```java
package Project1_Convertersnumber;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.math.BigDecimal;
public class Jnumberconvert extends JFrame {
    private JComboBox<String> fromcbb;
    private JComboBox<String> tocbb;
    private JButton Convert;
    private JButton Reset;
    private JButton Swap;
    private JTextField Enternumber;
    private JTextField resultnumberalsystem;
    private JTextField resultcomplement;
    private static final String[] BASES = {"Decimal", "Binary", "Octal", "Hexadecimal"};
    private static final int[] RADIX_VALUES = {10, 2, 8, 16};

    public Jnumberconvert() {

        setTitle("Number System Converter");
        setSize(580, 400); // Slightly larger dimensions
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel panel = new JPanel();
        panel.setBackground(new Color(0xFAFAD2));
        panel.setLayout(null);

        // Form
        fromcbb = new JComboBox<>(BASES);
        fromcbb.setBounds(30, 70, 220, 30); // Larger dimensions
        JLabel fromLabel = new JLabel("From:");
        fromLabel.setBounds(30, 40, 100, 20); // Adjusted position and size
        panel.add(fromLabel);
        panel.add(fromcbb);

        // To
        tocbb = new JComboBox<>(BASES);
        tocbb.setBounds(320, 70, 220, 30); // Larger dimensions
        JLabel toLabel = new JLabel("To:");
        toLabel.setBounds(320, 40, 100, 20); // Adjusted position and size
        panel.add(toLabel);
        panel.add(tocbb);

        // Initialize fromcbb and tocbb indices to reflect "Decimal to Hexadecimal" conversion
        fromcbb.setSelectedIndex(0); // Decimal
        tocbb.setSelectedIndex(3);   // Hexadecimal

        // Header label
        JLabel headerLabel = new JLabel();
        headerLabel.setBounds(30, 10, 300, 30); // Adjusted size
```

```
54          updateHeaderLabel(headerLabel); // Initialize header Label
55          panel.add(headerLabel);
56
57          // Convert button
58          Convert = new JButton("= Convert");
59          Convert.setBounds(30, 190, 120, 30); // Adjusted size
60          Convert.setBackground(new Color(0x00ff00));
61          panel.add(Convert);
62
63          // Reset button
64          Reset = new JButton("x Reset");
65          Reset.setBounds(170, 190, 120, 30); // Adjusted size
66          Reset.setBackground(new Color(0x808080));
67          panel.add(Reset);
68
69          // Swap button
70          Swap = new JButton("Swap");
71          Swap.setBounds(310, 190, 120, 30); // Adjusted size
72          Swap.setBackground(new Color(0x808080));
73          panel.add(Swap);
74
75          // Enter number JTextField
76          Enternumber = new JTextField();
77          Enternumber.setBounds(30, 130, 510, 30); // Larger
    dimensions
78          Enternumber.setEditable(true);
79          panel.add(Enternumber);
80
81          // Enter decimal to hex,...
82          JLabel LEnternumber = new JLabel();
83          LEnternumber.setBounds(30, 100, 510, 30); // Larger
    dimensions
84          updateEnternumber(LEnternumber);
85          panel.add(LEnternumber);
86
87          // Result number in the selected system JTextField
88          resultnumberalsystem = new JTextField();
89          resultnumberalsystem.setEditable(false);
90          resultnumberalsystem.setBounds(30, 250, 510, 30); // Larger
     dimensions
91          panel.add(resultnumberalsystem);
92
93          // Result number in the selected system JLabel
94          JLabel Lresultnumberalsystem = new JLabel();
95          Lresultnumberalsystem.setBounds(30, 220, 510, 30); //
    Larger dimensions
96          updateresultnumberalsystem(Lresultnumberalsystem);
97          panel.add(Lresultnumberalsystem);
98
99          // Result complement JTextfield
100         resultcomplement = new JTextField();
101         resultcomplement.setEditable(false);
102         resultcomplement.setBounds(30, 310, 510, 30); // Larger
    dimensions
103         panel.add(resultcomplement);
104
105         // Result complement JLabel
```

```
106         JLabel Lresultcomplement = new JLabel();
107         Lresultcomplement.setBounds(30, 280, 510, 30); // Larger
    dimensions
108         updateresultcomplement(Lresultcomplement);
109         panel.add(Lresultcomplement);
110 
111         setContentPane(panel);
112 
113         Convert.addActionListener(new ActionListener() {
114             @Override
115             public void actionPerformed(ActionEvent e) {
116 
117                 // Convert number system
118 
119                 try {
120 
121                     String input = Enternumber.getText().trim();
122                     int selectedFromIndex = fromcbb.
    getSelectedIndex();
123                     int selectedToIndex = tocbb.getSelectedIndex();
124 
125                     // Decimal to Hexadecimal
126                     if (selectedFromIndex == 0 && selectedToIndex
    == 3) {
127                         try {
128                             BigDecimal decimalValue = new
    BigDecimal(input);
129                             boolean isNegative = false;
130                             if (decimalValue.compareTo(BigDecimal.
