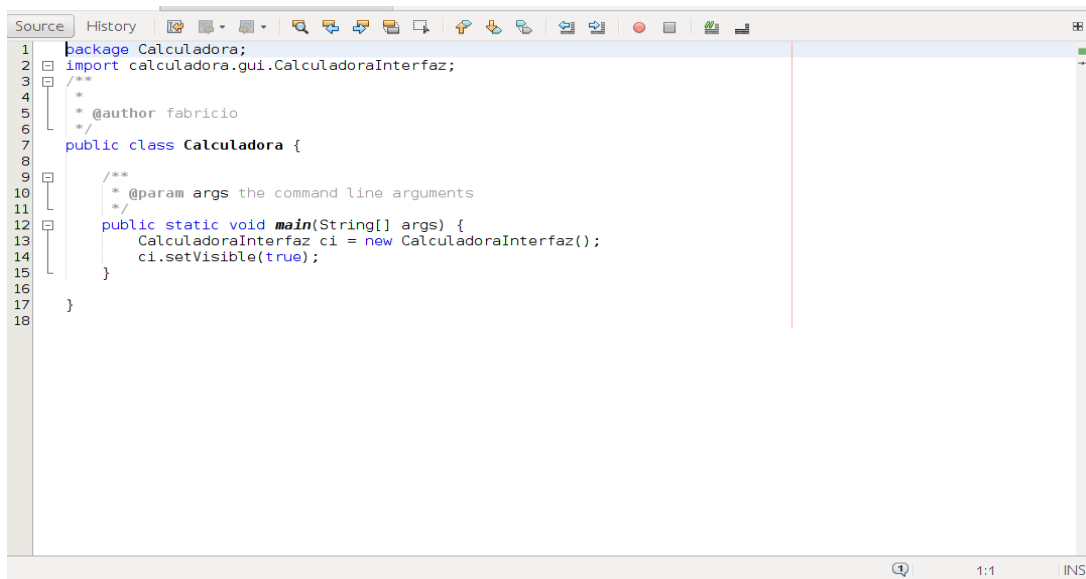


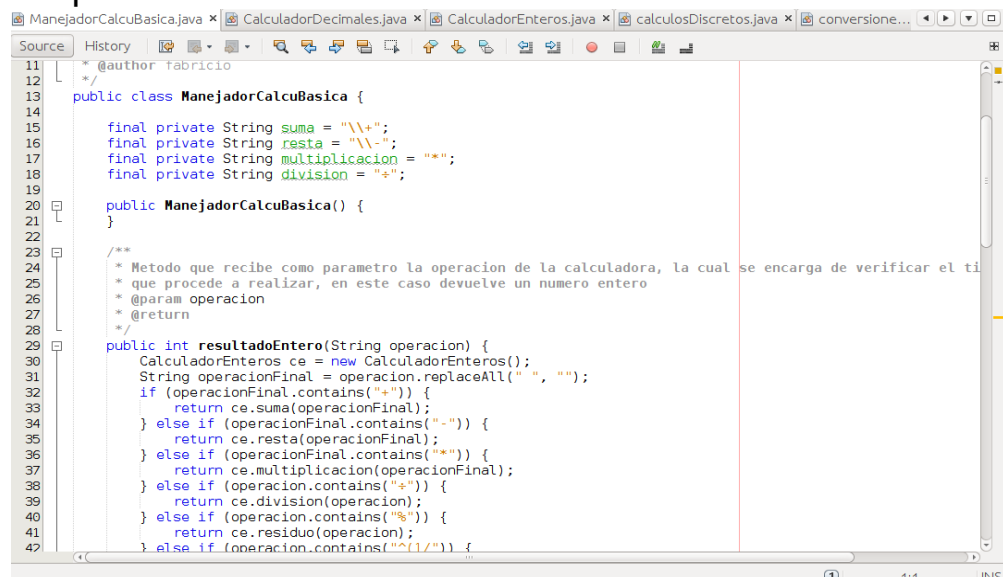
MANUAL TECNICO – CALCULADORA

1. Método “run” para que el proyecto pueda compilar e iniciar la ejecución del mismo.



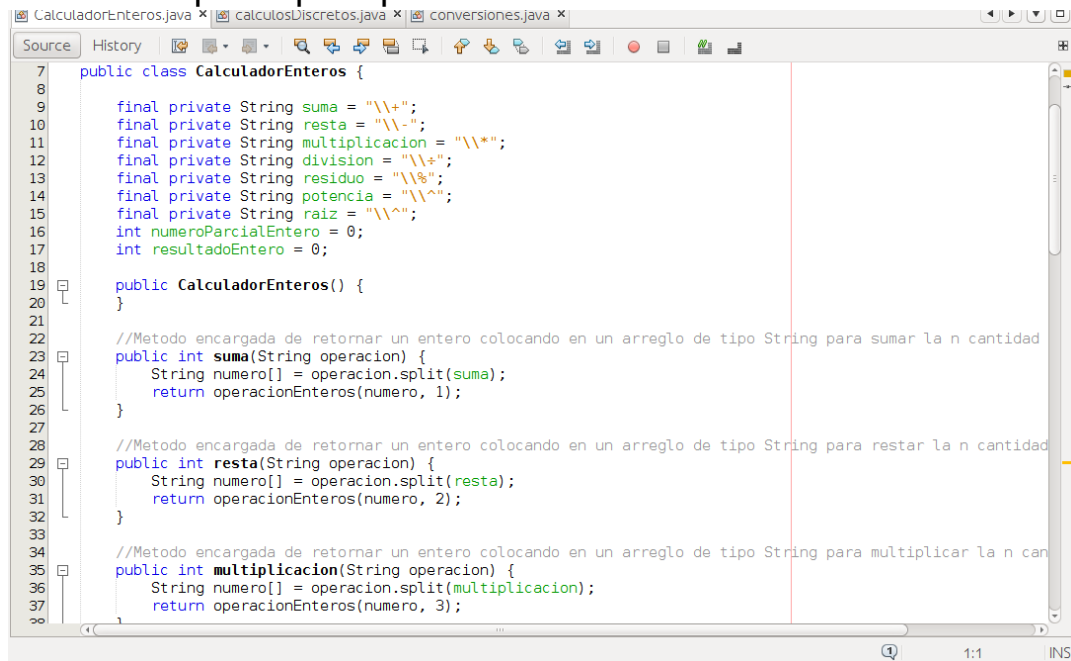
```
1 package Calculadora;
2 import calculadora.gui.CalculadoraInterfaz;
3
4 /**
5  * @author fabricio
6  */
7 public class Calculadora {
8
9     /**
10     * @param args the command line arguments
11     */
12     public static void main(String[] args) {
13         CalculadoraInterfaz ci = new CalculadoraInterfaz();
14         ci.setVisible(true);
15     }
16
17 }
18
```

