

```

1 package Project1_Convertersnumber;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7 import java.math.BigDecimal;
8 public class Jnumberconvert extends JFrame {
9     private JComboBox<String> fromcbb;
10    private JComboBox<String> tocbb;
11    private JButton Convert;
12    private JButton Reset;
13    private JButton Swap;
14    private JTextField Enternumber;
15    private JTextField resultnumberalsystem;
16    private JTextField resultcomplement;
17    private static final String[] BASES = {"Decimal", "Binary", "
Octal", "Hexadecimal"};
18    private static final int[] RADIX_VALUES = {10, 2, 8, 16};
19
20    public Jnumberconvert() {
21
22        setTitle("Number System Converter");
23        setSize(580, 400); // Slightly larger dimensions
24        setDefaultCloseOperation(EXIT_ON_CLOSE);
25        setLocationRelativeTo(null);
26
27        JPanel panel = new JPanel();
28        panel.setBackground(new Color(0xFAFAD2));
29        panel.setLayout(null);
30
31        // Form
32        fromcbb = new JComboBox<>(BASES);
33        fromcbb.setBounds(30, 70, 220, 30); // Larger dimensions
34        JLabel fromLabel = new JLabel("From:");
35        fromLabel.setBounds(30, 40, 100, 20); // Adjusted position
36        and size
37        panel.add(fromLabel);
38        panel.add(fromcbb);
39
40        // To
41        tocbb = new JComboBox<>(BASES);
42        tocbb.setBounds(320, 70, 220, 30); // Larger dimensions
43        JLabel toLabel = new JLabel("To:");
44        toLabel.setBounds(320, 40, 100, 20); // Adjusted position
45        and size
46        panel.add(toLabel);
47        panel.add(tocbb);
48
49        // Initialize fromcbb and tocbb indices to reflect "Decimal
50        to Hexadecimal" conversion
51        fromcbb.setSelectedIndex(0); // Decimal
52        tocbb.setSelectedIndex(3); // Hexadecimal
53
54        // Header label
55        JLabel headerLabel = new JLabel();
56        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     resultnumeralsystem = new JTextField();
89     resultnumeralsystem.setEditable(false);
90     resultnumeralsystem.setBounds(30, 250, 510, 30); // Larger
dimensions
91     panel.add(resultnumeralsystem);
92
93     // Result number in the selected system JLabel
94     JLabel Lresultnumeralsystem = new JLabel();
95     Lresultnumeralsystem.setBounds(30, 220, 510, 30); //
Larger dimensions
96     updateresultnumeralsystem(Lresultnumeralsystem);
97     panel.add(Lresultnumeralsystem);
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

```

```

150     .toString();
151         }
152         if (isNegative) {
153             hexValue = "-" + hexValue;
154         }
155         resultnumberalsystem.setText(hexValue);
156         resultcomplement.setText("N/A");
157     } catch (Exception ex) {
158         throw new RuntimeException(ex);
159     }
160
161     // Hexadecimal to Decimal
162     else if (selectedFromIndex == 3 &&
selectedToIndex == 0) {
163         try {
164             String hexValue = input.toUpperCase();
165             boolean isNegative = false;
166             if (hexValue.startsWith("-")) {
167                 hexValue = hexValue.substring(1);
168                 isNegative = true;
169             }
170             int dotIndex = hexValue.indexOf('.');
171             String hexIntegerPart;
172             String hexFractionalPart;
173             if (dotIndex != -1) {
174                 hexIntegerPart = hexValue.substring
(0, dotIndex);
175                 hexFractionalPart = hexValue.
substring(dotIndex + 1);
176             } else {
177                 hexIntegerPart = hexValue;
178                 hexFractionalPart = "";
179             }
180             int decimalIntegerPart = Integer.
parseInt(hexIntegerPart, 16);
181             double decimalFractionalPart = 0.0;
182             if (!hexFractionalPart.isEmpty()) {
183                 int fractionalHex = Integer.
parseInt(hexFractionalPart, 16);
184                 decimalFractionalPart =
fractionalHex / Math.pow(16, hexFractionalPart.length());
185             }
186             double decimalResult =
decimalIntegerPart + decimalFractionalPart;
187             if (isNegative) {
188                 resultnumberalsystem.setText(String
.format("%.10f", -decimalResult));
189             } else {
190                 resultnumberalsystem.setText(String
.format("%.10f", decimalResult));
191             }
192             resultcomplement.setText("N/A");
193         } catch (Exception ex) {
194             throw new RuntimeException(ex);
195         }
196     }

