Proyecto “Calculadora”

Estructura de Datos

Profesora: Silvia Guardati

26 de septiembre 2024

Integrantes:

Julia Rojas Pereyra 214639

Valentina Pineda Barrón 215365

Natalia Quintana Guzmán 214468

David Alberto Estrada Becerril 213453

Alejandro Castillo Pérez Arce 213804

Uriel Antonio Chávez Sánchez 208148

Índice

Descripción del problema

I. Objetivos:

El proyecto se basa en la elaboración de una calculadora capaz de realizar operaciones aritméticas usando el lenguaje Java en el IDE NetBeans. Para esto se requiere de diferentes clases capaces de cumplir con la funcionalidad básica de una calculadora (suma, resta, división, multiplicación y potencia), y una interfaz gráfica amigable con el usuario y sencilla de comprender. Al realizar este proyecto buscamos adentrarnos aún más en la programación orientada a objetos y encontrar diferentes soluciones a problemas utilizando las estructuras de datos vistas hasta este punto en el curso, como son las pilas, además de fortalecer el trabajo en equipo para entregar un proyecto completo y diligente a las especificaciones establecidas. Buscamos hacer un proyecto amigable con el usuario final, que sea intuitivo y fácil de usar.

II. Requisitos

1. Realizar el código en equipo.
2. Verificar la validez de la operación escrita por el usuario (revisar paréntesis, operadores, números válidos, entre otros.)
3. Una vez se revisa la operación y se confirma su validez, convertir de notación infija a la notación postfija.
4. Evaluar la operación en notación postfija para regresar un resultado adecuado.
5. Mientras el usuario pica los botones mostrar simultáneamente en la pantalla los datos recibidos y seguido de esto con el botón “ = ” el resultado o en su defecto hacer saber que hay error.

III. Restricciones

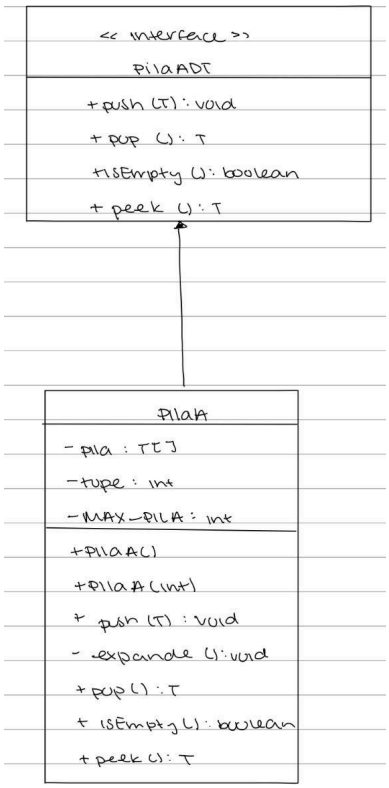
- Usar el IDE NetBeans
- Escribir el código en el lenguaje Java
- Entregar el proyecto completado el 26 de septiembre de 2024.

Debemos de tomar en cuenta que algunas restricciones que observamos al usar el programa es la sintaxis. A pesar de haber tenido la mayor precaución para escribir el método que revisa la sintaxis de las expresiones, es importante tomar en cuenta que el usuario tiene que ser muy precavido al ingresar las expresiones para que el programa no confunda ciertas expresiones, como los números negativos con el operador de resta.

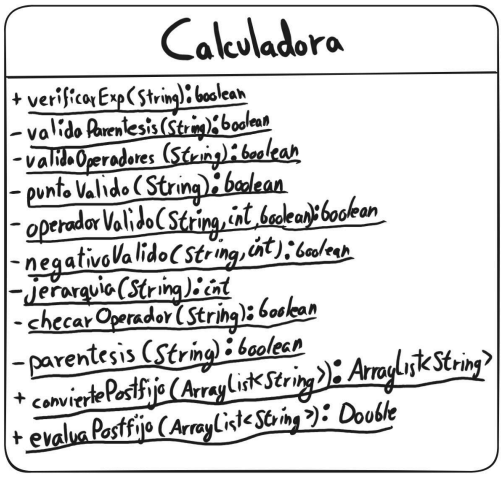
Solución diseñada

I. UML de las clases

A. Clase PilaADT y PilaA



B. Clase para revisar si la sintaxis de la operación ingresada en la calculadora está correcta y Clase para pasar de infijo a postfijo la operación ingresada



En la clase “Calculadora” tenemos tanto el método para revisar la sintaxis de la expresión ingresada como el método para pasar de infijo a postfijo.

D. Clase GUI

VistaCalculadora
+VistaCalculadora(); -initComponents(): void -borrarActionPerformed(evt: java.awt.event.ActionEvent): void -clearActionPerformedevt(evt: java.awt.event.ActionEvent): void -jButton7ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton8ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton9ActionPerformed(evt: java.awt.event.ActionEvent): void -divisionActionPerformed(evt: java.awt.event.ActionEvent): void -jButton4ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton5ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton6ActionPerformed(evt: java.awt.event.ActionEvent): void -multiplicacionActionPerformed(evt: java.awt.event.ActionEvent): void -restaActionPerformed(evt: java.awt.event.ActionEvent): void -jButton2ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton1ActionPerformed(evt: java.awt.event.ActionEvent): void -jButton3ActionPerformed(evt: java.awt.event.ActionEvent): void -negativoActionPerformed(evt: java.awt.event.ActionEvent): void -jButton0ActionPerformed(evt: java.awt.event.ActionEvent): void -puntoActionPerformed(evt: java.awt.event.ActionEvent): void -potenciaActionPerformed(evt: java.awt.event.ActionEvent): void -parentesisIzqActionPerformed(evt: java.awt.event.ActionEvent): void -parentesisDerActionPerformed(evt: java.awt.event.ActionEvent): void -igualActionPerformed(evt: java.awt.event.ActionEvent): void -sumaActionPerformed(evt: java.awt.event.ActionEvent): void +main(String[]args): void

