23/02/2023

Calculadora en Java con el uso de pilas y notación postfija

Integrantes del equipo:

Julián Hernández Alás

Rodrigo Demay Carranza

Robert Derek Bowden Barbero

Joshua Chaparro Sandoval

Óscar Larios Mancilla

Materia: Estructuras de Datos

Profesor: Silvia del Carmen Guardati Buemo

Índice

Descripción del problema

Las pilas son estructuras de datos de suma utilidad en una gran cantidad de aspectos. Como parte de un proyecto, nuestro equipo tuvo que integrar esta herramienta con el fin de programar una calculadora que pudiera checar una cierta entrada, convertirla de la notación infija correcta y traducirla a su equivalente en postfijo para evaluarla y mostrarla en una interfaz. Por lo tanto, los objetivos de este proyecto son los siguientes:

1. Hacer un programa en Java que, mediante el uso de pilas y la notación postfija, sea capaz de hacer operaciones básicas de calculadora (suma, resta, multiplicación, división y potencia)
2. Crear la interfaz correspondiente, para un uso sencillo y amigable
3. Administrar las tareas dentro de nuestro equipo para que todos los miembros estén integrados de forma dinámica

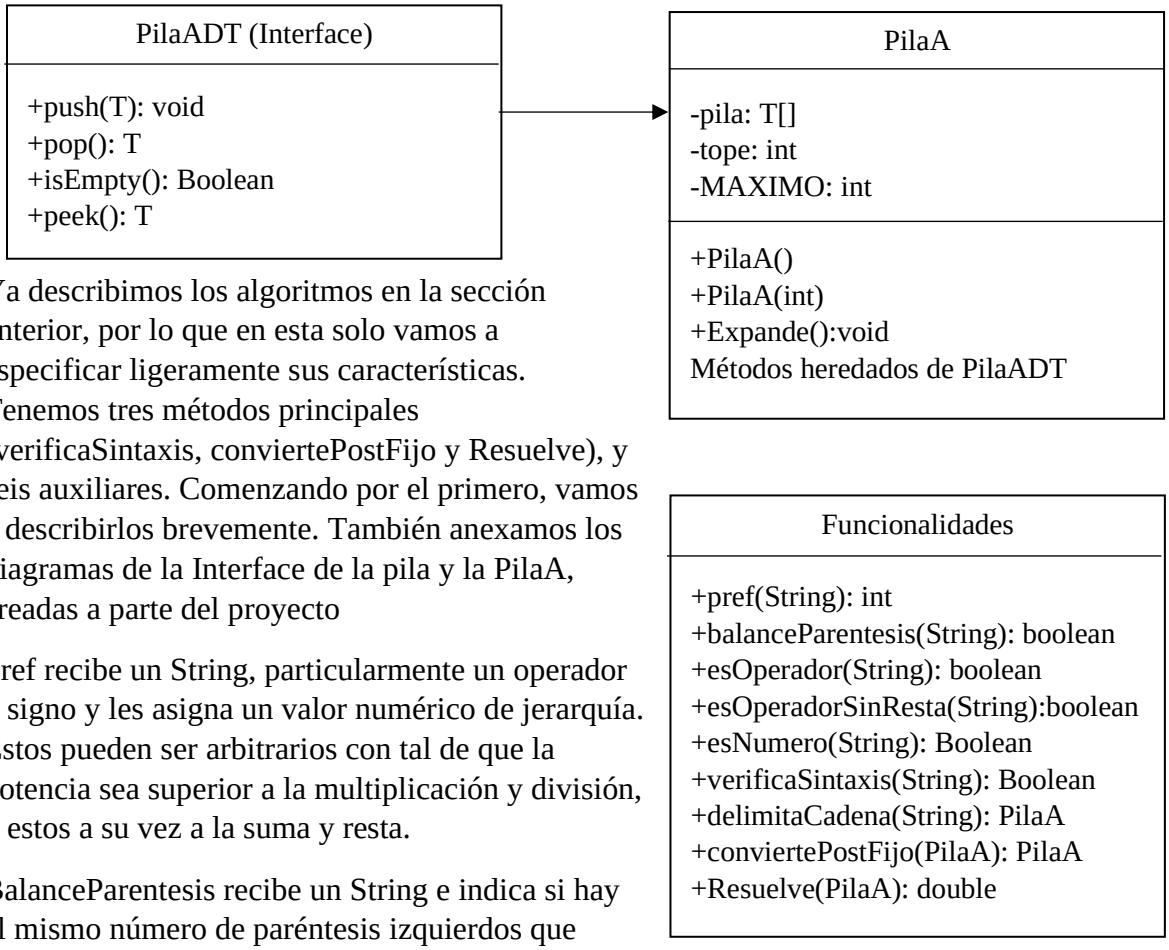
Particularmente, la solución debía cumplir una serie de características para ampliar su funcionamiento y estandarizar la solución en java. El código debía contemplar la implementación de números negativos, de varios dígitos y decimales, además de realizar sus contenidos exclusivamente en java. Particularmente, nuestro equipo decidió dividir el trabajo en tres métodos. El primero valida la sintaxis de la expresión infija, es decir, que posea paréntesis equilibrados, operadores válidos, jerarquía de operaciones, signos adyacentes compatibles (por ejemplo, 7+(6) es compatible, 7+*6 no), que el primer carácter fuera un operando o un paréntesis izquierdo y el último otro operando o un paréntesis derecho. Adicional a esto, había que tener en cuenta el uso del signo negativo, ya que se pueden formar números con este símbolo.