```

```

197         // Binary to Decimal
198         else if (selectedFromIndex == 1 &&
199 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
207 processing
208                     binaryValue = binaryValue.substring
209 (1);
210                 }
211                 String[] parts = binaryValue.split("\\.
212 ");
213                 // Convert the integer part to decimal
214                 int decimalIntegerPart = Integer.
215 parseInt(parts[0], 2);
216                 double decimalFractionalPart = 0.0;
217                 // Convert the fractional part to
218 decimal
219                 if (parts.length > 1 && !parts[1].
220 isEmpty()) {
221                     String fractionalPart = parts[1];
222                     for (int i = 0; i < fractionalPart.
223 length(); i++) {
224                         int digit = Character.
225 getNumericValue(fractionalPart.charAt(i));
226                         decimalFractionalPart += digit
227 / Math.pow(2, i + 1);
228                     }
229                 }
230                 double decimalResult =
231 decimalIntegerPart + decimalFractionalPart;
232                 if (isNegative) {
233                     decimalResult = -decimalResult;
234                 }
235                 resultnumberalsystem.setText(Double.
236 toString(decimalResult));
237                 resultcomplement.setText("N/A");
238             } catch (NumberFormatException ex) {
239                 resultnumberalsystem.setText("Invalid
240 input");
241             }
242         }
243     }
244
245     // Decimal to Binary
246     else if (selectedFromIndex == 0 &&
247 selectedToIndex == 1) {
248         try {
249             String decimalValueStr = input;
250             // Parse the decimal value
251             double decimalValue = Double.
252 parseDouble(decimalValueStr);
253             // 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         resultnumeralsystem.setText(
binaryValue.toString());
270         // Display "N/A" for Binary signed 2's
complement
271         resultcomplement.setText("N/A");
272     } catch (NumberFormatException ex) {
273         resultnumeralsystem.setText("Invalid
input");
274     }
275 }
276
277 // Binary to Hexadecimal conversion

```

```

278         else if (selectedFromIndex == 1 &&
279             selectedToIndex == 3) {
280             try {
281                 String binaryValue = input;
282                 // Check if the binary value is
283                 negative
284                 boolean isNegative = false;
285                 if (binaryValue.startsWith("-")) {
286                     isNegative = true;
287                     // Remove the negative sign for
288                     processing
289                     binaryValue = binaryValue.substring
290                     (1);
291                 }
292                 // Split the binary number into integer
293                 and fractional parts
294                 String[] parts = binaryValue.split("\\.
295                 ");
296                 String binaryIntegerPart = parts[0];
297                 String binaryFractionalPart = (parts.
298                 length > 1) ? parts[1] : "";
299                 // Pad the fractional part with zeros
300                 to have a length multiple of 4
301                 while (binaryFractionalPart.length() %
302                 4 != 0) {
303                     binaryFractionalPart += "0";
304                 }
305                 // Initialize variables for the
306                 hexadecimal parts
307                 String hexIntegerPart = "";
308                 String hexFractionalPart = "";
309                 // Convert the integer part to
310                 hexadecimal
311                 if (!binaryIntegerPart.isEmpty()) {
312                     hexIntegerPart = Integer.
313                     toHexString(Integer.parseInt(binaryIntegerPart, 2)).toUpperCase
314                     ();
315                 }
316                 // Convert the fractional part to
317                 hexadecimal
318                 for (int i = 0; i <
319                 binaryFractionalPart.length(); i += 4) {
320                     String nibble =
321                     binaryFractionalPart.substring(i, i + 4);
322                     hexFractionalPart += Integer.
323                     toHexString(Integer.parseInt(nibble, 2)).toUpperCase();
324                 }
325                 // Add a '-' sign to both the
326                 hexadecimal and decimal representations if it was a negative
327                 binary number
328                 if (isNegative) {
329                     hexIntegerPart = "-" +
330                     hexIntegerPart;
331                 }
332                 // Combine integer and fractional
333                 hexadecimal parts
334                 String hexValue = hexIntegerPart;