II. Algoritmos Principales

1. **Validación de la expresión**: Se comprobó que cada expresión matemática ingresada fuera válida.
2. **Conversión de infijo a postfijo**: Una vez validada la expresión, se implementó la conversión de notación infija a notación postfija.
3. **Evaluación de la expresión postfija**: Finalmente, se evaluaron las expresiones en notación postfija para que calculará el resultado.
4. **Interfaces gráficas**: Permite al usuario tener una vista más amistosa y adaptable a una calculadora..

Pruebas:

I. Descripción de las pruebas

En cada clase del proyecto de la calculadora se realizaron pruebas exhaustivas para asegurarse de que el programa no fallará y todos los casos estuvieran cubiertos. Se validó la expresión matemática, se probó la conversión de infijo a postfijo y se verificó la correcta evaluación de las expresiones. Además, se implementaron excepciones `RuntimeException` para manejar errores sin que el programa se interrumpiera. Las pruebas abarcan desde casos simples hasta complejos para asegurar la fiabilidad del sistema.

I. Resultados obtenidos y análisis

Validación de la expresión: Se comprobó que cada expresión matemática ingresada fuera válida, verificando el correcto uso de operadores, operandos, y la correcta estructura de paréntesis, corchetes y llaves.

Conversión de infijo a postfijo: Una vez validada la expresión, se implementaron pruebas para asegurar que la conversión de notación infija a notación postfija se realizara de manera correcta. Para ello, se probaron diferentes combinaciones de expresiones, con variados niveles de complejidad.

Evaluación de la expresión postfija: Finalmente, se evaluaron las expresiones en notación postfija para garantizar que el cálculo resultante fuera correcto y preciso.

Limitaciones de la solución

I. Casos no contemplados

Aunque se realizaron pruebas exhaustivas, hay ciertos casos que no se contemplaron en la implementación actual del programa, tales como:

- Expresiones con caracteres no válidos o no numéricos (letras o símbolos especiales no matemáticos).
- Operaciones con números extremadamente grandes que puedan exceder los límites de los tipos de datos numéricos soportados.
- Expresiones que incluyen funciones matemáticas avanzadas como seno, coseno o logaritmos, ya que estas operaciones no se implementaron en esta versión del programa.
- Notaciones que mezclen infijo y postfijo en una misma expresión.

II. Pruebas JUnit4

Para garantizar la funcionalidad del programa, se realizaron pruebas automáticas con JUnit4, cubriendo los siguientes aspectos:

- **Validación de la expresión:** Se probaron expresiones válidas e inválidas, asegurando que se capturen errores en la estructura de la expresión.
- **Conversión de infijo a postfijo:** Se verificaron distintos tipos de expresiones (con diferentes niveles de operadores y paréntesis) para confirmar que la conversión se realice correctamente.
- **Evaluación de expresiones postfijas:** Se incluyeron pruebas para evaluar expresiones simples (como suma y resta) y complejas (como multiplicación/división con varios operadores).

Cada prueba fue diseñada para lanzar excepciones en casos donde hubiera errores, garantizando que el programa responda adecuadamente y no se interrumpa inesperadamente.

III. Restricciones para usar el programa

1. **Formato de la expresión:** La expresión debe estar correctamente escrita, utilizando solo operadores válidos (+, -, *, /) y números. No se admiten letras ni caracteres especiales.
2. **Paréntesis balanceados:** Si la expresión incluye paréntesis, deben estar bien balanceados. Expresiones con paréntesis, corchetes o llaves mal estructurado no serán procesadas.
3. **División por cero:** El programa no admite divisiones por cero. Si se detecta, el usuario recibirá una advertencia y la evaluación será interrumpida.
4. **Sin funciones avanzadas:** No se admiten funciones matemáticas avanzadas como logaritmos, senos, cosenos o exponentes en esta versión.

Posibles mejoras y conclusiones

I. Conclusiones sobre el desarrollo del proyecto

El proyecto fue una herramienta de mucha ayuda para mejorar y reforzar el conocimiento respecto a pilas y algoritmos lógicos en sí. Realmente fue un reto analizar primero las soluciones, desarrollar nuestras habilidades de solución de problemas y pensar en nuevas maneras de acaparar el mayor número de casos posibles. Además, a lo largo del proyecto pudimos experimentar lo que es checar códigos ajenos para dar buena retroalimentación y poder hacer los procesos más eficientes.

II. Conclusiones sobre la experiencia de trabajar en equipo

Consideramos como una ventaja el haber trabajado en equipo, ya que compartimos los mismos conocimientos previos y experiencias en programación. No obstante, enfrentamos algunos problemas, ya que nunca habíamos hecho un proyecto en conjunto y cada uno tenía ideas diferentes sobre cómo abordar ciertas partes del desarrollo. Esto generó desafíos en las primeras etapas del proyecto, pero también nos brindó la oportunidad de aprender a comunicarnos y llegar a acuerdos. Sin embargo, fue un reto que requirió múltiples reuniones, en este caso, la tecnología y GitHub nos ayudaron a seguir trabajando a pesar de no estar en el mismo lugar. Todos los integrantes se mostraron abiertos al diálogo y al acuerdo, lo que resultó en un proyecto exitoso.