Por otro lado, los otros dos programas eran relativamente más sencillos, ya que solo requerían adaptar algoritmos bastante conocidos al lenguaje de programación correspondiente (java), por lo que su elaboración fue muy simple.

La interfaz que recibía y mostraba los datos tampoco presentó una gran dificultad. Solo necesitaba incluir los diez botones para cada dígito, uno por cada operador diferente (ocho, incluyendo el punto decimal), el botón para evaluar la expresión y otro para despejar el resultado.

Todo el código se escribió en Apache NetBeans 12. 6 con Java SE 17.0.1

Solución Diseñada



Ya describimos los algoritmos en la sección anterior, por lo que en esta solo vamos a especificar ligeramente sus características. Tenemos tres métodos principales (verificaSintaxis, conviertePostFijo y Resuelve), y seis auxiliares. Comenzando por el primero, vamos a describirlos brevemente. También anexamos los diagramas de la Interface de la pila y la PilaA, creadas a parte del proyecto

Pref recibe un String, particularmente un operador o signo y les asigna un valor numérico de jerarquía. Estos pueden ser arbitrarios con tal de que la potencia sea superior a la multiplicación y división, y estos a su vez a la suma y resta.

BalanceParentesis recibe un String e indica si hay el mismo número de paréntesis izquierdos que derechos y están ordenados correctamente, mostrando verdadero o falso dado el caso.

EsOperador solo indica si cierto String representa un símbolo de operación (+,-,*,/,^).

EsOperadorSinResta es el mismo algoritmo que el pasado, pero excluye la resta con el propósito de incluir los números negativos

EsNumero regresa verdadero o falso si la entrada representa un dígito del uno al nueve.

VerificaSintaxis es el primero de los tres métodos principales. Este recibe un String que representa la notación infija o, en otras palabras, es el argumento de entrada del usuario en la calculadora y nos devuelve un boolean que, mediante true o false nos indique su validez.

DelimitaCadena regresa la pila de Strings que vamos a transformar en la notación postfija posteriormente. Utiliza la mayoría de funciones anteriores para delimitar el orden de cada String que va a ser introducido a la pila de notación infija e introducirlo en dicha pila.

ConviertePostFijo recibe una pila, particularmente la que fue generada con el método DelimitaCadena, para transformarla a otra pila que guarde otro conjunto de Strings pero en notación postfija. Utiliza el algoritmo básico para ejecutar esta tarea, ya ampliamente desarrollado por muchas personas, por lo que solo fue necesario adaptarlo a java y utilizando los métodos como EsOperador o Pref para saber en qué orden introducir los Strings a la pila postfija.

El último, Resuelve, solo se encarga de evaluar toda la expresión postfija almacenada en la pila anterior, devolviendo el valor final de la entrada final, luego de pasar por todas las funciones anteriores.

La interfaz solo permite que introduzcas los dígitos, los operadores, los paréntesis, el igual y un botón para limpiar el área del resultado. En caso de que la expresión no sea correcta, escribe "Expresión Inválida", para poder escribir una nueva cadena.

Pruebas

A lo largo del proyecto comprobamos pertinentemente que el código funcionara correctamente con diferentes casos a tener en cuenta. Particularmente, nuestras pruebas demuestran que podemos traducir cualquier notación infija válida a postfija, y evaluarla sin ningún inconveniente. El apartado que más excepciones presenta es el primero, en el que por la mera abundancia de casos, tuvimos que hacer muy complejo el algoritmo para tenerlos todos en cuenta.

Pese a esta dificultad, en la prueba mostramos cómo nuestro programa puede:

1. Identificar paréntesis faltantes o desbalanceados
2. Operadores consecutivos erróneos
3. El signo negativo y sus diferentes interacciones con paréntesis, números y otros símbolos
4. Punto decimal dentro y fuera de los operandos
5. Primer carácter válido, es decir, un dígito, símbolo negativo o paréntesis de apertura
6. Último carácter válido, en otras palabras, un dígito o paréntesis de cierre
7. Paréntesis consecutivos y correctos (la expresión "((4))" es correcta, ")(4)(" no lo es aunque tenga los paréntesis balanceados)

De este modo garantizamos que se cumplan todos los casos para los que nuestra calculadora está programada y, en caso de que esta ejecución imprima un falso, se muestre un mensaje de error para el usuario.

**IMPORTANTE

Intentamos ejecutar la clase de prueba en las diferentes computadoras pero presentaron errores desconocidos durante el proceso. En el apéndice añadimos el código de la prueba en caso de que deseen arreglarlo.

Limitaciones

Refiriéndose a casos de posibles entradas equivocadas, con tal de que la expresión infija esté convencionalmente correcta, funciona en todos los casos. También se puede ejecutar la calculadora borrando el cuadro de resultados y reiniciando las operaciones. La única limitación es que no acepta paréntesis consecutivos (por ejemplo, ((4)) no lo tomaría como correcto). Fuera de esa excepción, cualquier expresión válida funciona.