```

```

314         if (!hexFractionalPart.isEmpty()) {
315             hexValue += "." + hexFractionalPart
;
316         }
317         resultnumeralsystem.setText(hexValue);
318         // Calculate the Decimal number
equivalent (integer + fractional)
319         double decimalEquivalent = 0.0;
320         if (!hexIntegerPart.isEmpty()) {
321             decimalEquivalent += Integer.
parseInt(binaryIntegerPart, 2);
322         }
323         if (!hexFractionalPart.isEmpty()) {
324             for (int i = 0; i <
hexFractionalPart.length(); i++) {
325                 char hexDigit =
hexFractionalPart.charAt(i);
326                 decimalEquivalent += Integer.
parseInt(String.valueOf(hexDigit), 16) / Math.pow(16, i + 1);
327             }
328         }
329         // Check if the binary value was
negative and adjust the decimal equivalent accordingly
330         if (isNegative) {
331             decimalEquivalent = -
decimalEquivalent;
332         }
333         // Display the Decimal number
equivalent (3 digits)
334         resultcomplement.setText(String.format(
"%0.3f", decimalEquivalent));
335     } catch (NumberFormatException ex) {
336         resultnumeralsystem.setText("N/A");
337         resultcomplement.setText("N/A");
338     }
339 }
340
341 // Hexadecimal to Binary conversion
342 else if (selectedFromIndex == 3 &&
selectedToIndex == 1) {
343     try {
344         String hexValue = input;
345         // Check if the hexadecimal value is
negative (if it starts with '-')
346         boolean isNegative = hexValue.
startsWith("-");
347         if (isNegative) {
348             // Remove the negative sign for
processing
349             hexValue = hexValue.substring(1);
350         }
351         // Split the hexadecimal number into
integer and fractional parts
352         String[] parts = hexValue.split("\\.");
353         String hexIntegerPart = parts[0];
354         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     resultnumeralsystem.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

```

```

391         if (!hexFractionalPart.isEmpty()) {
392             for (int i = 0; i <
hexFractionalPart.length(); i++) {
393                 char hexDigit =
hexFractionalPart.charAt(i);
394                 int digitValue = Integer.
parseInt(String.valueOf(hexDigit), 16);
395                 decimalEquivalent += digitValue
/ Math.pow(16, i + 1);
396             }
397         }
398         // Check if the hexadecimal value was
negative and adjust the decimal equivalent accordingly
399         if (isNegative) {
400             decimalEquivalent = -
decimalEquivalent;
401         }
402         // Display the Decimal number
equivalent (10 digits)
403         resultcomplement.setText(String.format(
"%0.10f", decimalEquivalent));
404     } catch (NumberFormatException ex) {
405         resultnumeralsystem.setText("N/A");
406         resultcomplement.setText("N/A");
407     }
408 }
409
410 // Binary to Octal conversion
411 else if (selectedFromIndex == 1 &&
selectedToIndex == 2) {
412     try {
413         String binaryValue = input;
414         // Check if the binary value is
negative
415         boolean isNegative = false;
416         if (binaryValue.startsWith("-")) {
417             isNegative = true;
418             // Remove the negative sign for
processing
419             binaryValue = binaryValue.substring
(1);
420         }
421         // Split the binary number into integer
and fractional parts
422         String[] parts = binaryValue.split("\\.
");
423         String binaryIntegerPart = parts[0];
424         String binaryFractionalPart = (parts.
length > 1) ? parts[1] : "";
425         // Pad the fractional part with zeros
to have a length multiple of 3
426         while (binaryFractionalPart.length() %
3 != 0) {
427             binaryFractionalPart += "0";
428         }
429         // Initialize variables for the octal
parts