    ZERO) < 0) {
131                                 isNegative = true;
132                                 decimalValue = decimalValue.abs();
133                             }
134                             int intPart = decimalValue.intValue();
135                             String hexIntPart = Integer.toHexString
    (intPart).toUpperCase();
136                             BigDecimal fractionalPart =
    decimalValue.subtract(new BigDecimal(intPart));
137                             StringBuilder hexFractionalPart = new
    StringBuilder();
138                             for (int i = 0; i < 16; i++) {
139                                 fractionalPart = fractionalPart.
    multiply(BigDecimal.valueOf(16));
140                                 int hexDigit = fractionalPart.
    intValue();
141                                 hexFractionalPart.append(Integer.
    toHexString(hexDigit).toUpperCase());
142                                 fractionalPart = fractionalPart.
    subtract(BigDecimal.valueOf(hexDigit));
143                                 if (fractionalPart.compareTo(
    BigDecimal.ZERO) == 0) {
144                                     break;
145                                 }
146                             }
147                             String hexValue = hexIntPart;
148                             if (hexFractionalPart.length() > 0) {
149                                 hexValue += "." + hexFractionalPart
```

```
    .toString();
                            }
                            if (isNegative) {
                                hexValue = "-" + hexValue;
                            }
                            resultnumberalsystem.setText(hexValue);
                            resultcomplement.setText("N/A");
                        } catch (Exception ex) {
                            throw new RuntimeException(ex);
                        }
                    }

                    // Hexadecimal to Decimal
                    else if (selectedFromIndex == 3 &&
    selectedToIndex == 0) {
                        try {
                            String hexValue = input.toUpperCase();
                            boolean isNegative = false;
                            if (hexValue.startsWith("-")) {
                                hexValue = hexValue.substring(1);
                                isNegative = true;
                            }
                            int dotIndex = hexValue.indexOf('.');
                            String hexIntegerPart;
                            String hexFractionalPart;
                            if (dotIndex != -1) {
                                hexIntegerPart = hexValue.substring
    (0, dotIndex);
                                hexFractionalPart = hexValue.
    substring(dotIndex + 1);
                            } else {
                                hexIntegerPart = hexValue;
                                hexFractionalPart = "";
                            }
                            int decimalIntegerPart = Integer.
    parseInt(hexIntegerPart, 16);
                            double decimalFractionalPart = 0.0;
                            if (!hexFractionalPart.isEmpty()) {
                                int fractionalHex = Integer.
    parseInt(hexFractionalPart, 16);
                                decimalFractionalPart =
    fractionalHex / Math.pow(16, hexFractionalPart.length());
                            }
                            double decimalResult =
    decimalIntegerPart + decimalFractionalPart;
                            if (isNegative) {
                                resultnumberalsystem.setText(String
    .format("%.10f", -decimalResult));
                            } else {
                                resultnumberalsystem.setText(String
    .format("%.10f", decimalResult));
                            }
                            resultcomplement.setText("N/A");
                        } catch (Exception ex) {
                            throw new RuntimeException(ex);
                        }
                    }
```

```
197
198                     // Binary to Decimal
199                     else if (selectedFromIndex == 1 &&
    selectedToIndex == 0) {
200                         try {
201                             String binaryValue = input;
202                             // Check for negative binary number
203                             boolean isNegative = false;
204                             if (binaryValue.startsWith("-")) {
205                                 isNegative = true;
206                                 // Remove the negative sign for
    processing
207                                 binaryValue = binaryValue.substring
    (1);
208                             }
209                             String[] parts = binaryValue.split("\\.
    ");
210                             // Convert the integer part to decimal
211                             int decimalIntegerPart = Integer.
    parseInt(parts[0], 2);
212                             double decimalFractionalPart = 0.0;
213                             // Convert the fractional part to
    decimal
214                             if (parts.length > 1 && !parts[1].
    isEmpty()) {
215                                 String fractionalPart = parts[1];
216                                 for (int i = 0; i < fractionalPart.
    length(); i++) {
217                                     int digit = Character.
    getNumericValue(fractionalPart.charAt(i));
218                                     decimalFractionalPart += digit
    / Math.pow(2, i + 1);
219                                 }
220                             }
221                             double decimalResult =
    decimalIntegerPart + decimalFractionalPart;
222                             if (isNegative) {
223                                 decimalResult = -decimalResult;
224                             }
225                             resultnumberalsystem.setText(Double.
    toString(decimalResult));
226                             resultcomplement.setText("N/A");
227                         } catch (NumberFormatException ex) {
228                             resultnumberalsystem.setText("Invalid
    input");
229                         }
230                     }
231
232                     // Decimal to Binary
233                     else if (selectedFromIndex == 0 &&
    selectedToIndex == 1) {
234                         try {
235                             String decimalValueStr = input;
236                             // Parse the decimal value
237                             double decimalValue = Double.