2. Clase que maneja los tipos de resultados que puedan existir en las operaciones.



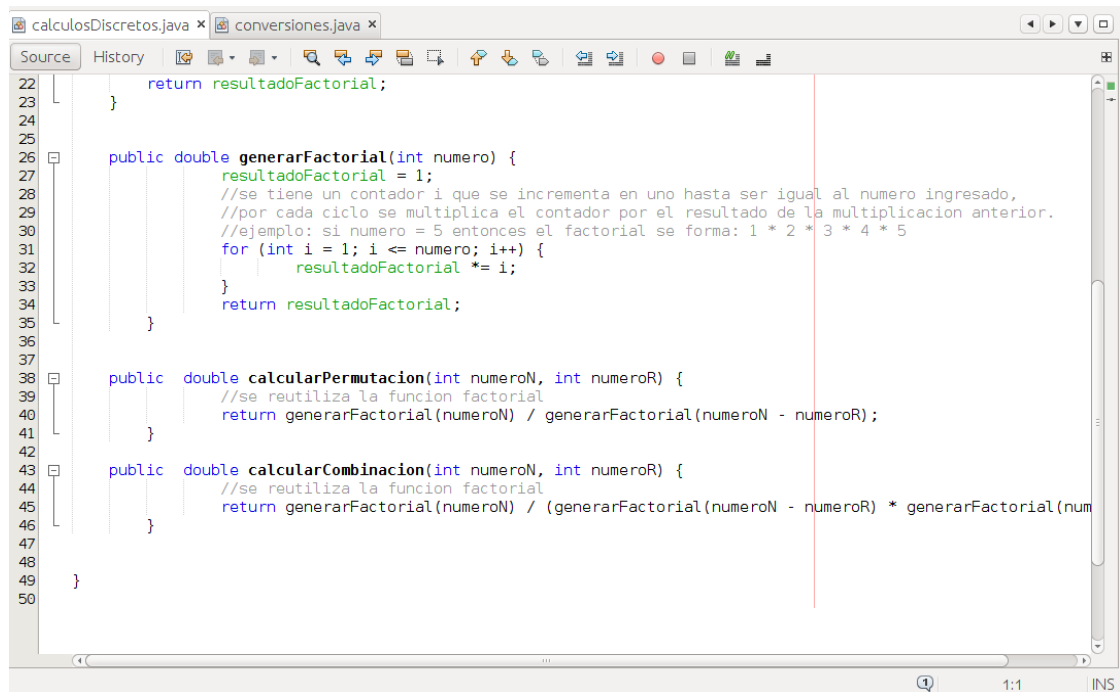
```
11 /**
12  * @author fabricio
13  */
14 public class ManejadorCalcuBasica {
15
16     final private String suma = "\\+";
17     final private String resta = "\\-";
18     final private String multiplicacion = "\\*";
19     final private String division = "\\÷";
20
21     public ManejadorCalcuBasica() {
22     }
23
24     /**
25     * Metodo que recibe como parametro la operacion de la calculadora, la cual se encarga de verificar el ti
26     * que procede a realizar, en este caso devuelve un numero entero
27     * @param operacion
28     * @return
29     */
30     public int resultadoEntero(String operacion) {
31         CalculadorEnteros ce = new CalculadorEnteros();
32         String operacionFinal = operacion.replaceAll(" ", "");
33         if (operacionFinal.contains("%")) {
34             return ce.suma(operacionFinal);
35         } else if (operacionFinal.contains("-")) {
36             return ce.resta(operacionFinal);
37         } else if (operacionFinal.contains("*")) {
38             return ce.multiplicacion(operacionFinal);
39         } else if (operacionFinal.contains("/")) {
40             return ce.division(operacion);
41         } else if (operacionFinal.contains("%")) {
42             return ce.residuo(operacion);
43         } else if (operacionFinal.contains("^")) {
44             return ce.potencia(operacion);
45         }
46     }
47 }
```

3. Clase donde se definen los métodos para las operaciones que el sistema requiere para poder funcionar como una calculadora.