```

```

430         String octalIntegerPart = "";
431         String octalFractionalPart = "";
432         // Convert the integer part to octal
433         if (!binaryIntegerPart.isEmpty()) {
434             octalIntegerPart = Integer.
toOctalString(Integer.parseInt(binaryIntegerPart, 2));
435         }
436         // Convert the fractional part to octal
437         for (int i = 0; i <
binaryFractionalPart.length(); i += 3) {
438             String nibble =
binaryFractionalPart.substring(i, i + 3);
439             octalFractionalPart += Integer.
toOctalString(Integer.parseInt(nibble, 2));
440         }
441         // Add a '-' sign to the octal
representation if it was a negative binary number
442         if (isNegative) {
443             octalIntegerPart = "-" +
octalIntegerPart;
444         }
445         // Combine integer and fractional octal
parts
446         String octalValue = octalIntegerPart;
447         if (!octalFractionalPart.isEmpty()) {
448             octalValue += "." +
octalFractionalPart;
449         }
450         resultnumeralsystem.setText(octalValue
);
451         // Calculate the Decimal number
equivalent (integer + fractional)
452         double decimalEquivalent = 0.0;
453         if (!octalIntegerPart.isEmpty()) {
454             decimalEquivalent += Integer.
parseInt(binaryIntegerPart, 2);
455         }
456         if (!octalFractionalPart.isEmpty()) {
457             for (int i = 0; i <
octalFractionalPart.length(); i++) {
458                 char octalDigit =
octalFractionalPart.charAt(i);
459                 decimalEquivalent += Integer.
parseInt(String.valueOf(octalDigit), 8) / Math.pow(8, i + 1);
460             }
461         }
462         // Check if the binary value was
negative and adjust the decimal equivalent accordingly
463         if (isNegative) {
464             decimalEquivalent = -
decimalEquivalent;
465         }
466         // Display the Decimal number
equivalent (3 digits)
467         resultcomplement.setText(String.format(
"%0.3f", decimalEquivalent));
468     } catch (NumberFormatException ex) {

```

```

469         resultnumeralsystem.setText("N/A");
470         resultcomplement.setText("N/A");
471     }
472 }
473
474     // Octal to Binary conversion
475     else if (selectedFromIndex == 2 &&
selectedToIndex == 1) {
476         try {
477             String octalValue = input;
478             // Check if the octal value is negative
479             boolean isNegative = false;
480             if (octalValue.startsWith("-")) {
481                 isNegative = true;
482                 // Remove the negative sign for
processing
483                 octalValue = octalValue.substring
(1);
484             }
485             // Split the octal number into integer
and fractional parts
486             String[] parts = octalValue.split("\\.");
487             String octalIntegerPart = parts[0];
488             String octalFractionalPart = (parts.
length > 1) ? parts[1] : "";
489             // Initialize variables for the binary
parts
490             String binaryIntegerPart = "";
491             String binaryFractionalPart = "";
492             // Convert the integer part to binary
493             if (!octalIntegerPart.isEmpty()) {
494                 binaryIntegerPart = Integer.
toBinaryString(Integer.parseInt(octalIntegerPart, 8));
495             }
496             // Convert the fractional part to
binary
497             for (int i = 0; i < octalFractionalPart
.length(); i++) {
498                 char octalDigit =
octalFractionalPart.charAt(i);
499                 String binaryNibble = Integer.
toBinaryString(Integer.parseInt(String.valueOf(octalDigit), 8))
;
500                 // Ensure that each binary nibble
has 3 digits
501                 while (binaryNibble.length() < 3) {
502                     binaryNibble = "0" +
binaryNibble;
503                 }
504                 binaryFractionalPart +=
binaryNibble;
505             }
506             // Combine integer and fractional
binary parts
507             String binaryValue = binaryIntegerPart;
508             if (!binaryFractionalPart.isEmpty()) {