    parseDouble(decimalValueStr);
238                             // Check for negative decimal value
```

```
239                             boolean isNegative = false;
240                             if (decimalValue < 0) {
241                                 isNegative = true;
242                                 decimalValue = -decimalValue; //
    Make it positive for processing
243                             }
244                             // Initialize variables for binary
    parts
245                             StringBuilder binaryIntegerPart = new
    StringBuilder();
246                             StringBuilder binaryFractionalPart =
    new StringBuilder();
247                             // Convert the integer part to binary
248                             int integerPart = (int) decimalValue;
249                             do {
250                                 binaryIntegerPart.insert(0,
    integerPart % 2);
251                                 integerPart /= 2;
252                             } while (integerPart > 0);
253                             // Convert the fractional part to
    binary
254                             double fractionalPart = decimalValue -
    (int) decimalValue;
255                             for (int i = 0; i < 16; i++) { //
    Convert up to 16 bits of fractional part
256                                 fractionalPart *= 2;
257                                 binaryFractionalPart.append((int)
    fractionalPart);
258                                 fractionalPart -= (int)
    fractionalPart;
259                             }
260                             // Combine integer and fractional
    binary parts
261                             StringBuilder binaryValue = new
    StringBuilder(binaryIntegerPart);
262                             if (binaryFractionalPart.length() > 0)
    {
263                                 binaryValue.append('.').append(
    binaryFractionalPart);
264                             }
265                             // If it was a negative decimal number,
     add a '-' sign to the binary result
266                             if (isNegative) {
267                                 binaryValue.insert(0, '-');
268                             }
269                             resultnumberalsystem.setText(
    binaryValue.toString());
270                             // Display "N/A" for Binary signed 2's
    complement
271                             resultcomplement.setText("N/A");
272                         } catch (NumberFormatException ex) {
273                             resultnumberalsystem.setText("Invalid
    input");
274                         }
275                     }
276
277                     // Binary to Hexadecimal conversion
```

```
                } else if (selectedFromIndex == 1 &&
selectedToIndex == 3) {
                    try {
                        String binaryValue = input;
                        // Check if the binary value is
negative
                        boolean isNegative = false;
                        if (binaryValue.startsWith("-")) {
                            isNegative = true;
                            // Remove the negative sign for
processing
                            binaryValue = binaryValue.substring
(1);
                        }
                        // Split the binary number into integer
 and fractional parts
                        String[] parts = binaryValue.split("\\.
");
                        String binaryIntegerPart = parts[0];
                        String binaryFractionalPart = (parts.
length > 1) ? parts[1] : "";
                        // Pad the fractional part with zeros
to have a length multiple of 4
                        while (binaryFractionalPart.length() %
4 != 0) {
                            binaryFractionalPart += "0";
                        }
                        // Initialize variables for the
hexadecimal parts
                        String hexIntegerPart = "";
                        String hexFractionalPart = "";
                        // Convert the integer part to
hexadecimal
                        if (!binaryIntegerPart.isEmpty()) {
                            hexIntegerPart = Integer.
toHexString(Integer.parseInt(binaryIntegerPart, 2)).toUpperCase
();
                        }
                        // Convert the fractional part to
hexadecimal
                        for (int i = 0; i <
binaryFractionalPart.length(); i += 4) {
                            String nibble =
binaryFractionalPart.substring(i, i + 4);
                            hexFractionalPart += Integer.
toHexString(Integer.parseInt(nibble, 2)).toUpperCase();
                        }
                        // Add a '-' sign to both the
hexadecimal and decimal representations if it was a negative
binary number
                        if (isNegative) {
                            hexIntegerPart = "-" +
hexIntegerPart;
                        }
                        // Combine integer and fractional
hexadecimal parts
                        String hexValue = hexIntegerPart;
```

```
                            if (!hexFractionalPart.isEmpty()) {
                                hexValue += "." + hexFractionalPart
    ;
                            }
                            resultnumberalsystem.setText(hexValue);
                            // Calculate the Decimal number
    equivalent (integer + fractional)
                            double decimalEquivalent = 0.0;
                            if (!hexIntegerPart.isEmpty()) {
                                decimalEquivalent += Integer.
    parseInt(binaryIntegerPart, 2);
                            }
                            if (!hexFractionalPart.isEmpty()) {
                                for (int i = 0; i <
    hexFractionalPart.length(); i++) {
                                    char hexDigit =
    hexFractionalPart.charAt(i);
                                    decimalEquivalent += Integer.