Requerimos de varias sesiones por Zoom para discutir y ajustar los algoritmos y el código, buscando mejorar el programa de la calculadora con cada nueva prueba o caso que surgía. Finalmente, logramos una calculadora funcional que valida correctamente las expresiones, convierte de infijo a postfijo y evalúa con precisión, cumpliendo con nuestras expectativas de calidad y robustez.

Apéndice del código:

Clase RevisorSintaxis: Método para revisar la sintaxis de una operación matemática

verificarExp

```
/*
Metodo que comprueba la correcta sintaxis de la expresion dada por el usuario
@param String expresion
@return boolean. Dictamina si la expresión puede convertirse y evaluarse
Utiliza los métodos de validaParentesis y validaOperadores, igualmente, tienen el mismo paramétro que el principal y regresasn igualmente un boolean.
Necesita de un boolean verdadero de ambos métodos para el caso de éxito.
validaOperadores cuanta con más submetodos que se explicaran después
*/
 public static boolean verificarExp(String expresion) {
    boolean resp = false;
    boolean parentesis = false;
    boolean operadores = false;

    parentesis = validaParentesis(expresion);
    operadores = validaOperadores(expresion);

    if (parentesis && operadores) {
        resp = true;
    }

    return resp;
}
```

validaParentesis

```
    /*
    Metodo auxiliar privado a verificarExp(string) que comprueba el correcto uso de parentesis en la expresión
    @param String expresion
    @return boolean
    Los casos que proveemos para dictaminar que el uso de parentesis es correcto:
    1. Parentesis balanceados
    2. Parentesis no vacios
    Se iterara en la expresion hasta su final o hasta que una bandera nos determine que la expresión ya no es valida según los casos establecidos anteriormente.
    Se hace uso de una estructura tipo pila y de un switch.
    Para su éxito debe de encontrarese vacía la pila, ademas, de que la bandera debe mantenerse en true
    */
private static boolean validaParentesis(String expresion) {
    boolean balanceado;
    int i = 0;
    balanceado = true;
    PilaADT<Character> pilaAux = new PilaA<>();
    char c;

    while (i < expresion.length() && balanceado) {
        c = expresion.charAt(i);

        if (c == '(') {
            pilaAux.push(c);
        } else if (c == ')') {
            if (pilaAux.isEmpty() || i == 0 || expresion.charAt(i - 1) == '(') {


                balanceado = false;  // No hay '(' que le corresponde
            } else {
                pilaAux.pop(); //Sacamos su '(' correspondiente
            }
        }
        i++;
    }

    return balanceado && pilaAux.isEmpty();
}
```

validaOperadores

```
    /*
    Metodo auxiliar a verificarExp que comprueba el uso correcto de sus distintos operadores, y terminos decimales y negativos.
    @param String expresion
    @return boolean. Determinado por el valor de la variable boolean esValido
    Hara uso de los metodos auxiliares: puntoValido(String, int), operadorValido(String, int, boolean), negativoValido(String, int)
    Iterara por toda la expresion, hasta que termine o que la bandera esValido sea false
    */
private static boolean validaOperadores(String expresion) {
    boolean esValido = true;
    int i = 0;

    while (i < expresion.length() && esValido) {
        char c = expresion.charAt(i);

        switch (c) {
            case '.':  // Punto decimal
                if (!puntoValido(expresion, i))
                    esValido = false;
                break;

            case '+':
            case '-':
            case '*':
            case '/':
            case '^':  // Potencia
```

```
                if (!operadorValido(expresion, i, c == '/')) // Si es '/' se valida también la división entre 0
                    esValido = false;

                break;

            case '?': // Signo para numero negativo
                if (!negativoValido(expresion, i))
                    esValido = false;
                break;

            default:
                break;
        }

        i++;
    }
    return esValido;
}
```

puntoValido

```
/*
Metodo aux. a validaOperadores(String).
@param String expresion, int pos
@return boolean.
Verifica que el punto decimal esté rodeado por dígitos
*/
    private static boolean puntoValido(String expresion, int pos) {
        boolean digitoIzquierda, digitoDerecha;

        digitoIzquierda = pos > 0 && Character.isDigit(expresion.charAt(pos - 1));
        digitoDerecha = pos < expresion.length() - 1 && Character.isDigit(expresion.charAt(pos + 1));

        return digitoIzquierda && digitoDerecha;
    }
```

operadorValido

```
    /*
    Metodo aux. a validaOperadores(String).
    @param String expresion, int pos, boolean esDivision
    @return boolean.
    Verifica que los operadores sean validos, que esten entre parentesis o terminos
    */
    private static boolean operadorValido(String expresion, int pos, boolean esDivision) {
        // Verificar que haya un dígito o paréntesis a la izquierda y derecha del operador
        boolean validoIzq, validoDer;

        validoIzq = pos > 0 && (Character.isDigit(expresion.charAt(pos - 1)) || expresion.charAt(pos - 1) == ')');
        validoDer = pos < expresion.length() - 1 && (Character.isDigit(expresion.charAt(pos + 1)) || expresion.charAt(pos + 1) == '(' || expresion.charAt(pos + 1) == '?');

        // Si es una división, además de lo anterior, verificar si divide entre 0. Tener en cuenta casos donde directamente este dividendo con 0, pero aceptar casos como 5/0.00029
        if (esDivision && pos < expresion.length() - 1) {
            int j = pos + 1;

            while (j < expresion.length() && (Character.isDigit(expresion.charAt(j)) || expresion.charAt(j) == '.' || expresion.charAt(j) == '?'))
                j++;

            if (expresion.charAt(j-1) == '0')
                validoDer = false;
        }

        return validoIzq && validoDer;
    }
```