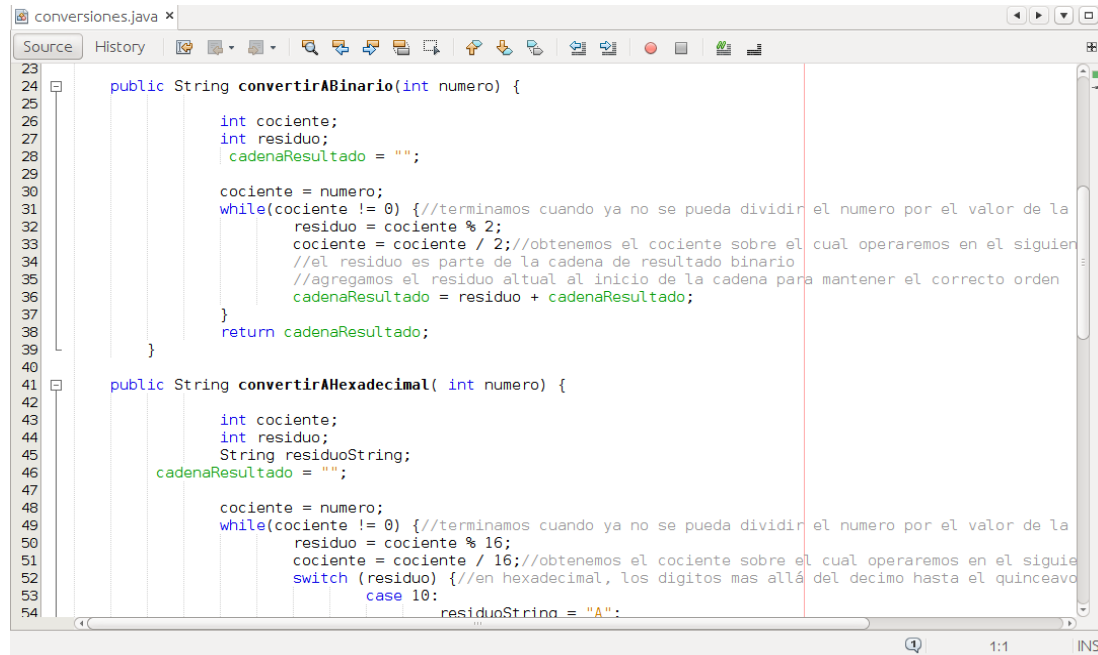
```
7 public class CalculadorEnteros {
8
9     final private String suma = "\\+";
10    final private String resta = "\\-";
11    final private String multiplicacion = "\\*";
12    final private String division = "\\+";
13    final private String residuo = "\\%";
14    final private String potencia = "\\^";
15    final private String raiz = "\\^";
16    int numeroParcialEntero = 0;
17    int resultadoEntero = 0;
18
19    public CalculadorEnteros() {
20    }
21
22    //Metodo encargada de retornar un entero colocando en un arreglo de tipo String para sumar la n cantidad
23    public int suma(String operacion) {
24        String numero[] = operacion.split(suma);
25        return operacionEnteros(numero, 1);
26    }
27
28    //Metodo encargada de retornar un entero colocando en un arreglo de tipo String para restar la n cantidad
29    public int resta(String operacion) {
30        String numero[] = operacion.split(resta);
31        return operacionEnteros(numero, 2);
32    }
33
34    //Metodo encargada de retornar un entero colocando en un arreglo de tipo String para multiplicar la n cantidad
35    public int multiplicacion(String operacion) {
36        String numero[] = operacion.split(multiplicacion);
37        return operacionEnteros(numero, 3);
38    }
39 }
```

4. Clase donde se define el método de cálculos discretos (permutación, factorial y combinación). En esta clase se utiliza la reutilización de código para poder optimizar las líneas del código.



```
22 }
23 return resultadoFactorial;
24 }
25
26 public double generarFactorial(int numero) {
27     resultadoFactorial = 1;
28     //se tiene un contador i que se incrementa en uno hasta ser igual al numero ingresado,
29     //por cada ciclo se multiplica el contador por el resultado de la multiplicacion anterior.
30     //ejemplo: si numero = 5 entonces el factorial se forma: 1 * 2 * 3 * 4 * 5
31     for (int i = 1; i <= numero; i++) {
32         resultadoFactorial *= i;
33     }
34     return resultadoFactorial;
35 }
36
37 public double calcularPermutacion(int numeroN, int numeroR) {
38     //se reutiliza la funcion factorial
39     return generarFactorial(numeroN) / generarFactorial(numeroN - numeroR);
40 }
41
42 public double calcularCombinacion(int numeroN, int numeroR) {
43     //se reutiliza la funcion factorial
44     return generarFactorial(numeroN) / (generarFactorial(numeroN - numeroR) * generarFactorial(numeroR));
45 }
46 }
47
48
49 }
50 }
```

5. Método donde se define el sub menú de conversiones numéricas (binario, octal y hexadecimal). Se utilizan vectores y cadenas para poder hacer la conversión.