    parseInt(String.valueOf(hexDigit), 16) / Math.pow(16, i + 1);
                                }
                            }
                            // Check if the binary value was
    negative and adjust the decimal equivalent accordingly
                            if (isNegative) {
                                decimalEquivalent = -
    decimalEquivalent;
                            }
                            // Display the Decimal number
    equivalent (3 digits)
                            resultcomplement.setText(String.format(
    "%.3f", decimalEquivalent));
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Hexadecimal to Binary conversion
                    else if (selectedFromIndex == 3 &&
    selectedToIndex == 1) {
                        try {
                            String hexValue = input;
                            // Check if the hexadecimal value is
    negative (if it starts with '-')
                            boolean isNegative = hexValue.
    startsWith("-");
                            if (isNegative) {
                                // Remove the negative sign for
    processing
                                hexValue = hexValue.substring(1);
                            }
                            // Split the hexadecimal number into
    integer and fractional parts
                            String[] parts = hexValue.split("\\.");
                            String hexIntegerPart = parts[0];
                            String hexFractionalPart = (parts.
    length > 1) ? parts[1] : "";
```

```
355                             // Initialize variables for the binary
    parts
356                             StringBuilder binaryIntegerPart = new
    StringBuilder();
357                             StringBuilder binaryFractionalPart =
    new StringBuilder();
358                             // Convert the integer part to binary
359                             if (!hexIntegerPart.isEmpty()) {
360                                 binaryIntegerPart.append(Integer.
    toBinaryString(Integer.parseInt(hexIntegerPart, 16)));
361                             }
362                             // Convert the fractional part to
    binary
363                             if (!hexFractionalPart.isEmpty()) {
364                                 for (int i = 0; i <
    hexFractionalPart.length(); i++) {
365                                     char hexDigit =
    hexFractionalPart.charAt(i);
366                                     // Convert each hex digit to
    binary and pad with leading zeros as needed
367                                     String binaryNibble = Integer.
    toBinaryString(Integer.parseInt(String.valueOf(hexDigit), 16));
368                                     while (binaryNibble.length() <
    4) {
369                                         binaryNibble = "0" +
    binaryNibble;
370                                     }
371                                     binaryFractionalPart.append(
    binaryNibble);
372                                 }
373                             }
374                             // Combine integer and fractional
    binary parts
375                             String binaryValue = binaryIntegerPart.
    toString();
376                             if (binaryFractionalPart.length() > 0)
    {
377                                 binaryValue += "." +
    binaryFractionalPart.toString();
378                             }
379                             // Add a '-' sign to the binary
    representation if it was a negative hexadecimal number
380                             if (isNegative) {
381                                 binaryValue = "-" + binaryValue;
382                             }
383                             resultnumberalsystem.setText(
    binaryValue);
384                             // Calculate the Decimal number
    equivalent
385                             double decimalEquivalent = 0.0;
386                             // Convert the integer part to decimal
387                             if (!hexIntegerPart.isEmpty()) {
388                                 decimalEquivalent += Integer.
    parseInt(hexIntegerPart, 16);
389                             }
390                             // Convert the fractional part to
    decimal
```

```
                            if (!hexFractionalPart.isEmpty()) {
                                for (int i = 0; i < hexFractionalPart.length(); i++) {
                                    char hexDigit = hexFractionalPart.charAt(i);
                                    int digitValue = Integer.parseInt(String.valueOf(hexDigit), 16);
                                    decimalEquivalent += digitValue / Math.pow(16, i + 1);
                                }
                            }
                            // Check if the hexadecimal value was negative and adjust the decimal equivalent accordingly
                            if (isNegative) {
                                decimalEquivalent = -decimalEquivalent;
                            }
                            // Display the Decimal number equivalent (10 digits)
                            resultcomplement.setText(String.format("%.10f", decimalEquivalent));
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Binary to Octal conversion
                    else if (selectedFromIndex == 1 && selectedToIndex == 2) {
                        try {
                            String binaryValue = input;
                            // Check if the binary value is negative
                            boolean isNegative = false;
                            if (binaryValue.startsWith("-")) {
                                isNegative = true;
                                // Remove the negative sign for processing
                                binaryValue = binaryValue.substring(1);
                            }
                            // Split the binary number into integer and fractional parts
                            String[] parts = binaryValue.split("\\.");
                            String binaryIntegerPart = parts[0];
                            String binaryFractionalPart = (parts.length > 1) ? parts[1] : "";
                            // Pad the fractional part with zeros to have a length multiple of 3
                            while (binaryFractionalPart.length() % 3 != 0) {
                                binaryFractionalPart += "0";
                            }
                            // Initialize variables for the octal parts
```

```
                            String octalIntegerPart = "";
                            String octalFractionalPart = "";
                            // Convert the integer part to octal
                            if (!binaryIntegerPart.isEmpty()) {
                                octalIntegerPart = Integer.