negativoValido

```java
    /*
Metodo aux. a validaOperadores(String).
@param String expresion, int pos
@return boolean.
Verifica que el termino negativo sea valido, osea, que sea != 0
Tener en cuenta los casos como 0000014, ya que sigue siendo valido, parecido al caso de la division
*/
private static boolean negativoValido(String expresion, int pos) {
    boolean validaIzq, validaDer;

    validaIzq = pos == 0
            || expresion.charAt(pos - 1) == '/'
            || expresion.charAt(pos - 1) == '+'
            || expresion.charAt(pos - 1) == '-'
            || expresion.charAt(pos - 1) == '*'
            || expresion.charAt(pos - 1) == '^';

    validaDer = pos < expresion.length() - 1 && (Character.isDigit(expresion.charAt(pos + 1))
            || (expresion.charAt(pos + 1) == '.' && pos + 2 < expresion.length() && Character.isDigit(expresion.charAt(pos + 2))));

    return validaIzq && validaDer;
}
```

jerarquia

```java
/**
 * Método auxiliar para asignar el valor de los operadores y obtener una
 * jerarquía
 * <pre>
 * Utiliza un switch para identificar el tipo de operador
 * Cada caso asigna un valor distinto de acuerdo a la jerarquía de las operaciones
 * <pre>
 * @param operador String
 * @return int valor asignado al operador
 */
private static int jerarquia(String operador) {
    int jerarquia = 0;
    switch (operador) {
        case "*": //Tanto la multiplicacion como la división son de la misma jerarquía
            jerarquia = 3;
            break;
        case "/":
            jerarquia = 3;
            break;
        case "+": //La suma y la resta son de la misma jerarquía
            jerarquia = 2;
            break;
        case "-":
            jerarquia = 2;
            break;
        case "^": //La potencia tiene la máxima jerarquía después del paréntesis
            jerarquia = 4;
            break;
        case "(": //Menor jerarquía para que lo saque al momento de encontrar su contraparte
            jerarquia = 1;
            break;
        default:
            jerarquia = 0;
            break;
    }
    return jerarquia;
}
```

checarOperador

```java
/**
 * Método auxiliar para revisar si el String es operador
 *
 * @param c String
 * @return boolean <ul>
 * <li> true: si es un operador "+","-","*","/","^"
 * <li> false: si no es un operador
 * <ul>
 */

private static boolean checarOperador(String c) {
    boolean resp = false;
    if (c.equals("+") || c.equals("-") || c.equals("*") || c.equals("/") || c.equals("^")) {
        resp = true;
    }
    return resp;
}
```

parentesis

```java
/**
 * Método auxiliar para revisar si el String es un paréntesis
 *
 * @param c String
 * @return boolean <ul>
 * <li> true: si el String es "(" o ")"
 * <li> false: si el String no es "(" o ")"
 * <ul>
 */
private static boolean parentesis(String c) {
    boolean resp = false;
    if (c.equals("(") || c.equals(")")) {
        resp = true;
    }
    return resp;
}
```

Clase Infijo a Postfijo: Método para convertir a postfijo y evaluarlo

conviertePosfijo

```java
/**
 * Método para convertir una expresión en String de infijo a postfijo
 *
 * @param infijo El ArrayList en el que está guardada la expresión en infijo
 * como escrita por el cliente
 * @return ArrayList de String: expresión convertida a postfijo
 * @see parentesis
 * @see checarOperador
 */

public static ArrayList<String> conviertePostfijo(ArrayList<String> infijo) {
    PilaA<String> pila = new PilaA(100);  //Pila en donde guarda los operadores
    ArrayList<String> postfijo = new ArrayList(); //ArrayList del resultado
    String c;
    for (int i = 0; i < infijo.size(); i++) {
        c = infijo.get(i);
        if (checarOperador(c)) { //Si el char es un operador revisa la jerarquia de pila.peek() y saca o mete de la pila
            while (!pila.isEmpty() && jerarquia(pila.peek()) >= jerarquia(c)) {
                postfijo.add(pila.pop());
            }
            pila.push(c);
        }
        if (!checarOperador(c) && !parentesis(c)) //Si no es operador o parentesis, agrega al ArrayList postfijo
        {
            postfijo.add(c);
        } else {
            if (c.equals("(")) //Agrega a la pila directo
            {
                pila.push(c);
            }
            if (c.equals(")")) {
                while (!pila.peek().equals("(")) //Saca todo de la pila hasta que no encuentre el parentesis abierto
                {
                    postfijo.add(pila.pop());
                }
                pila.pop();
            }
        }
    } //Una vez que se termino de revisar el ArrayList infijo, vacia la pila de operadores
    while (!pila.isEmpty()) {
        postfijo.add(pila.pop());
    }
    return postfijo;
}
```