El uso de números complejos también está restringido por Java, por lo que no lo tomamos como un caso a tener en cuenta. Cuando se encuentra con uno, Java solo muestra NAN, por lo que decidimos dejarlo como una opción disponible.

Para el uso de la calculadora solo se requiere NetBeans con la versión correspondiente indicada más arriba.

Áreas de Oportunidad y Conclusiones

Honestamente, el proyecto fue muy entretenido y no hubo prácticamente ningún problema durante el proceso. El trabajo en equipo fue impecable y cada parte hizo su trabajo de forma ordenada y puntual. Probablemente el método para revisar la notación infija no es el más eficiente, ya que tiene muchos casos en cuenta y conlleva muchas operaciones y ciclos, pero debido a la complejidad del problema, lo vimos necesario. Nuestro uso de GitHub no fue el más eficiente y preferimos pasar las diferentes partes del trabajo por otros medios (Gmail y Whatsapp), para facilitar enormemente el movimiento de líneas de código sin la necesidad de todo el proyecto. Además, los métodos de prueba de JUnit tuvieron problemas para su integración con el resto del código, por lo que los casos de prueba fueron manules.

Cumplimos las fechas estipuladas y acabamos muchas secciones antes de lo que teníamos planeado. Aunque no tuviéramos un miembro del equipo comunicador, no fue un inconveniente porque compensamos esta carencia con un mayor esfuerzo por parte de todos. Aprendimos mucho con el proyecto y nos parece que hicimos un gran equipo después de todo.

Apéndice

Métodos

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
```

```
/**
 * <pre>
 * Clase funcionalidades
 * Esta clase contiene todas las funcionalidades de la calculadora,
 * usa datos genéricos
 * </pre>
 * @author <ul>
 *      <li> Joshua Chaparro Sandoval </li>
 *      <li> Julián Hernández Alás </li>
 *      <li> Robert Derek Bowden Barbero </li>
 *      <li> Rodrigo Demay Carranza </li>
 *      <li> Óscar Larios Mancilla </li>
 * </ul>
 *
 */
public class Funcionalidades <T>{

    /**
     * <pre>
     * Esta función revisa si la cadena introducida es un operador, y si si lo es te denota su
jerarquia
     * </pre>
     * @param pref String cadena de texto, de preferencia de tamaño 1, a revisar
     * @return Int <ul>
     * <li> En caso de que la cadena sea un "+" o un "-", regresa un 3 </li>
     * <li> En caso de que la cadena sea un "*" o un "/", regresa un 4 </li>
     * <li> En caso de que la cadena sea un "^", regresa un 5 </li>
     * <li> Sino, regresa un 0 </li>
     * </ul>
     */
    public static int pref(String pref){
        int resp=0;

        if(pref.equals("+") || pref.equals("-"))
            resp=3;
        if(pref.equals("*") || pref.equals("/"))
           resp= 4;
        if(pref.equals("^"))
            resp=5;


        return resp;}
```

```
/**
 * <pre>
 * Esta función revisa que los parentesis de la cadena introducida estén balanceados
 * </pre>
 * @param cadena String Recibe la cadena que se quiere verificar
 * @return boolean <ul>
 * <li> Si los parentesis estan balanceados, regresa true </li>
 * <li> Si los parentesis no estan balanceados, regresa false </li>
 * </ul>
 */
public static boolean balanceParentesis(String cadena){
    boolean resp=false;
    int letras= cadena.length(), i=0;
    PilaA pila1= new PilaA();
    char letra;

    if(!cadena.equals("")){

    while(i<letras && i!=-1){
        letra=cadena.charAt(i);

        if(letra=='('
            pila1.push('(');

        if(letra==')'){
            if(!pila1.isEmpty() && pila1.peek().equals('('))
              pila1.pop();
            else
              i=-2;}

        i++;}

    if(i!=-1 && pila1.isEmpty())
        resp=true;}

   return resp;}
/**
 * Esta función revisa si la cadena introducida es un operador
 * @param caracter String Recibe una cadena de texto, de preferencia de tamaño 1
 * @return boolean <ul>
 * <li> Si la cadena es un operador, regresa true </li>
 * <li> Si la cadena no es un operador, regresa false </li>
 * </ul>
```

```
 */
public static boolean esOperador(String caracter){
   boolean resp=false;

   switch(caracter){
      case"+":resp=true;break;
      case"-":resp=true;break;
      case"*":resp=true;break;
      case"/":resp=true;break;
      case"^":resp=true;break;}

   return resp;}

/**
 * Esta función revisa si la cadena introducida es un operador, excluyendo a la resta
 * @param caracter String Recibe una cadena de texto, de preferencia de tamaño 1
 * @return boolean <ul>
 * <li> Si la cadena es un operador, regresa true </li>
 * <li> Si la cadena no es un operador, regresa false </li>
 * </ul>
 */
public static boolean esOperadorSinResta(String caracter){
   boolean resp=false;

   switch(caracter){
      case"+":resp=true;break;
      case"*":resp=true;break;
      case"/":resp=true;break;
      case"^":resp=true;break;}

   return resp;}

/**
 * Esta función revisa si la cadena introducida es un numero
 * @param caracter String Recibe una cadena de texto, de preferencia de tamaño 1
 * @return boolean <ul>
 * <li> Si la cadena es un numero, regresa true </li>
 * <li> Si la cadena no es un numero, regresa false </li>
 * </ul>
 */
public static boolean esNumero(String caracter){
   boolean resp=false;
```