toOctalString(Integer.parseInt(binaryIntegerPart, 2));
                            }
                            // Convert the fractional part to octal
                            for (int i = 0; i <
binaryFractionalPart.length(); i += 3) {
                                String nibble =
binaryFractionalPart.substring(i, i + 3);
                                octalFractionalPart += Integer.
toOctalString(Integer.parseInt(nibble, 2));
                            }
                            // Add a '-' sign to the octal
representation if it was a negative binary number
                            if (isNegative) {
                                octalIntegerPart = "-" +
octalIntegerPart;
                            }
                            // Combine integer and fractional octal
 parts
                            String octalValue = octalIntegerPart;
                            if (!octalFractionalPart.isEmpty()) {
                                octalValue += "." +
octalFractionalPart;
                            }
                            resultnumberalsystem.setText(octalValue
);
                            // Calculate the Decimal number
equivalent (integer + fractional)
                            double decimalEquivalent = 0.0;
                            if (!octalIntegerPart.isEmpty()) {
                                decimalEquivalent += Integer.
parseInt(binaryIntegerPart, 2);
                            }
                            if (!octalFractionalPart.isEmpty()) {
                                for (int i = 0; i <
octalFractionalPart.length(); i++) {
                                    char octalDigit =
octalFractionalPart.charAt(i);
                                    decimalEquivalent += Integer.
parseInt(String.valueOf(octalDigit), 8) / Math.pow(8, i + 1);
                                }
                            }
                            // Check if the binary value was
negative and adjust the decimal equivalent accordingly
                            if (isNegative) {
                                decimalEquivalent = -
decimalEquivalent;
                            }
                            // Display the Decimal number
equivalent (3 digits)
                            resultcomplement.setText(String.format(
"%.3f", decimalEquivalent));
                        } catch (NumberFormatException ex) {
```

```
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Octal to Binary conversion
                    else if (selectedFromIndex == 2 &&
    selectedToIndex == 1) {
                        try {
                            String octalValue = input;
                            // Check if the octal value is negative
                            boolean isNegative = false;
                            if (octalValue.startsWith("-")) {
                                isNegative = true;
                                // Remove the negative sign for
    processing
                                octalValue = octalValue.substring
    (1);
                            }
                            // Split the octal number into integer
    and fractional parts
                            String[] parts = octalValue.split("\\."
    );
                            String octalIntegerPart = parts[0];
                            String octalFractionalPart = (parts.
    length > 1) ? parts[1] : "";
                            // Initialize variables for the binary
    parts
                            String binaryIntegerPart = "";
                            String binaryFractionalPart = "";
                            // Convert the integer part to binary
                            if (!octalIntegerPart.isEmpty()) {
                                binaryIntegerPart = Integer.
    toBinaryString(Integer.parseInt(octalIntegerPart, 8));
                            }
                            // Convert the fractional part to
    binary
                            for (int i = 0; i < octalFractionalPart
    .length(); i++) {
                                char octalDigit =
    octalFractionalPart.charAt(i);
                                String binaryNibble = Integer.
    toBinaryString(Integer.parseInt(String.valueOf(octalDigit), 8))
    ;
                                // Ensure that each binary nibble
    has 3 digits
                                while (binaryNibble.length() < 3) {
                                    binaryNibble = "0" +
    binaryNibble;
                                }
                                binaryFractionalPart +=
    binaryNibble;
                            }
                            // Combine integer and fractional
    binary parts
                            String binaryValue = binaryIntegerPart;
                            if (!binaryFractionalPart.isEmpty()) {
```

```
                                binaryValue += "." +
    binaryFractionalPart;
                            }
                            // Display the Binary number with a
    negative sign if applicable
                            resultnumberalsystem.setText(isNegative
     ? "-" + binaryValue : binaryValue);
                            // Calculate the Decimal number
    equivalent (integer + fractional)
                            double decimalEquivalent = 0.0;
                            if (!binaryIntegerPart.isEmpty()) {
                                decimalEquivalent += Integer.
    parseInt(binaryIntegerPart, 2);
                            }
                            if (!binaryFractionalPart.isEmpty()) {
                                for (int i = 0; i <
    binaryFractionalPart.length(); i++) {
                                    char binaryDigit =
    binaryFractionalPart.charAt(i);
                                    decimalEquivalent += Integer.
    parseInt(String.valueOf(binaryDigit), 2) / Math.pow(2, i + 1);
                                }
                            }
                            // Display the Decimal number with a
    negative sign if applicable
                            resultcomplement.setText(isNegative ? "
    -" + decimalEquivalent : String.valueOf(decimalEquivalent));
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Octal to Hexadecimal conversion
                    else if (selectedFromIndex == 2 &&
    selectedToIndex == 3) {
                        try {
                            String octalValue = input;
                            // Check if the octal value is negative
                            boolean isNegative = false;
                            if (octalValue.startsWith("-")) {
                                isNegative = true;
                                // Remove the negative sign for
    processing
                                octalValue = octalValue.substring
    (1);
                            }
                            // Split the octal number into integer
    and fractional parts
                            String[] parts = octalValue.split("\\."