```

```

509         binaryValue += "." +
binaryFractionalPart;
510     }
511     // Display the Binary number with a
negative sign if applicable
512     resultnumeralsystem.setText(isNegative
? "-" + binaryValue : binaryValue);
513     // Calculate the Decimal number
equivalent (integer + fractional)
514     double decimalEquivalent = 0.0;
515     if (!binaryIntegerPart.isEmpty()) {
516         decimalEquivalent += Integer.
parseInt(binaryIntegerPart, 2);
517     }
518     if (!binaryFractionalPart.isEmpty()) {
519         for (int i = 0; i <
binaryFractionalPart.length(); i++) {
520             char binaryDigit =
binaryFractionalPart.charAt(i);
521             decimalEquivalent += Integer.
parseInt(String.valueOf(binaryDigit), 2) / Math.pow(2, i + 1);
522         }
523     }
524     // Display the Decimal number with a
negative sign if applicable
525     resultcomplement.setText(isNegative ? "-"
+ decimalEquivalent : String.valueOf(decimalEquivalent));
526     } catch (NumberFormatException ex) {
527         resultnumeralsystem.setText("N/A");
528         resultcomplement.setText("N/A");
529     }
530 }
531
532 // Octal to Hexadecimal conversion
533 else if (selectedFromIndex == 2 &&
selectedToIndex == 3) {
534     try {
535         String octalValue = input;
536         // Check if the octal value is negative
537         boolean isNegative = false;
538         if (octalValue.startsWith("-")) {
539             isNegative = true;
540             // Remove the negative sign for
processing
541             octalValue = octalValue.substring
(1);
542         }
543         // Split the octal number into integer
and fractional parts
544         String[] parts = octalValue.split("\\.");
545         String octalIntegerPart = parts[0];
546         String octalFractionalPart = (parts.
length > 1) ? parts[1] : "0"; // Default to "0" if no
fractional part
547         // Convert the octal value to decimal
548         int decimalIntegerPart = Integer.

```

```

549     parseInt(octalIntegerPart, 8);
        // Convert the fractional part to
550     decimal
        double decimalFractionalPart = 0.0;
551     for (int i = 0; i < octalFractionalPart
        .length(); i++) {
552         char octalDigit =
        octalFractionalPart.charAt(i);
553         int digitValue = Character.
        getNumericValue(octalDigit);
554         decimalFractionalPart += digitValue
        / Math.pow(8, i + 1);
555     }
556     // Combine the integer and fractional
    parts
557     double decimalValue =
    decimalIntegerPart + decimalFractionalPart;
558     // Convert the decimal value to
    hexadecimal using a custom function
559     String hexValue = convertDecimalToHex(
    decimalValue);
560     // Add a '-' sign to the hexadecimal
    representation if it was a negative octal number
561     if (isNegative) {
562         hexValue = "-" + hexValue;
563     }
564     // Display the Hexadecimal number with
    a negative sign if applicable
565     resultnumeralsystem.setText(hexValue);
566     // Format the Decimal answer to match
    the desired format
567     String decimalString = String.format(
    "%.6f", decimalValue);
568     if (decimalString.endsWith(".000000"))
    {
569         decimalString = decimalString.
        substring(0, decimalString.length() - 7); // Remove trailing
        ".000000"
570     }
571     // Display the Decimal number
    equivalent with a negative sign if applicable
572     resultcomplement.setText(isNegative ? "-"
    + decimalString : decimalString);
573     } catch (NumberFormatException ex) {
574         resultnumeralsystem.setText("N/A");
575         resultcomplement.setText("N/A");
576     }
577 }
578
579 // Hexadecimal to Octal conversion
580 else if (selectedFromIndex == 3 &&
    selectedToIndex == 2) {
581     try {
582         String hexValue = input;
583         // Check if the hex value is negative
584         boolean isNegative = false;
585         if (hexValue.startsWith("-")) {

```

```

586         isNegative = true;
587         // Remove the negative sign for
processing
588         hexValue = hexValue.substring(1);
589     }
590     // Remove the "0x" prefix if present
591     if (hexValue.startsWith("0x") ||
hexValue.startsWith("0X")) {
592         hexValue = hexValue.substring(2);
593     }
594     // Split the hex number into integer
and fractional parts
595     String[] parts = hexValue.split("\\.");
596     String hexIntegerPart = parts[0];
597     String hexFractionalPart = (parts.
length > 1) ? parts[1] : "0"; // Default to "0" if no
fractional part
598     // Convert the hex value to decimal
599     double decimalIntegerPart = Integer.
parseInt(hexIntegerPart, 16);
600     // Convert the fractional part to
decimal
601     double decimalFractionalPart = 0.0;
602     for (int i = 0; i < hexFractionalPart.
length(); i++) {
603         char hexDigit = hexFractionalPart.
charAt(i);
604         int digitValue = Character.digit(
hexDigit, 16);
605         decimalFractionalPart += digitValue
/ Math.pow(16, i + 1);
606     }
607     // Combine the integer and fractional
parts
608     double decimalValue =
decimalIntegerPart + decimalFractionalPart;
609     // Convert the decimal value to octal
610     String octalValue =
convertDecimalToOctal(decimalValue);
611     // Add a '-' sign to the octal
representation if it was a negative hex number
612     if (isNegative) {
613         octalValue = "-" + octalValue;
614     }
615     // Display the Octal number with a
negative sign if applicable
616     resultnumeralsystem.setText(octalValue
);
617     // Format the Decimal answer to match
the desired format
618     String decimalString = String.format("
%.9f", decimalValue);
619     if (decimalString.endsWith(".000000000"
)) {
620         decimalString = decimalString.
substring(0, decimalString.length() - 10); // Remove trailing
".000000000"