evaluaPostfijo

```java
/**
 * Método para evaluar una expresión en postfijo
 * <pre>
 * Toma una expresión en postfijo y a través de pilas evalúa las operaciones
 * A través de un switch se identifica al operador y asigna la operación a realizar correspondiente
 * La única operación restringida es la división por cero a través de una bandera
 * <pre>
 * @param postfijo ArrayList de la expresión infija convertida a posfija
 * @return Double: resultado final de la operación ingresada por el usuario
 * @see checarOperador
 */

public static double evaluaPostfijo(ArrayList<String> postfijo) {

    PilaA<Double> pila = new PilaA(100);
    double n1,n2;
    int i = 0;
    boolean bandera = true;
    String c;
    while (i < postfijo.size() && bandera) {
        c = postfijo.get(i);
        if (!checarOperador(c)) {
            pila.push(Double.parseDouble(c));
        } else {
            n1 = pila.pop();
            n2 = pila.pop();
            switch (c) {
                case "+":
                    pila.push(n1 + n2);
                    break;
                case "-":
                    pila.push(n2 - n1);
                    break;
                case "/":
                    if (n1 == 0) {
                        // Si intentamos dividir por cero, en lugar de lanzar una excepción,
                        // devolvemos Double.NaN (Not-a-Number) para indicar un error en la operación.
                        return Double.NaN;
                    } else {
                        pila.push(n2 / n1);
                    }
                    break;
                case "^":
                    pila.push(Math.pow(n2, n1));
                    break;


                  default:
                     pila.push(n1 * n2);
                     break;
              }
          }
          i++;
      }
      return pila.pop();
  }
```

Clase Vista: La interfaz gráfica de la calculadora

```
/**
 * Creates new form VistaCalculadora2
 */
public VistaCalculadora() {
    initComponents();

}
```

initComponents

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

    campoTexto = new javax.swing.JTextField();
    borrar = new javax.swing.JButton();
    clear = new javax.swing.JButton();
    jButton7 = new javax.swing.JButton();
    jButton8 = new javax.swing.JButton();
    jButton9 = new javax.swing.JButton();
    division = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    jButton5 = new javax.swing.JButton();
    jButton6 = new javax.swing.JButton();
    multiplicacion = new javax.swing.JButton();
    resta = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton1 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    negativo = new javax.swing.JButton();
    jButton0 = new javax.swing.JButton();
    punto = new javax.swing.JButton();
    potencia = new javax.swing.JButton();
    parentesisIzq = new javax.swing.JButton();
    parentesisDer = new javax.swing.JButton();
    igual = new javax.swing.JButton();
    suma = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Calculadora");
    setBackground(new java.awt.Color(153, 204, 255));
    setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

    campoTexto.setEditable(false);
    campoTexto.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N

    borrar.setBackground(new java.awt.Color(243, 166, 53));
    borrar.setForeground(new java.awt.Color(255, 255, 255));
    borrar.setText("⌫");
    borrar.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
    borrar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            borrarActionPerformed(evt);
        }
```

```
    }
});

clear.setBackground(new java.awt.Color(243, 166, 53));
clear.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
clear.setForeground(new java.awt.Color(255, 255, 255));
clear.setText("C");
clear.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
clear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        clearActionPerformed(evt);
    }
});

jButton7.setBackground(new java.awt.Color(72, 161, 241));
jButton7.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton7.setForeground(new java.awt.Color(255, 255, 255));
jButton7.setText("7");
jButton7.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});

jButton8.setBackground(new java.awt.Color(72, 161, 241));
jButton8.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton8.setForeground(new java.awt.Color(255, 255, 255));
jButton8.setText("8");
jButton8.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton8ActionPerformed(evt);
    }
});

jButton9.setBackground(new java.awt.Color(72, 161, 241));
jButton9.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton9.setForeground(new java.awt.Color(255, 255, 255));
jButton9.setText("9");
jButton9.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton9ActionPerformed(evt);
    }
});
```

```
division.setBackground(new java.awt.Color(243, 166, 53));
division.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
division.setForeground(new java.awt.Color(255, 255, 255));
division.setText("/");
division.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
division.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        divisionActionPerformed(evt);
    }
});

jButton4.setBackground(new java.awt.Color(72, 161, 241));
jButton4.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton4.setForeground(new java.awt.Color(255, 255, 255));
jButton4.setText("4");
jButton4.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jButton5.setBackground(new java.awt.Color(72, 161, 241));
jButton5.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton5.setForeground(java.awt.Color.white);
jButton5.setText("5");
jButton5.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jButton6.setBackground(new java.awt.Color(72, 161, 241));
jButton6.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton6.setForeground(java.awt.Color.white);
jButton6.setText("6");
jButton6.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton6ActionPerformed(evt);
    }
});
```

```
multiplicacion.setBackground(new java.awt.Color(243, 166, 53));
multiplicacion.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
multiplicacion.setForeground(new java.awt.Color(255, 255, 255));
multiplicacion.setText("*");
multiplicacion.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
multiplicacion.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        multiplicacionActionPerformed(evt);
    }
});

resta.setBackground(new java.awt.Color(243, 166, 53));
resta.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
resta.setForeground(new java.awt.Color(255, 255, 255));
resta.setText("-");
resta.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
resta.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        restaActionPerformed(evt);
    }
});