```
    switch(caracter){
        case"1":resp=true;break;
        case"2":resp=true;break;
        case"3":resp=true;break;
        case"4":resp=true;break;
        case"5":resp=true;break;
        case"6":resp=true;break;
        case"7":resp=true;break;
        case"8":resp=true;break;
        case"9":resp=true;break;
        case"0":resp=true;break;
    }

    return resp;}

/**
 * Esta función revisa que la cadena tenga una sintaxis matematica correcta
 * @param cadena String Recibe una cadena de texto
 * @return boolean <ul>
 * <li> Si la cadena tiene una sintaxis correcta, regresa true </li>
 * <li> Si la cadena no tiene una sintaxis correnta, regresa false </li>
 * </ul>
 * @see <ul>
 * <li> balanceParentesis </li>
 * <li> esOperador </li>
 * <li> esNumero </li>
 * </ul>
 */
public static boolean verificaSintaxis(String cadena){
    PilaA pila= new PilaA();PilaA pila1= new PilaA();
    boolean resp=false, sigue=true;
    String letra;
    int i=0;

    while(i<cadena.length() && i!=-1){
        letra=String.valueOf(cadena.charAt(i));

        if(esNumero(letra)){                //NUMEROS
            if(pila.isEmpty() || !String.valueOf(pila.peek()).equals(")"))
                pila.push(letra);
            else
                i=-2;}
```

```
        if(esOperadorSinResta(letra)){     // + / * ^
           if(!pila.isEmpty() && !esOperador(String.valueOf(pila.peek())) &&
!String.valueOf(pila.peek()).equals(".") && !String.valueOf(pila.peek()).equals("("))
              pila.push(letra);
           else
              i=-2;}

        if(letra.equals("-")){            // -
           if(String.valueOf(pila.peek()).equals("."))
              i=-2;
           if(pila.isEmpty())
             pila.push(letra);

           else{
              if(String.valueOf(pila.peek()).equals("-")){
                 pila1.push(pila.pop());

                 if(pila.isEmpty() || String.valueOf(pila.peek()).equals("-"))
                    i=-2;
                 else{
                    pila.push(pila1.pop());
                    pila.push(letra);}}

              else
                 pila.push(letra);}        }

        if(letra.equals(".")){            // .
           if(esNumero(String.valueOf(pila.peek()))){

              sigue=true;
                 while(sigue && i!=-2 && pila.peek()!=null){
                    if(String.valueOf(pila.peek()).equals("."))
                       i=-2;
                    else{
                       if(esOperador(String.valueOf(pila.peek())))
                          sigue=false;

                       else
                          pila1.push(pila.pop());}}

              while(pila1.peek()!=null)
                 pila.push(pila1.pop());
```

```
            if(i!=-2)
               pila.push(letra);}

         else
            i=-2;}


      if(letra.equals("(")){          // (
         if(pila.isEmpty() || esOperador(String.valueOf(pila.peek())))
           pila.push(letra);
         else
           i=-2;}

      if(letra.equals(")")){          // )
         if(!pila.isEmpty() && esNumero(String.valueOf(pila.peek())))
            pila.push(letra);
         if(String.valueOf((pila.peek())).equals(")"))
            pila.push(letra);
         else
            i=-2;}

      i++;}

   if(pila.isEmpty())
      i=-1;

   if(esOperador(String.valueOf(pila.peek())) || String.valueOf(pila.peek()).equals(".") ||
String.valueOf(pila.peek()).equals("("))
      i=-1;

   if(i!=-1 && balanceParentesis(cadena))
      resp=true;

   return resp;}


  /**
   * Esta función delimita los números y regresa una pila
   * @param cadena String Recibe una cadena de texto
   * @return PilaA Regresa una PilaA con la cadena delimitada
   * @see <ul>
   * <li> esOperador </li>
   * <li> esOperadorSinResta </li>
```

```
 * <li> esNumero </li>
 * </ul>
 */
public static PilaA delimitaCadena(String cadena){
   PilaA pila1= new PilaA(); PilaA pila2= new PilaA();
   String numero="", letra="";
   int i=0;

   while(i<cadena.length()){
      letra=String.valueOf(cadena.charAt(i));

      if(i==cadena.length()-1){
         if(letra.equals(")")){
            if(pila1.peek().equals(")"))
               pila1.push(letra);
            else{
               pila1.push(numero);
               pila1.push(letra);}}
         else{
            numero=numero+letra;
            pila1.push(numero);}}

      else{

         if(esNumero(letra) || letra.equals("."))
            numero=numero+letra;

         if(esOperadorSinResta(letra)){
            if(pila1.isEmpty()){
               pila1.push(numero);
               pila1.push(letra);}
            else
               if(pila1.peek().equals(")"))
                  pila1.push(letra);
               else{
                  pila1.push(numero);
                  pila1.push(letra); }

               numero="";    }

         if(letra.equals(")")){
            if(pila1.peek().equals(")"))
              pila1.push(letra);
```