```

```

621     }
622     // Display the Decimal number
equivalent with a negative sign if applicable
623     resultcomplement.setText(isNegative ? "-" + decimalString : decimalString);
624     } catch (NumberFormatException ex) {
625         resultnumeralsystem.setText("N/A");
626         resultcomplement.setText("N/A");
627     }
628 }
629
630 // Octal to Decimal conversion
631 else if (selectedFromIndex == 2 &&
selectedToIndex == 0) {
632     try {
633         String octalValue = input;
634         // Check if the octal value is negative
635         boolean isNegative = false;
636         if (octalValue.startsWith("-")) {
637             isNegative = true;
638             // Remove the negative sign for
processing
639             octalValue = octalValue.substring
(1);
640         }
641         // Split the octal number into integer
and fractional parts
642         String[] parts = octalValue.split("\\.");
643         String octalIntegerPart = parts[0];
644         String octalFractionalPart = (parts.
length > 1) ? parts[1] : "0"; // Default to "0" if no
fractional part
645         // Convert the octal integer part to
decimal
646         double decimalIntegerPart = 0.0;
647         for (int i = octalIntegerPart.length()
- 1; i >= 0; i--) {
648             char octalDigit = octalIntegerPart.
charAt(i);
649             int digitValue = Character.
getNumericValue(octalDigit);
650             decimalIntegerPart += digitValue *
Math.pow(8, octalIntegerPart.length() - i - 1);
651         }
652         // Convert the octal fractional part to
decimal
653         double decimalFractionalPart = 0.0;
654         for (int i = 0; i < octalFractionalPart
.length(); i++) {
655             char octalDigit =
octalFractionalPart.charAt(i);
656             int digitValue = Character.
getNumericValue(octalDigit);
657             decimalFractionalPart += digitValue
/ Math.pow(8, i + 1);
658         }

```



```

659                                     // Combine the integer and fractional
        parts
660                                     double decimalValue =
        decimalIntegerPart + decimalFractionalPart;
661                                     // Display the Decimal number with a
        negative sign if applicable
662                                     String decimalString = String.format("
        %.6f", decimalValue);
663                                     if (decimalString.endsWith(".000000"))
        {
664                                         decimalString = decimalString.
        substring(0, decimalString.length() - 7); // Remove trailing
        ".000000"
665                                     }
666                                     // Add a '-' sign to the decimal
        representation if it was a negative octal number
667                                     if (isNegative) {
668                                         decimalString = "-" + decimalString
        ;
669                                     }
670                                     resultnumeralsystem.setText(
        decimalString);
671                                     // Calculate and display the
        Hexadecimal number
672                                     String hexValue = convertDecimalToHex(
        decimalValue);
673                                     // Add a '-' sign to the hexadecimal
        representation if it was a negative octal number
674                                     if (isNegative) {
675                                         hexValue = "-" + hexValue;
676                                     }
677                                     resultcomplement.setText(hexValue);
678                                     } catch (NumberFormatException ex) {
679                                         resultnumeralsystem.setText("N/A");
680                                         resultcomplement.setText("N/A");
681                                     }
682                                 }
683
684                                     // Decimal to Octal conversion
685                                     else if (selectedFromIndex == 0 &&
        selectedToIndex == 2) {
686                                         try {
687                                             double decimalValue = Double.
        parseDouble(input);
688                                             // Check if the decimal value is
        negative
689                                             boolean isNegative = false;
690                                             if (decimalValue < 0) {
691                                                 isNegative = true;
692                                                 // Make it positive for processing
693                                                 decimalValue = Math.abs(
        decimalValue);
694                                             }
695                                             // Separate the integer and fractional
        parts
696                                             int integerPart = (int) decimalValue;
697                                             double fractionalPart = decimalValue -