jButton2.setBackground(new java.awt.Color(72, 161, 241));
jButton2.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton2.setForeground(java.awt.Color.white);
jButton2.setText("2");
jButton2.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton1.setBackground(new java.awt.Color(72, 161, 241));
jButton1.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton1.setForeground(java.awt.Color.white);
jButton1.setText("1");
jButton1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```

```
jButton3.setBackground(new java.awt.Color(72, 161, 241));
jButton3.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton3.setForeground(java.awt.Color.white);
jButton3.setText("3");
jButton3.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

negativo.setBackground(new java.awt.Color(72, 161, 241));
negativo.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
negativo.setForeground(new java.awt.Color(255, 255, 255));
negativo.setText("+/-");
negativo.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
negativo.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        negativoActionPerformed(evt);
    }
});

jButton0.setBackground(new java.awt.Color(72, 161, 241));
jButton0.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
jButton0.setForeground(new java.awt.Color(255, 255, 255));
jButton0.setText("0");
jButton0.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
jButton0.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton0ActionPerformed(evt);
    }
});

punto.setBackground(new java.awt.Color(72, 161, 241));
punto.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
punto.setForeground(new java.awt.Color(255, 255, 255));
punto.setText(".");
punto.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
punto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        puntoActionPerformed(evt);
    }
});
```

```
potencia.setBackground(new java.awt.Color(243, 166, 53));
potencia.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
potencia.setForeground(new java.awt.Color(255, 255, 255));
potencia.setText("^");
potencia.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
potencia.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        potenciaActionPerformed(evt);
    }
});

parentesisIzq.setBackground(new java.awt.Color(41, 206, 88));
parentesisIzq.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
parentesisIzq.setForeground(new java.awt.Color(255, 255, 255));
parentesisIzq.setText("(");
parentesisIzq.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
parentesisIzq.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        parentesisIzqActionPerformed(evt);
    }
});

parentesisDer.setBackground(new java.awt.Color(41, 206, 88));
parentesisDer.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
parentesisDer.setForeground(new java.awt.Color(255, 255, 255));
parentesisDer.setText(")");
parentesisDer.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
parentesisDer.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        parentesisDerActionPerformed(evt);
    }
});

igual.setBackground(new java.awt.Color(41, 206, 88));
igual.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
igual.setForeground(new java.awt.Color(255, 255, 255));
igual.setText("=");
igual.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
igual.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        igualActionPerformed(evt);
    }
});
```

```
suma.setBackground(new java.awt.Color(243, 166, 53));
suma.setFont(new java.awt.Font("Century Gothic", 1, 14)); // NOI18N
suma.setForeground(new java.awt.Color(255, 255, 255));
suma.setText("+");
suma.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
suma.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        sumaActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(campoTexto)
            .addGroup(layout.createSequentialGroup()
                .addComponent(parentesisIzq, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(parentesisDer, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(igual, javax.swing.GroupLayout.PREFERRED_SIZE, 142, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(multiplicacion, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(suma, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addComponent(negativo, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
```

```
                        .addComponent(jButton0, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(punto, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(resta, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(jButton7, javax.swing.GroupLayout.DEFAULT_SIZE, 62, Short.MAX_VALUE)
                            .addComponent(clear, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(potencia, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(division, javax.swing.GroupLayout.PREFERRED_SIZE, 62, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addComponent(borrar, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(31, 31, 31)
            .addComponent(campoTexto, javax.swing.GroupLayout.PREFERRED_SIZE, 88, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(borrar, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(potencia, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(clear, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(division, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton9, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(multiplicacion, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(suma, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton0, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(negativo, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(punto, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(resta, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(parentesisDer, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(parentesisIzq, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(igual, javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(13, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>//GEN-END:initComponents
```

borrarActionPerformed

```
/**
 * Elimina el último carácter del campo de texto.
 *
 * Si el campo de texto no está vacío, se elimina el último carácter de la cadena
 * actual. Si está vacío, no hace nada.
 *
 * @param evt El evento que dispara la acción.
 */
private void borrarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_borrarActionPerformed
    // TODO add your handling code here:
    String cadena = campoTexto.getText();
    if (!cadena.isEmpty()) {
    StringBuilder sb = new StringBuilder(cadena);
    sb.deleteCharAt(sb.length() - 1);
    campoTexto.setText(sb.toString());
    }
}//GEN-LAST:event_borrarActionPerformed
```

clearActionPerformed

```
/**
 * Limpia todo el contenido del campo de texto, dejando la calculadora en un estado inicial.
 *
 * Este método se utiliza para reiniciar el campo de entrada de la calculadora. Al ser llamado, borra
 * todo el contenido del campo de texto, lo que permite al usuario comenzar una nueva operación sin
 * necesidad de eliminar cada carácter de forma manual.
 *
 * Uso típico:
 * - Este método es comúnmente utilizado cuando se presiona el botón "C" (clear) en una calculadora,
 *   que borra cualquier expresión o número que haya sido ingresado.
 * @param evt El evento que dispara la acción.
 */
private void clearActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_clearActionPerformed
    // TODO add your handling code here:
    campoTexto.setText("");
}//GEN-LAST:event_clearActionPerformed
```

jButtonAction performed (uno por cada número del 0-9)

```
/**
 * Añade el número 0 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton0ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton0ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"0");
}//GEN-LAST:event_jButton0ActionPerformed
```

```java
/**
 * Añade el número 7 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton7ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"7");
}//GEN-LAST:event_jButton7ActionPerformed

/**
 * Añade el número 8 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton8ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"8");
}//GEN-LAST:event_jButton8ActionPerformed

/**
 * Añade el número 9 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton9ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"9");
}//GEN-LAST:event_jButton9ActionPerformed

/**
 * Añade el operador de división al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void divisionActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_divisionActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"/");
}//GEN-LAST:event_divisionActionPerformed
/**
 * Añade el número 4 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
```

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton4ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"4");
}//GEN-LAST:event_jButton4ActionPerformed

