialog(null, "Enter the number of decimal places:", decimalPlaces);
        try {
            decimalPlaces = Integer.parseInt(input); // set the number of decimal places
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Please enter a valid number.", "Error", JOptionPane.ERROR_MESSAGE);// error handling
        }
    }
```

Figure 38 method for the decimal places button

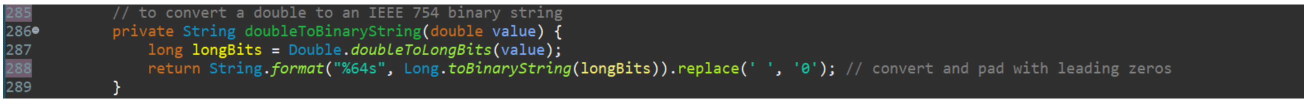
The delete button when pressed will delete the character present before the cursor which makes editing easier.



```
    // delete  button
    private void deleteCharacter() {
        if (activeField != null) {
            int position = activeField.getCaretPosition();
            if (position > 0) {
                String text = activeField.getText();
                activeField.setText(text.substring(0, position - 1) + text.substring(position)); // remove the character
                activeField.setCaretPosition(position - 1); // move cursor back one position
            }
        }
    }
```

Figure 39 method for the delete button

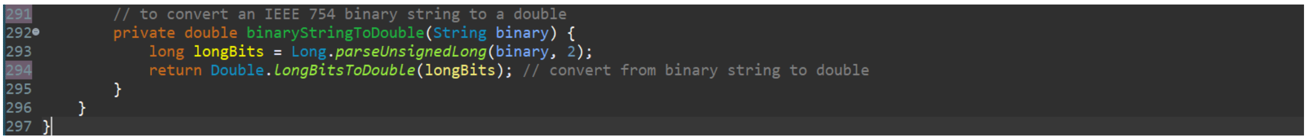
This method converts a double-precision floating-point number into its IEEE 754 binary string representation. It first converts the double to its IEEE 754 bit representation using Double.doubleToLongBits(), which returns the bit pattern as a long. Then, it converts this long value to a binary string using Long.toBinaryString(). The String.format("%64s", ...) ensures that the binary string is padded with leading zeros to make it 64 bits long. The replace(' ', '0') call replaces any space characters in the formatted string with zeros to ensure proper padding.



```
    // to convert a double to an IEEE 754 binary string
    private String doubleToBinaryString(double value) {
        long longBits = Double.doubleToLongBits(value);
        return String.format("%64s", Long.toBinaryString(longBits)).replace(' ', '0'); // convert and pad with leading zeros
    }
```

Figure 40 method to make a double into an IEEE 754 binary string

This method converts an IEEE 754 binary string representation back into a double-precision floating-point number. It first parses the binary string as an unsigned long using Long.parseUnsignedLong(binary, 2), which interprets the binary string as a base-2 (binary) number. Then, it converts this long value back to a double using Double.longBitsToDouble(), which interprets the bit pattern as a double-precision floating-point number according to the IEEE 754 standard.



```
        // to convert an IEEE 754 binary string to a double
        private double binaryStringToDouble(String binary) {
            long longBits = Long.parseUnsignedLong(binary, 2);
            return Double.longBitsToDouble(longBits); // convert from binary string to double
        }
    }
}
```

Figure 41 method to convert an IEEE binary string into a double

Conclusion:

To sum up, this project has met its goals by producing a strong and easy-to-use calculator app that makes decimal float-points into binary strings using IEEE 754 double precision. Doing this has only made it more clear just how important requirements analysis, design, implementation, and testing are. Indeed, if one of those steps were done poorly, the resulting app would be of little use to our users.

References:

Java calculator app 🖩 (2020) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=dfhmTyRTCSQ&t=12s (Accessed: 15 July 2024).

Java GUI Tutorial - Make a GUI in 13 Minutes #99 (2020) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=5o3fMLPY7qY (Accessed: 16 July 2024).

Java Convert String to double - javatpoint (no date) www.javatpoint.com. Available at: https://www.javatpoint.com/java-string-to-double (Accessed: 16 July 2024).

Java Convert double to String - javatpoint (no date) www.javatpoint.com. Available at: https://www.javatpoint.com/java-double-to-string (Accessed: 16 July 2024).

GeeksforGeeks (2021) Java Swing: Simple Calculator, GeeksforGeeks. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/java-swing-simple-calculator/ (Accessed: 9 July 2024).