    );
                            String octalIntegerPart = parts[0];
                            String octalFractionalPart = (parts.
    length > 1) ? parts[1] : "0"; // Default to "0" if no
    fractional part
                            // Convert the octal value to decimal
                            int decimalIntegerPart = Integer.
```

```
    parseInt(octalIntegerPart, 8);
                        // Convert the fractional part to decimal
                        double decimalFractionalPart = 0.0;
                        for (int i = 0; i < octalFractionalPart.length(); i++) {
                            char octalDigit = octalFractionalPart.charAt(i);
                            int digitValue = Character.getNumericValue(octalDigit);
                            decimalFractionalPart += digitValue / Math.pow(8, i + 1);
                        }
                        // Combine the integer and fractional parts
                        double decimalValue = decimalIntegerPart + decimalFractionalPart;
                        // Convert the decimal value to hexadecimal using a custom function
                        String hexValue = convertDecimalToHex(decimalValue);
                        // Add a '-' sign to the hexadecimal representation if it was a negative octal number
                        if (isNegative) {
                            hexValue = "-" + hexValue;
                        }
                        // Display the Hexadecimal number with a negative sign if applicable
                        resultnumberalsystem.setText(hexValue);
                        // Format the Decimal answer to match the desired format
                        String decimalString = String.format("%.6f", decimalValue);
                        if (decimalString.endsWith(".000000")) {
                            decimalString = decimalString.substring(0, decimalString.length() - 7); // Remove trailing ".000000"
                        }
                        // Display the Decimal number equivalent with a negative sign if applicable
                        resultcomplement.setText(isNegative ? "-" + decimalString : decimalString);
                    } catch (NumberFormatException ex) {
                        resultnumberalsystem.setText("N/A");
                        resultcomplement.setText("N/A");
                    }
                }

                // Hexadecimal to Octal conversion
                else if (selectedFromIndex == 3 && selectedToIndex == 2) {
                    try {
                        String hexValue = input;
                        // Check if the hex value is negative
                        boolean isNegative = false;
                        if (hexValue.startsWith("-")) {
```

```
                                isNegative = true;
                                // Remove the negative sign for
    processing
                                hexValue = hexValue.substring(1);
                            }
                            // Remove the "0x" prefix if present
                            if (hexValue.startsWith("0x") ||
    hexValue.startsWith("0X")) {
                                hexValue = hexValue.substring(2);
                            }
                            // Split the hex number into integer
    and fractional parts
                            String[] parts = hexValue.split("\\.");
                            String hexIntegerPart = parts[0];
                            String hexFractionalPart = (parts.
    length > 1) ? parts[1] : "0"; // Default to "0" if no
    fractional part
                            // Convert the hex value to decimal
                            double decimalIntegerPart = Integer.
    parseInt(hexIntegerPart, 16);
                            // Convert the fractional part to
    decimal
                            double decimalFractionalPart = 0.0;
                            for (int i = 0; i < hexFractionalPart.
    length(); i++) {
                                char hexDigit = hexFractionalPart.
    charAt(i);
                                int digitValue = Character.digit(
    hexDigit, 16);
                                decimalFractionalPart += digitValue
     / Math.pow(16, i + 1);
                            }
                            // Combine the integer and fractional
    parts
                            double decimalValue =
    decimalIntegerPart + decimalFractionalPart;
                            // Convert the decimal value to octal
                            String octalValue =
    convertDecimalToOctal(decimalValue);
                            // Add a '-' sign to the octal
    representation if it was a negative hex number
                            if (isNegative) {
                                octalValue = "-" + octalValue;
                            }
                            // Display the Octal number with a
    negative sign if applicable
                            resultnumberalsystem.setText(octalValue
    );
                            // Format the Decimal answer to match
    the desired format
                            String decimalString = String.format("
    %.9f", decimalValue);
                            if (decimalString.endsWith(".000000000"
    )) {
                                decimalString = decimalString.
    substring(0, decimalString.length() - 10); // Remove trailing
    ".000000000"
```

```
                            }
                            // Display the Decimal number
    equivalent with a negative sign if applicable
                            resultcomplement.setText(isNegative ? "
    -" + decimalString : decimalString);
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Octal to Decimal conversion
                    else if (selectedFromIndex == 2 &&
    selectedToIndex == 0) {
                        try {
                            String octalValue = input;
                            // Check if the octal value is negative
                            boolean isNegative = false;
                            if (octalValue.startsWith("-")) {
                                isNegative = true;
                                // Remove the negative sign for
    processing
                                octalValue = octalValue.substring
    (1);
                            }
                            // Split the octal number into integer
    and fractional parts
                            String[] parts = octalValue.split("\\."