```
                else{
                   pila1.push(numero);
                   pila1.push(letra);}
                numero="";}

             if(letra.equals("(")){
                if(!numero.equals("")){
                   pila1.push(numero);
                   pila1.push(letra);
                   numero="";}
                else
                   pila1.push(letra);}

             if(letra.equals("-")){
                if(!numero.equals("")){
                   pila1.push(numero); pila1.push(letra); numero="";}

                else{
                if(pila1.isEmpty() || esOperador(String.valueOf(pila1.peek())) ||
pila1.peek().equals("(") /*|| pila1.peek().equals(")")*/)
                   numero="-";

                else{
                   if(numero.equals(""))
                      pila1.push(letra);

                   else{
                      pila1.push(numero);
                      pila1.push(letra);}numero="";}

                   }}}

                i++;}

          while(!pila1.isEmpty())
             pila2.push(pila1.pop());

          return pila2;}


    /**
     * Esta función convierte la expresión de infijo a postfijo
     * @param Expresion PilaA Recibe una pila en infijo
```

```
 * @return PilaA Regresa una PilaA en postfijo
 * @see <ul>
 * <li> esOperador </li>
 * </ul>
 */
public static PilaA conviertePostFijo(PilaA Expresion){
   PilaA Operadores= new PilaA(); PilaA Salida= new PilaA();
   String valor;

   while(!Expresion.isEmpty()){
      valor=String.valueOf(Expresion.peek());


      if(esOperador(valor)|| valor.equals("(") || valor.equals(")")){

         if(esOperador(valor)){

            if(Operadores.isEmpty())
           Operadores.push(Expresion.pop());

         else{
           if(pref(valor) < pref(String.valueOf(Operadores.peek())))
              Salida.push(Operadores.pop());


           if(pref(valor) == pref(String.valueOf(Operadores.peek())))
               Salida.push(Operadores.pop());

           Operadores.push(Expresion.pop());}}

      if(valor.equals("("))
        Operadores.push(Expresion.pop());

      if(valor.equals(")")){
         while(!String.valueOf(Operadores.peek()).equals("("))
            Salida.push(Operadores.pop());

            Operadores.pop();Expresion.pop();}}

      else
         Salida.push(Expresion.pop());}

   while(!Operadores.isEmpty())
```

```
            Salida.push(Operadores.pop());

        while(!Salida.isEmpty())
            Operadores.push(Salida.pop());


        return Operadores;}


    /**
     * Esta función resuelve la expresión en formato postfija
     * @param pila PilaA Recibe una pila en postfija
     * @return double regresa el resultado de la expresión postfija
     */
    public static <T> double Resuelve (PilaA pila){
        PilaA pila1= new PilaA();
        double operacion1, operacion2, resp=0.0;
        Object aux;

            while(!pila.isEmpty()){
                aux=String.valueOf(pila.peek());

                if(!aux.equals("+") && !aux.equals("-") && !aux.equals("*") && !aux.equals("^")
&& !aux.equals("/"))
                    pila1.push(pila.pop());

                else{
                    operacion1=Double.parseDouble(String.valueOf(pila1.pop()));
                    operacion2=Double.parseDouble(String.valueOf(pila1.pop()));

                    switch (String.valueOf(aux)){
                        case "+": resp=operacion2+operacion1;break;
                        case "-": resp=operacion2-operacion1;break;
                        case "*": resp=operacion2*operacion1;break;
                        case "/": resp=operacion2/operacion1;break;
                        case "^": resp=Math.pow(operacion2, operacion1);break;}