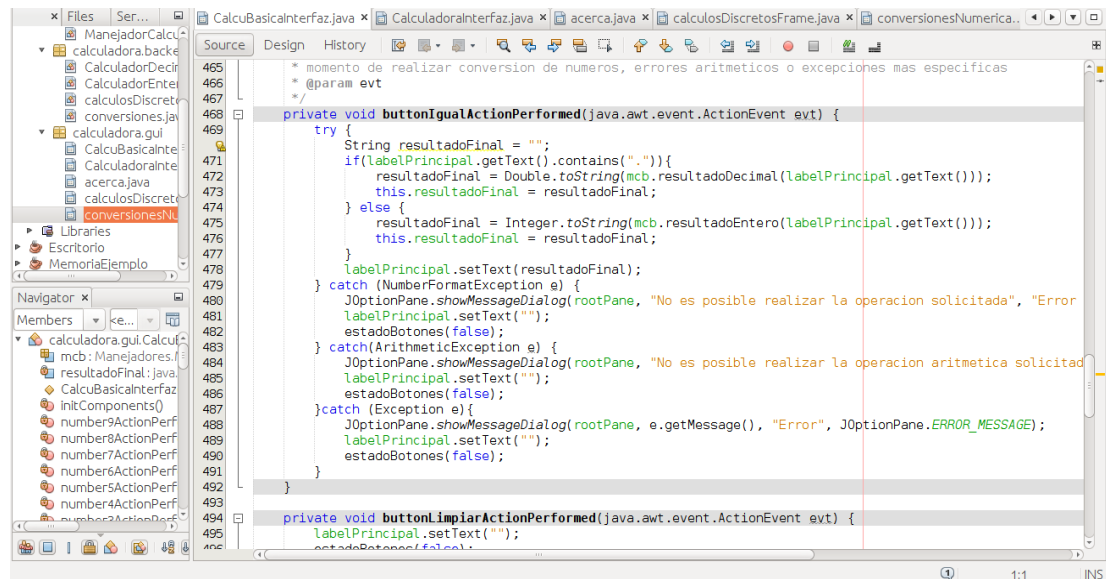


```
23
24 public String convertirABinario(int numero) {
25
26     int cociente;
27     int residuo;
28     cadenaResultado = "";
29
30     cociente = numero;
31     while(cociente != 0) { //terminamos cuando ya no se pueda dividir el numero por el valor de la
32         residuo = cociente % 2;
33         cociente = cociente / 2; //obtenemos el cociente sobre el cual operaremos en el siguiente
34         //el residuo es parte de la cadena de resultado binario
35         //agregamos el residuo actual al inicio de la cadena para mantener el correcto orden
36         cadenaResultado = residuo + cadenaResultado;
37     }
38     return cadenaResultado;
39 }
40
41 public String convertirAHexadecimal( int numero) {
42
43     int cociente;
44     int residuo;
45     String residuoString;
46     cadenaResultado = "";
47
48     cociente = numero;
49     while(cociente != 0) { //terminamos cuando ya no se pueda dividir el numero por el valor de la
50         residuo = cociente % 16;
51         cociente = cociente / 16; //obtenemos el cociente sobre el cual operaremos en el siguiente
52         switch (residuo) { //en hexadecimal, los digitos mas allá del decimo hasta el quinceavo
53             case 10:
54                 residuoString = "A";
55             case 11:
56                 residuoString = "B";
57             case 12:
58                 residuoString = "C";
59             case 13:
60                 residuoString = "D";
61             case 14:
62                 residuoString = "E";
63             case 15:
64                 residuoString = "F";
65             default:
66                 residuoString = String.valueOf(residuo);
67         }
68         cadenaResultado = residuoString + cadenaResultado;
69     }
70     return cadenaResultado;
71 }
```