    );
                            String octalIntegerPart = parts[0];
                            String octalFractionalPart = (parts.
    length > 1) ? parts[1] : "0"; // Default to "0" if no
    fractional part
                            // Convert the octal integer part to
    decimal
                            double decimalIntegerPart = 0.0;
                            for (int i = octalIntegerPart.length()
    - 1; i >= 0; i--) {
                                char octalDigit = octalIntegerPart.
    charAt(i);
                                int digitValue = Character.
    getNumericValue(octalDigit);
                                decimalIntegerPart += digitValue *
    Math.pow(8, octalIntegerPart.length() - i - 1);
                            }
                            // Convert the octal fractional part to
     decimal
                            double decimalFractionalPart = 0.0;
                            for (int i = 0; i < octalFractionalPart
    .length(); i++) {
                                char octalDigit =
    octalFractionalPart.charAt(i);
                                int digitValue = Character.
    getNumericValue(octalDigit);
                                decimalFractionalPart += digitValue
     / Math.pow(8, i + 1);
                            }
```

```
                            // Combine the integer and fractional
    parts
                            double decimalValue =
    decimalIntegerPart + decimalFractionalPart;
                            // Display the Decimal number with a
    negative sign if applicable
                            String decimalString = String.format("
    %.6f", decimalValue);
                            if (decimalString.endsWith(".000000"))
    {
                                decimalString = decimalString.
    substring(0, decimalString.length() - 7); // Remove trailing
    ".000000"
                            }
                            // Add a '-' sign to the decimal
    representation if it was a negative octal number
                            if (isNegative) {
                                decimalString = "-" + decimalString
    ;
                            }
                            resultnumberalsystem.setText(
    decimalString);
                            // Calculate and display the
    Hexadecimal number
                            String hexValue = convertDecimalToHex(
    decimalValue);
                            // Add a '-' sign to the hexadecimal
    representation if it was a negative octal number
                            if (isNegative) {
                                hexValue = "-" + hexValue;
                            }
                            resultcomplement.setText(hexValue);
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }

                    // Decimal to Octal conversion
                    else if (selectedFromIndex == 0 &&
    selectedToIndex == 2) {
                        try {
                            double decimalValue = Double.
    parseDouble(input);
                            // Check if the decimal value is
    negative
                            boolean isNegative = false;
                            if (decimalValue < 0) {
                                isNegative = true;
                                // Make it positive for processing
                                decimalValue = Math.abs(
    decimalValue);
                            }
                            // Separate the integer and fractional
    parts
                            int integerPart = (int) decimalValue;
                            double fractionalPart = decimalValue -
```

```
    integerPart;
                            // Convert the integer part to octal
                            StringBuilder octalIntegerPart = new
    StringBuilder();
                            while (integerPart > 0) {
                                int remainder = integerPart % 8;
                                octalIntegerPart.insert(0,
    remainder);
                                integerPart /= 8;
                            }
                            // If the integer part is zero, the
    octal representation is "0"
                            if (octalIntegerPart.length() == 0) {
                                octalIntegerPart.append("0");
                            }
                            // Convert the fractional part to octal
                            StringBuilder octalFractionalPart = new
     StringBuilder();
                            for (int i = 0; i < 6; i++) {
                                fractionalPart *= 8;
                                int digit = (int) fractionalPart;
                                octalFractionalPart.append(digit);
                                fractionalPart -= digit;
                            }
                            // Combine the integer and fractional
    parts
                            StringBuilder octalValue = new
    StringBuilder(octalIntegerPart.toString());
                            if (octalFractionalPart.length() > 0) {
                                octalValue.append('.').append(
    octalFractionalPart.toString());
                            }
                            // Add a '-' sign to the octal
    representation if it was a negative decimal number
                            if (isNegative) {
                                octalValue.insert(0, '-');
                            }
                            resultnumberalsystem.setText(octalValue
    .toString());
                            // Calculate and display the
    Hexadecimal number
                            String hexValue = convertDecimalToHex(
    decimalValue);
                            // Add a '-' sign to the hexadecimal
    representation if it was a negative decimal number
                            if (isNegative) {
                                hexValue = "-" + hexValue;
                            }
                            resultcomplement.setText(hexValue);