                    pila1.push(String.valueOf(resp));
                    pila.pop();}}

        return Double.parseDouble(String.valueOf(pila1.peek()));}
}
```

Interfaz gráfica

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */

/**
 *
 * @author olari
 */
public class InterfazGrafica extends javax.swing.JFrame {

    /**
     * Creates new form InterfazGrafica
     */
    public InterfazGrafica() {
        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">
    private void initComponents() {

        jButton4 = new javax.swing.JButton();
        jButton7 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        jPanel4 = new javax.swing.JPanel();
        boton4 = new javax.swing.JButton();
        boton0 = new javax.swing.JButton();
        botonParentesisA = new javax.swing.JButton();
        botonPunto = new javax.swing.JButton();
        botonBorrarIndividual = new javax.swing.JButton();
        botonIgual = new javax.swing.JButton();
        botonSuma = new javax.swing.JButton();
```

```
        botonResta = new javax.swing.JButton();
        botonBorrar1 = new javax.swing.JButton();
        boton6 = new javax.swing.JButton();
        boton3 = new javax.swing.JButton();
        boton2 = new javax.swing.JButton();
        botonParentesisC = new javax.swing.JButton();
        botonPotencia = new javax.swing.JButton();
        botonDivision = new javax.swing.JButton();
        boton7 = new javax.swing.JButton();
        boton8 = new javax.swing.JButton();
        boton9 = new javax.swing.JButton();
        boton5 = new javax.swing.JButton();
        botonMultiplicacion = new javax.swing.JButton();
        boton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        Panel = new javax.swing.JTextArea();

        jButton4.setText("jButton1");

jButton4.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBor
der.RAISED));
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jButton7.setText("jButton1");

jButton7.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBor
der.RAISED));
        jButton7.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton7ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 100, Short.MAX_VALUE)
        );
```

```
jPanel1Layout.setVerticalGroup(
   jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGap(0, 100, Short.MAX_VALUE)
);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Calculadora");
setBackground(new java.awt.Color(0, 104, 83));

jPanel4.setBackground(new java.awt.Color(0, 104, 83));
jPanel4.setForeground(new java.awt.Color(102, 255, 51));

boton4.setBackground(new java.awt.Color(255, 255, 255));
boton4.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton4.setText("4");
boton4.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton4.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      boton4ActionPerformed(evt);
   }
});

boton0.setBackground(new java.awt.Color(255, 255, 255));
boton0.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton0.setText("0");
boton0.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton0.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      boton0ActionPerformed(evt);
   }
});

botonParentesisA.setBackground(new java.awt.Color(255, 255, 255));
botonParentesisA.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonParentesisA.setText("(");
botonParentesisA.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonParentesisA.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      botonParentesisAActionPerformed(evt);
   }
});

botonPunto.setBackground(new java.awt.Color(255, 255, 255));
```

```
botonPunto.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonPunto.setText(".");
botonPunto.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonPunto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonPuntoActionPerformed(evt);
    }
});

botonBorrarIndividual.setBackground(new java.awt.Color(255, 255, 255));
botonBorrarIndividual.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonBorrarIndividual.setText("DEL");
botonBorrarIndividual.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonBorrarIndividual.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonBorrarIndividualActionPerformed(evt);
    }
});

botonIgual.setBackground(new java.awt.Color(255, 255, 255));
botonIgual.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonIgual.setForeground(new java.awt.Color(0, 104, 83));
botonIgual.setText("=");
botonIgual.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonIgual.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonIgualActionPerformed(evt);
    }
});

botonSuma.setBackground(new java.awt.Color(255, 255, 255));
botonSuma.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonSuma.setText("+");
botonSuma.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonSuma.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonSumaActionPerformed(evt);
    }
});

botonResta.setBackground(new java.awt.Color(255, 255, 255));
botonResta.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonResta.setText("−");
```

```
botonResta.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonResta.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      botonRestaActionPerformed(evt);
   }
});

botonBorrar1.setBackground(new java.awt.Color(255, 255, 255));
botonBorrar1.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonBorrar1.setText("AC");
botonBorrar1.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonBorrar1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      botonBorrar1ActionPerformed(evt);
   }
});

boton6.setBackground(new java.awt.Color(255, 255, 255));
boton6.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton6.setText("6");
boton6.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton6.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      boton6ActionPerformed(evt);
   }
});

boton3.setBackground(new java.awt.Color(255, 255, 255));
boton3.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton3.setText("3");
boton3.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      boton3ActionPerformed(evt);
   }
});

boton2.setBackground(new java.awt.Color(255, 255, 255));
boton2.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton2.setText("2");
boton2.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton2.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        boton2ActionPerformed(evt);
    }
});

botonParentesisC.setBackground(new java.awt.Color(255, 255, 255));
botonParentesisC.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonParentesisC.setText(")");
botonParentesisC.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonParentesisC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonParentesisCActionPerformed(evt);
    }
});

botonPotencia.setBackground(new java.awt.Color(255, 255, 255));
botonPotencia.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonPotencia.setText("^");
botonPotencia.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonPotencia.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonPotenciaActionPerformed(evt);
    }
});

botonDivision.setBackground(new java.awt.Color(255, 255, 255));
botonDivision.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonDivision.setText("÷ ");
botonDivision.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonDivision.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonDivisionActionPerformed(evt);
    }
});

boton7.setBackground(new java.awt.Color(255, 255, 255));
boton7.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton7.setText("7");
boton7.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        boton7ActionPerformed(evt);
    }
});
```

```
boton8.setBackground(new java.awt.Color(255, 255, 255));
boton8.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton8.setText("8");
boton8.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        boton8ActionPerformed(evt);
    }
});

boton9.setBackground(new java.awt.Color(255, 255, 255));
boton9.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton9.setText("9");
boton9.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        boton9ActionPerformed(evt);
    }
});

boton5.setBackground(new java.awt.Color(255, 255, 255));
boton5.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
boton5.setText("5");
boton5.setBorder(javax.swing.BorderFactory.createCompoundBorder());
boton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        boton5ActionPerformed(evt);
    }
});

botonMultiplicacion.setBackground(new java.awt.Color(255, 255, 255));
botonMultiplicacion.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
botonMultiplicacion.setText("×");
botonMultiplicacion.setBorder(javax.swing.BorderFactory.createCompoundBorder());
botonMultiplicacion.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        botonMultiplicacionActionPerformed(evt);
    }
});

boton1.setBackground(new java.awt.Color(255, 255, 255));
boton1.setFont(new java.awt.Font("Tahoma", 0, 20)); // NOI18N
```

```
        boton1.setText("1");
        boton1.setBorder(javax.swing.BorderFactory.createCompoundBorder());
        boton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                boton1ActionPerformed(evt);
            }
        });

        jScrollPane1.setBorder(null);

        Panel.setFont(new java.awt.Font("Yu Gothic UI Semilight", 1, 24)); // NOI18N