```

```

integerPart;
698                                     // Convert the integer part to octal
699                                     StringBuilder octalIntegerPart = new
StringBuilder();
700                                     while (integerPart > 0) {
701                                         int remainder = integerPart % 8;
702                                         octalIntegerPart.insert(0,
remainder);
703                                         integerPart /= 8;
704                                     }
705                                     // If the integer part is zero, the
octal representation is "0"
706                                     if (octalIntegerPart.length() == 0) {
707                                         octalIntegerPart.append("0");
708                                     }
709                                     // Convert the fractional part to octal
710                                     StringBuilder octalFractionalPart = new
StringBuilder();
711                                     for (int i = 0; i < 6; i++) {
712                                         fractionalPart *= 8;
713                                         int digit = (int) fractionalPart;
714                                         octalFractionalPart.append(digit);
715                                         fractionalPart -= digit;
716                                     }
717                                     // Combine the integer and fractional
parts
718                                     StringBuilder octalValue = new
StringBuilder(octalIntegerPart.toString());
719                                     if (octalFractionalPart.length() > 0) {
720                                         octalValue.append('.').append(
octalFractionalPart.toString());
721                                     }
722                                     // Add a '-' sign to the octal
representation if it was a negative decimal number
723                                     if (isNegative) {
724                                         octalValue.insert(0, '-');
725                                     }
726                                     resultnumeralsystem.setText(octalValue
.toString());
727                                     // Calculate and display the
Hexadecimal number
728                                     String hexValue = convertDecimalToHex(
decimalValue);
729                                     // Add a '-' sign to the hexadecimal
representation if it was a negative decimal number
730                                     if (isNegative) {
731                                         hexValue = "-" + hexValue;
732                                     }
733                                     resultcomplement.setText(hexValue);
734                                     } catch (NumberFormatException ex) {
735                                         resultnumeralsystem.setText("N/A");
736                                         resultcomplement.setText("N/A");
737                                     }
738                                 }
739                                 } catch (Exception ex) {
740                                     ex.printStackTrace();
741                                 }

```

```

742     }
743 });
744
745 Reset.addActionListener(new ActionListener() {
746     @Override
747     public void actionPerformed(ActionEvent e) {
748         Enternumber.setText("");
749         resultnumberalsystem.setText("");
750         resultcomplement.setText("");
751     }
752 });
753
754 Swap.addActionListener(new ActionListener() {
755     @Override
756     public void actionPerformed(ActionEvent e) {
757
758         int selectedFromIndex = fromcbb.getSelectedIndex();
759         int selectedToIndex = tocbb.getSelectedIndex();
760         fromcbb.setSelectedIndex(selectedToIndex);
761         tocbb.setSelectedIndex(selectedFromIndex);
762         // Update the UI components based on the swapped
values
763         updateHeaderLabel(headerLabel);
764         updateEnternumber(LEnternumber);
765         updateresultnumberalsystem(Lresultnumberalsystem);
766         updateresultcomplement(Lresultcomplement);
767
768     }
769 });
770 }
771
772 // Custom function to convert decimal to octal
773 private static String convertDecimalToOctal(double decimalValue
) {
774     StringBuilder result = new StringBuilder();
775     int intValue = (int) decimalValue;
776     double fractionalPart = decimalValue - intValue;
777     result.append(Integer.toOctalString(intValue));
778     if (fractionalPart > 0) {
779         result.append('.');
780         for (int i = 0; i < 12; i++) {
781             fractionalPart *= 8;
782             int digit = (int) fractionalPart;
783             result.append(digit);
784             fractionalPart -= digit;
785         }
786     }
787     return result.toString();
788 }
789
790 // Custom function to convert decimal to hexadecimal
791 private static String convertDecimalToHex(double decimalValue)
{
792     String hex = "0123456789ABCDEF";
793     StringBuilder result = new StringBuilder(" ");
794     int intValue = (int) decimalValue;
795     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 updateresultnumeralsystem(JLabel
resultnumeralsystem) {
818     String selectedTo = tocbb.getSelectedItem().toString();
819     String conversionMode = selectedTo + " number ";
820     resultnumeralsystem.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