                        } catch (NumberFormatException ex) {
                            resultnumberalsystem.setText("N/A");
                            resultcomplement.setText("N/A");
                        }
                    }
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
```

```
            }
        });

        Reset.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Enternumber.setText("");
                resultnumberalsystem.setText("");
                resultcomplement.setText("");
            }
        });

        Swap.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

                int selectedFromIndex = fromcbb.getSelectedIndex();
                int selectedToIndex = tocbb.getSelectedIndex();
                fromcbb.setSelectedIndex(selectedToIndex);
                tocbb.setSelectedIndex(selectedFromIndex);
                // Update the UI components based on the swapped
    values
                updateHeaderLabel(headerLabel);
                updateEnternumber(LEnternumber);
                updateresultnumberalsystem(Lresultnumberalsystem);
                updateresultcomplement(Lresultcomplement);

            }
        });
    }

    // Custom function to convert decimal to octal
    private static String convertDecimalToOctal(double decimalValue
    ) {
        StringBuilder result = new StringBuilder();
        int intValue = (int) decimalValue;
        double fractionalPart = decimalValue - intValue;
        result.append(Integer.toOctalString(intValue));
        if (fractionalPart > 0) {
            result.append('.');
            for (int i = 0; i < 12; i++) {
                fractionalPart *= 8;
                int digit = (int) fractionalPart;
                result.append(digit);
                fractionalPart -= digit;
            }
        }
        return result.toString();
    }

    // Custom function to convert decimal to hexadecimal
    private static String convertDecimalToHex(double decimalValue)
    {
        String hex = "0123456789ABCDEF";
        StringBuilder result = new StringBuilder(" ");
        int intValue = (int) decimalValue;
        double fractionalPart = decimalValue - intValue;
```

```
796         result.append(Integer.toHexString(intValue));
797         if (fractionalPart > 0) {
798             result.append('.');
799             for (int i = 0; i < 12; i++) {
800                 fractionalPart *= 16;
801                 int digit = (int) fractionalPart;
802                 result.append(hex.charAt(digit));
803                 fractionalPart -= digit;
804             }
805         }
806         return result.toString().toUpperCase();
807     }
808 
809     private void updateHeaderLabel(JLabel headerLabel) {
810 
811         String selectedFrom = fromcbb.getSelectedItem().toString();
812         String selectedTo = tocbb.getSelectedItem().toString();
813         String conversionMode = selectedFrom + " to " + selectedTo
    + " converter";
814         headerLabel.setText(conversionMode);
815     }
816 
817     private void updateresultnumberalsystem(JLabel
    resultnumberalsystem) {
818         String selectedTo = tocbb.getSelectedItem().toString();
819         String conversionMode = selectedTo + " number ";
820         resultnumberalsystem.setText(conversionMode);
821     }
822 
823     private void updateresultcomplement(JLabel lresultcomplement) {
824         String selectedFrom = fromcbb.getSelectedItem().toString();
825         String selectedTo = tocbb.getSelectedItem().toString();
826         // Swap selectedFrom and selectedTo based on the specific
    conversions
827         if (selectedFrom.equals("Binary") && selectedTo.equals("
    Decimal")) {
828             lresultcomplement.setText("Decimal from signed 2's
    complement");
829         } else if (selectedFrom.equals("Decimal") && selectedTo.
    equals("Binary")) {
830             lresultcomplement.setText("Binary signed 2's complement
    ");
831         } else if (selectedFrom.equals("Binary") && selectedTo.
    equals("Octal")) {
832             lresultcomplement.setText("Decimal number");
833         } else if (selectedFrom.equals("Octal") && selectedTo.
    equals("Binary")) {
834             lresultcomplement.setText("Decimal number");
835         } else if (selectedFrom.equals("Binary") && selectedTo.
    equals("Hexadecimal")) {
836             lresultcomplement.setText("Decimal number");
837         } else if (selectedFrom.equals("Hexadecimal") && selectedTo
    .equals("Binary")) {
838             lresultcomplement.setText("Decimal number");
839         } else if (selectedFrom.equals("Decimal") && selectedTo.
    equals("Hexadecimal")) {
840             lresultcomplement.setText("Hex signed 2's complement");
```

```
841         } else if (selectedFrom.equals("Hexadecimal") && selectedTo
    .equals("Decimal")) {
842             lresultcomplement.setText("Decimal from signed 2's
    complement");
843         } else if (selectedFrom.equals("Decimal") && selectedTo.
    equals("Octal")) {
844             lresultcomplement.setText("Hex number");
845         } else if (selectedFrom.equals("Octal") && selectedTo.
    equals("Decimal")) {
846             lresultcomplement.setText("Hex number");
847         } else if (selectedFrom.equals("Octal") && selectedTo.
    equals("Hexadecimal")) {
848             lresultcomplement.setText("Decimal number");
849         } else if (selectedFrom.equals("Hexadecimal") && selectedTo
    .equals("Octal")) {
850             lresultcomplement.setText("Decimal number");
851         } else {
852             lresultcomplement.setText("Result Complement Label");
853         }
854     }
855
856     private void updateEnternumber(JLabel Enternumber) {
857         String selectedFrom = fromcbb.getSelectedItem().toString();
858         String selectedTo = tocbb.getSelectedItem().toString();
859         String conversionMode = "Enter " + selectedFrom + " number
    ";
860         Enternumber.setText(conversionMode);
861     }
862 }
```

Listing 1: ConverterNumber Java Code