        Panel.setLineWrap(true);
        Panel.setTabSize(0);
        Panel.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT_CURSOR));
        Panel.setKeymap(null);
        jScrollPane1.setViewportView(Panel);

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1)
                    .addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(boton7, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(boton4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(boton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(botonBorrarIndividual,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(botonBorrar1, javax.swing.GroupLayout.PREFERRED_SIZE,
65, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(2, 2, 2)
```

```
.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel4Layout.createSequentialGroup()
                        .addComponent(boton0, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(1, 1, 1)
                        .addComponent(botonPunto,
javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
, false)
                            .addComponent(boton5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(boton8, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(botonParentesisA,
javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(boton2, javax.swing.GroupLayout.PREFERRED_SIZE,
65, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(1, 1, 1)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(boton6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(boton9, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(botonParentesisC,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addComponent(boton3, javax.swing.GroupLayout.PREFERRED_SIZE,
65, javax.swing.GroupLayout.PREFERRED_SIZE))))
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                        .addComponent(botonResta, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(botonSuma, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(botonMultiplicacion,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(botonDivision,
javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(botonPotencia,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(botonIgual, javax.swing.GroupLayout.PREFERRED_SIZE,
69, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 64,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(boton0, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(botonPunto, javax.swing.GroupLayout.PREFERRED_SIZE,
57, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(botonBorrarIndividual,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(botonParentesisC,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(botonParentesisA,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(botonBorrar1,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(59, 59, 59)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                            .addComponent(boton9,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(boton8,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(boton7,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(1, 1, 1)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                            .addComponent(boton6,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addComponent(boton5,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(boton4,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(175, 175, 175)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                        .addComponent(boton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(boton3,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(boton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGap(57, 57, 57)))
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addComponent(botonPotencia, javax.swing.GroupLayout.PREFERRED_SIZE,
57, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(1, 1, 1)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addComponent(botonDivision,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(1, 1, 1)
                        .addComponent(botonMultiplicacion,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(1, 1, 1)
                        .addComponent(botonResta, javax.swing.GroupLayout.PREFERRED_SIZE,
57, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(174, 174, 174)
                        .addComponent(botonSuma, javax.swing.GroupLayout.PREFERRED_SIZE,
57, javax.swing.GroupLayout.PREFERRED_SIZE)))
```

```
                    .addComponent(botonIgual, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(0, 8, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        pack();
    }// </editor-fold>

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void botonParentesisCActionPerformed(java.awt.event.ActionEvent evt) {
        if(Panel.getText().equals("Expresion Invalida"))
            Panel.setText("");

        Panel.setText(Panel.getText()+")");

    }

    private void botonPotenciaActionPerformed(java.awt.event.ActionEvent evt) {
        if(Panel.getText().equals("Expresion Invalida"))
            Panel.setText("");
        Panel.setText(Panel.getText()+"^");
    }
```

```
private void botonDivisionActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"/");
}

private void boton7ActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"7");
}

private void boton8ActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"8");
}

private void boton9ActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"9");
}

private void botonMultiplicacionActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"*");
}

private void boton0ActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+"0");
}

private void botonPuntoActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
      Panel.setText("");
   Panel.setText(Panel.getText()+".");
}

private void botonSumaActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"+");
}

private void botonRestaActionPerformed(java.awt.event.ActionEvent evt) {
  if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"-");
}

private void boton6ActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"6");
}

private void boton3ActionPerformed(java.awt.event.ActionEvent evt) {
   if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"3");
}

private void boton2ActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"2");
}

private void boton5ActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"5");
}

private void boton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"1");
}

private void boton4ActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
```

```
        Panel.setText("");
    Panel.setText(Panel.getText()+"4");
}

private void botonParentesisAActionPerformed(java.awt.event.ActionEvent evt) {
    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    Panel.setText(Panel.getText()+"(");
}

private void botonBorrarIndividualActionPerformed(java.awt.event.ActionEvent evt) {

    if(Panel.getText().equals("Expresion Invalida"))
        Panel.setText("");
    String newResp="";

    for(int i=0;i<Panel.getText().length()-1;i++)
        newResp=newResp+String.valueOf(Panel.getText().charAt(i));

    Panel.setText(newResp);

}

private void botonBorrar1ActionPerformed(java.awt.event.ActionEvent evt) {
    Panel.setText("");
}

private void botonIgualActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String texto=Panel.getText();
    boolean resp=Funcionalidades.verificaSintaxis(texto);
    PilaA pila1 = new PilaA(); PilaA pila2 = new PilaA();
    double valor;

    if(resp){
        pila1= Funcionalidades.delimitaCadena(texto);
        pila2= Funcionalidades.conviertePostFijo(pila1);
        valor=Funcionalidades.Resuelve(pila2);

        Panel.setText(String.valueOf(valor));}

    else
    Panel.setText("Expresion Invalida");
```

```
    }

    /**
     * @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(InterfazGrafica.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(InterfazGrafica.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(InterfazGrafica.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(InterfazGrafica.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 InterfazGrafica().setVisible(true);
```

```
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JTextArea Panel;
private javax.swing.JButton boton0;
private javax.swing.JButton boton1;
private javax.swing.JButton boton2;
private javax.swing.JButton boton3;
private javax.swing.JButton boton4;
private javax.swing.JButton boton5;
private javax.swing.JButton boton6;
private javax.swing.JButton boton7;
private javax.swing.JButton boton8;
private javax.swing.JButton boton9;
private javax.swing.JButton botonBorrar1;
private javax.swing.JButton botonBorrarIndividual;
private javax.swing.JButton botonDivision;
private javax.swing.JButton botonIgual;
private javax.swing.JButton botonMultiplicacion;
private javax.swing.JButton botonParentesisA;
private javax.swing.JButton botonParentesisC;
private javax.swing.JButton botonPotencia;
private javax.swing.JButton botonPunto;
private javax.swing.JButton botonResta;
private javax.swing.JButton botonSuma;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton7;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel4;
private javax.swing.JScrollPane jScrollPane1;
// End of variables declaration
}
```