/**
 * Añade el número 5 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton5ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"5");
}//GEN-LAST:event_jButton5ActionPerformed

/**
 * Añade el número 6 al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton6ActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"6");
}//GEN-LAST:event_jButton6ActionPerformed
```

```
 /**
  * Añade el número 2 al campo de texto.
  *
  * @param evt El evento que dispara la acción.
  */
 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton2ActionPerformed
     // TODO add your handling code here:
     String texto=campoTexto.getText();
     campoTexto.setText(texto+"2");
 }//GEN-LAST:event_jButton2ActionPerformed

 /**
  * Añade el número 1 al campo de texto.
  *
  * @param evt El evento que dispara la acción.
  */
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton1ActionPerformed
     // TODO add your handling code here:
     String texto=campoTexto.getText();
     campoTexto.setText(texto+"1");
 }//GEN-LAST:event_jButton1ActionPerformed

 /**
  * Añade el número 3 al campo de texto.
  *
  * @param evt El evento que dispara la acción.
  */
 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_jButton3ActionPerformed
     // TODO add your handling code here:
     String texto=campoTexto.getText();
     campoTexto.setText(texto+"3");
 }//GEN-LAST:event_jButton3ActionPerformed
```

multiplicacionActionPerformed

```
/**
 * Añade el operador de multiplicación al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void multiplicacionActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_multiplicacionActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"*");
}//GEN-LAST:event_multiplicacionActionPerformed
```

restaActionPerformed

```
/**
 * Añade el operador de resta al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void restaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_restaActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"-");
}//GEN-LAST:event_restaActionPerformed
```

sumaActionPerformed

```
 /**
  * Añade el operador de suma al campo de texto.
  *
  * @param evt El evento que dispara la acción.
  */
 private void sumaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_sumaActionPerformed
     // TODO add your handling code here:
     String texto=campoTexto.getText();
     campoTexto.setText(texto+"+");
 }//GEN-LAST:event_sumaActionPerformed
```

negativoActionPerformed

```
/**
 * Añade el operador de negativo al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void negativoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_negativoActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"?");
}//GEN-LAST:event_negativoActionPerformed
```

puntoActionPerformed

```
/**
 * Añade un punto al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void puntoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_puntoActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+".");
}//GEN-LAST:event_puntoActionPerformed
```

potenciaActionPerformed

```
/**
 * Añade el operador de potencia al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void potenciaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_potenciaActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"^");
}//GEN-LAST:event_potenciaActionPerformed
```

parentesisIzqActionPerformed y parentesisDerActionPerformed

```
/**
 * Añade un paréntesis izquierdo al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void parentesisIzqActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_parentesisIzqActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+"(");
}//GEN-LAST:event_parentesisIzqActionPerformed

/**
 * Añade un paréntesis derecho al campo de texto.
 *
 * @param evt El evento que dispara la acción.
 */
private void parentesisDerActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_parentesisDerActionPerformed
    // TODO add your handling code here:
    String texto=campoTexto.getText();
    campoTexto.setText(texto+")");
}//GEN-LAST:event_parentesisDerActionPerformed
```

igualActionPerformed

```java
/**
 * Evalúa la expresión ingresada en el campo de texto y muestra el resultado.
 *
 * Este método sigue los siguientes pasos:
 * 1. Verifica que la expresión ingresada sea válida utilizando {@link Calculadora#verificarExp(String)}.
 * 2. Convierte la expresión infija a una lista en notación postfija usando {@link Calculadora#conviertePostfijo(List)}.
 * 3. Evalúa la expresión postfija usando {@link Calculadora#evaluaPostfijo(List)}.
 *
 * Detalles del proceso:
 * - Se convierte la cadena de la expresión en un {@link ArrayList} de tipo String, separando números y operadores.
 *   Esto prepara la expresión para pasarla de infija a postfija
 *
 * Casos considerados:
 * - Los números y operadores se extraen y almacenan en el ArrayList para su posterior evaluación.
 * - Si la expresión es correcta, se muestra el resultado en el campo de texto.
 *   En caso contrario, muestra "Syntax ERROR".
 *
 * @param evt El evento que dispara la acción.
 */
private void igualActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_igualActionPerformed
// TODO add your handling code here:
    String expresion=campoTexto.getText();
    if(Calculadora.verificarExp(expresion)){
        ArrayList<String> texto = new ArrayList<>();
        String numero="";
        for(int i=0;i<expresion.length();i++){
            char c=expresion.charAt(i);

            // Si el carácter es un dígito o un punto, es parte de un número
            if(Character.isDigit(c) || c=='.')
                numero+=c;
            else{
                // Si el número no está vacío, agrégalo a la lista
                if(!numero.equals("")){
                    texto.add(numero); // Agrega el número
                    numero=""; // Resetea la variable para el siguiente número
                }
                texto.add(c+""); // Agrega el operador o cualquier otro carácter
            }
        }
        if(!numero.equals(""))

            texto.add(numero); //En caso de que hubiera un número al final, lo añade
        double resultado=Calculadora.evaluaPostfijo(Calculadora.conviertePostfijo(texto));
        if(resultado==Double.NaN)
            campoTexto.setText("Syntax ERROR");
        else
            campoTexto.setText(resultado+"");

    } else
        campoTexto.setText("Syntax ERROR");

}//GEN-LAST:event_igualActionPerformed
```

main con try y catch de cada excepcion

```
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>
    try {
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (Exception ex) {
    java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VistaCalculadora().setVisible(true);
        }
```