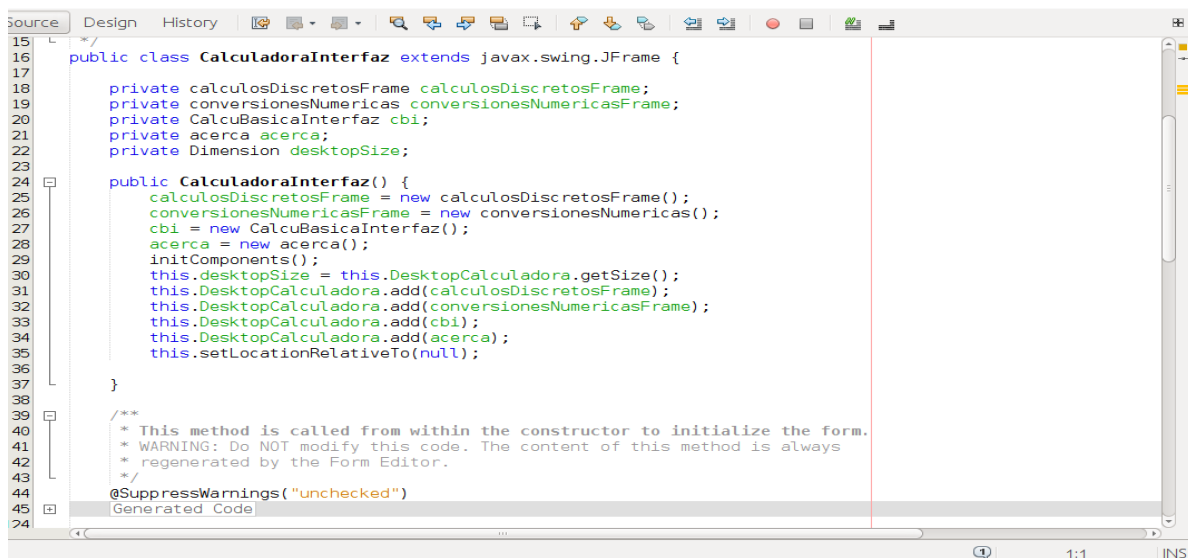
En todas las clases vistas con anterioridad se trabaja todo lo que se conoce como “backend”. Permitiendo esto una mayor flexibilidad y mejor manejo de código cuando este se requiera en el área de interfaz gráfica.