```
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton borrar;
    private javax.swing.JTextField campoTexto;
    private javax.swing.JButton clear;
    private javax.swing.JButton division;
    private javax.swing.JButton igual;
    private javax.swing.JButton jButton0;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JButton multiplicacion;
    private javax.swing.JButton negativo;
    private javax.swing.JButton parentesisDer;
    private javax.swing.JButton parentesisIzq;
    private javax.swing.JButton potencia;
    private javax.swing.JButton punto;
    private javax.swing.JButton resta;
    private javax.swing.JButton suma;
    // End of variables declaration//GEN-END:variables
}
```

Clase PruebaCalculadora: Pruebas para la detección de errores

```java
package calculadora;

/*
 * @author DAVID ESTRADA // 2123453
 */

public class PruebaCalculadora {
    public static void main(String[] args) {
        // TODO code application logic here
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(VistaCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new VistaCalculadora().setVisible(true);
            }
```

```
/* Prueba Detección de errores
Calculadora calc1 = new Calculadora();

String expresion1 = "5+3.2*(7-4)(";
String expresion2 = "5/0";
String expresion3 = "(5+3*2)";
String expresion4 = "5+*3";
String expresion5 = "0.5+3";
String expresion6 = "5+3+()";
String expresion7 = "5.2+3.8";
String expresion8 = "5*(3+2)/2-1";
String expresion9 = "5+3*2)";
String expresion10 = "(5+(3*2))+3)";
String expresion11 = "5*(+3)"; // Esta mal
String expresion12 = "5/0.0"; // Esta mal

System.out.println(expresion1);
System.out.println(expresion2);
System.out.println(expresion3);
System.out.println(expresion4);
System.out.println(expresion5);
System.out.println(expresion6);
System.out.println(expresion7);
System.out.println(expresion8);
System.out.println(expresion9);
System.out.println(expresion10);
System.out.println(expresion11);
System.out.println(expresion12);


 if (calc1.verificarExp(expresion1))
    System.out.println("La expresion " + expresion1 + " es valida");
 else
    System.out.println("La expresion " + expresion1 + " es invalida");
if (calc1.verificarExp(expresion2))
    System.out.println("La expresion " + expresion2 + " es valida");
else
    System.out.println("La expresion " + expresion2 + " es invalida");

 if (calc1.verificarExp(expresion3))
    System.out.println("La expresion " + expresion3 + " es valida");
else
    System.out.println("La expresion " + expresion3 + " es invalida");
 if (calc1.verificarExp(expresion4))
    System.out.println("La expresion " + expresion4 + " es valida");
 else
    System.out.println("La expresion " + expresion4 + " es invalida");
 if (calc1.verificarExp(expresion5))
    System.out.println("La expresion " + expresion5 + " es valida");
```

```
        else
           System.out.println("La expresion " + expresion5 + " es invalida");
        if (calc1.verificarExp(expresion6))
           System.out.println("La expresion " + expresion6 + " es valida");
        else
           System.out.println("La expresion " + expresion6 + " es invalida");
        if (calc1.verificarExp(expresion7))
           System.out.println("La expresion " + expresion7 + " es valida");
        else
           System.out.println("La expresion " + expresion7+ " es invalida");
        if (calc1.verificarExp(expresion8))
           System.out.println("La expresion " + expresion8 + " es valida");
        else
           System.out.println("La expresion " + expresion8 + " es invalida");
        if (calc1.verificarExp(expresion9))
           System.out.println("La expresion " + expresion9 + " es valida");
        else
           System.out.println("La expresion " + expresion9 + " es invalida");
        if (calc1.verificarExp(expresion10))
           System.out.println("La expresion " + expresion10 + " es valida");
        else
           System.out.println("La expresion " + expresion10 + " es invalida");
        if (calc1.verificarExp(expresion11))
           System.out.println("La expresion " + expresion11 + " es valida");
        else
           System.out.println("La expresion " + expresion11 + " es invalida");
        if (calc1.verificarExp(expresion12))
           System.out.println("La expresion " + expresion12 + " es valida");
        else
           System.out.println("La expresion " + expresion12 + " es invalida");
       */

       /*
       PilaADT<String> pila = new PilaA<>();
       pila.push("-");
       pila.push("2");
       pila.push("^");
       pila.push("2");
       pila.push("/");


        pila.push("5");
        pila.push("2");
        System.out.println(pila);
        System.out.println(pila.peek());
*/
/*
        ConvertidorAPosFijo convertidor = new ConvertidorAPosFijo();

        String infijo1 = "3*4+2"; // Cadena infix a convertir a postfix
        String postfijo1 = convertidor.convierteAPostFijo(infijo1);
        System.out.println("Infijo: " + infijo1);
        System.out.println("Postfijo: " + postfijo1);

        String infijo2 = "(%3+4)^2-1"; // Cadena infix a convertir a postfix
        String postfijo2 = convertidor.convierteAPostFijo(infijo2);
        System.out.println("Infijo: " + infijo2);
        System.out.println("Postfijo: " + postfijo2);
*/
    }
```

Fuentes Consultadas

1. Documentación oficial de Java:
- URL: https://docs.oracle.com/javase/8/docs/

2. Algoritmo de conversión de infijo a postfijo:
- *GeeksforGeeks - Infix to Postfix Conversion*: https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/

3. Manejo de excepciones en Java:
- URL: https://www.baeldung.com/java-exceptions

4. GitHub para colaboración:
- URL: https://guides.github.com/