INTERFAZ GRAFICA

6. Se adjunta como quedan los botones y la llamada de los infernal frames en el frame inicial para poder tener conectado todo el sistema grafico que el usuario podrá visualizar. Se añaden excepción para que el sistema de la calculadora sea lo más eficaz posible.



```
465 * momento de realizar conversión de números, errores aritméticos o excepciones más específicas
466 * @param evt
467 */
468 private void buttonIgualActionPerformed(java.awt.event.ActionEvent evt) {
469     try {
470         String resultadoFinal = "";
471         if (labelPrincipal.getText().contains(".")) {
472             resultadoFinal = Double.toString(mcb.resultadoDecimal(labelPrincipal.getText()));
473             this.resultadoFinal = resultadoFinal;
474         } else {
475             resultadoFinal = Integer.toString(mcb.resultadoEntero(labelPrincipal.getText()));
476             this.resultadoFinal = resultadoFinal;
477         }
478         labelPrincipal.setText(resultadoFinal);
479     } catch (NumberFormatException e) {
480         JOptionPane.showMessageDialog(rootPane, "No es posible realizar la operación solicitada", "Error");
481         labelPrincipal.setText("");
482         estadoBotones(false);
483     } catch (ArithmeticException e) {
484         JOptionPane.showMessageDialog(rootPane, "No es posible realizar la operación aritmética solicitada", "Error");
485         labelPrincipal.setText("");
486         estadoBotones(false);
487     } catch (Exception e) {
488         JOptionPane.showMessageDialog(rootPane, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
489         labelPrincipal.setText("");
490         estadoBotones(false);
491     }
492 }
493
494 private void buttonLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
495     labelPrincipal.setText("");
496     estadoBotones(false);
497 }
```



```
15
16 public class CalculadoraInterfaz extends javax.swing.JFrame {
17
18     private calculosDiscretosFrame calculosDiscretosFrame;
19     private conversionesNumericas conversionesNumericasFrame;
20     private CalcuBasicaInterfaz cbi;
21     private acerca acerca;
22     private Dimension desktopSize;
23
24     public CalculadoraInterfaz() {
25         calculosDiscretosFrame = new calculosDiscretosFrame();
26         conversionesNumericasFrame = new conversionesNumericasFrame();
27         cbi = new CalcuBasicaInterfaz();
28         acerca = new acerca();
29         initComponents();
30         this.desktopSize = this.DesktopCalculadora.getSize();
31         this.DesktopCalculadora.add(calculosDiscretosFrame);
32         this.DesktopCalculadora.add(conversionesNumericasFrame);
33         this.DesktopCalculadora.add(cbi);
34         this.DesktopCalculadora.add(acerca);
35         this.setLocationRelativeTo(null);
36     }
37
38     /**
39      * This method is called from within the constructor to initialize the form.
40      * WARNING: Do NOT modify this code. The content of this method is always
41      * regenerated by the Form Editor.
42      */
43     @SuppressWarnings("unchecked")
44     // Generated Code
45 }
```

```
Source Design History
143 private void discretosButtonActionPerformed(java.awt.event.ActionEvent evt) {
144     switch (discretosComboBox.getSelectedIndex()) {
145     case 0:
146         try{
147             discretos.generarFactorial(Integer.parseInt(enteroNText.getText()));
148             resultadoText.setText(Integer.toString(discretos.getResultadoFactorial()));
149         }catch(NumberFormatException e){
150             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
151         }
152     }
153     break;
154     case 1:
155         try{
156             discretos.calcularPermutacion(Integer.parseInt(enteroNText.getText()), Integer.parseInt(e
157             resultadoText.setText(Integer.toString(discretos.getResultadoFactorial()));
158         }catch(NumberFormatException e){
159             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
160         }
161     }
162     break;
163     case 2:
164         try{
165             discretos.calcularCombinacion(Integer.parseInt(enteroNText.getText()), Integer.parseInt(e
166             resultadoText.setText(Integer.toString(discretos.getResultadoFactorial()));
167         }catch(NumberFormatException e){
168             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
169         }
170     }
171     break;
172     default:
173         break;
174 }
```

```
Source Design History
97 private void convertirButtonActionPerformed(java.awt.event.ActionEvent evt) {
98     switch (conversionesComboBox.getSelectedIndex()) {
99     case 0:
100         try{
101             conversiones.convertirABinario(Integer.parseInt(numeroText.getText()));
102             resultadoText.setText(conversiones.getResultado());
103         }catch(NumberFormatException e){
104             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
105         }
106     }
107     break;
108     case 1:
109         try{
110             conversiones.convertirAOctal(Integer.parseInt(numeroText.getText()));
111             resultadoText.setText(conversiones.getResultado());
112         }catch(NumberFormatException e){
113             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
114         }
115     }
116     break;
117     case 2:
118         try{
119             conversiones.convertirAHexadecimal(Integer.parseInt(numeroText.getText()));
120             resultadoText.setText(conversiones.getResultado());
121         }catch(NumberFormatException e){
122             JOptionPane.showMessageDialog(null,"ERROR\nVerifique el dato ingresado.", "Error de Típo")
123         }
124     }
125     break;
126     default:
127         break;
128 }
```