Clase de prueba JUnit

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/UnitTests/JUnit5TestClass.java to edit this template
```

```
 */

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

/**
 *
 * @author RDEMAYC
 */
public class FuncionalidadesTest {

    public FuncionalidadesTest() {
    }

    /**
     *
     * @throws Exception
     */
    @org.junit.jupiter.api.BeforeAll
    public static void setUpClass() throws Exception {
    }

    @org.junit.jupiter.api.AfterAll
    public static void tearDownClass() throws Exception {
    }

    @org.junit.jupiter.api.BeforeEach
    public void setUp() throws Exception {
    }

    @org.junit.jupiter.api.AfterEach
    public void tearDown() throws Exception {
    }
   /*
    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
```

```
public static void tearDownClass() {
}

@BeforeEach
public void setUp() {
}

@AfterEach
public void tearDown() {
}

/**
 * Test of pref method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testPref() {
    System.out.println("pref");
    String pref = "*";
    int expResult = 4;
    int result = Funcionalidades.pref(pref);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}

/**
 * Test of balanceParentesis method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testBalanceParentesis() {
    System.out.println("balanceParentesis");
    String cadena = "()";
    boolean expResult = true;
    boolean result = Funcionalidades.balanceParentesis(cadena);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}

/**
 * Test of esOperador method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
```

```
public void testEsOperador() {
    System.out.println("esOperador");
    String caracter = "+";
    boolean expResult = true;
    boolean result = Funcionalidades.esOperador(caracter);
    assertEquals(expResult, result);
}

/**
 * Test of esOperadorSinResta method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testEsOperadorSinResta() {
    System.out.println("esOperadorSinResta");
    String caracter = "-";
    boolean expResult = false;
    boolean result = Funcionalidades.esOperadorSinResta(caracter);
    assertEquals(expResult, result);
}

    @org.junit.jupiter.api.Test
public void testEsOperadorSinResta2() {
    System.out.println("esOperadorSinResta");
    String caracter = "+";
    boolean expResult = true;
    boolean result = Funcionalidades.esOperadorSinResta(caracter);
    assertEquals(expResult, result);
}
/**
 * Test of esNumero method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testEsNumero() {
    System.out.println("esNumero");
    String caracter = "1";
    boolean expResult = true;
    boolean result = Funcionalidades.esNumero(caracter);
    assertEquals(expResult, result);
}


 @org.junit.jupiter.api.Test
public void testEsNumero2() {
```

```
    System.out.println("esNumero");
    String caracter = "+";
    boolean expResult = false;
    boolean result = Funcionalidades.esNumero(caracter);
    assertEquals(expResult, result);
}

/**
 * Test of verificaSintaxis method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testVerificaSintaxis() {
    System.out.println("verificaSintaxis");
    String cadena = "(1+1/3-47*(72))";
    boolean expResult = true;
    boolean result = Funcionalidades.verificaSintaxis(cadena);
    assertEquals(expResult, result);
}

@org.junit.jupiter.api.Test
public void testVerificaSintaxis2() {
    System.out.println("verificaSintaxis");
    String cadena = "(hola como estas?)";
    boolean expResult = false;
    boolean result = Funcionalidades.verificaSintaxis(cadena);
    assertEquals(expResult, result);
}
/**
 * Test of delimitaCadena method, of class Funcionalidades.
 */
@org.junit.jupiter.api.Test
public void testDelimitaCadena() {
    System.out.println("delimitaCadena");
    String cadena = "1+1";
    PilaA res = new PilaA();
    res.push(1);
    res.push("+");
    res.push(1);

    PilaA expResult = res;
    PilaA result = Funcionalidades.delimitaCadena(cadena);
    assertEquals(expResult, result);
```

```
    }

    /**
     * Test of conviertePostFijo method, of class Funcionalidades.
     */
    @org.junit.jupiter.api.Test
    public void testConviertePostFijo() {
        System.out.println("conviertePostFijo");
        PilaA res = new PilaA();
        res.push(1);
        res.push("+");
        res.push(1);
        PilaA Expresion = res;
        PilaA p2= new PilaA();
        p2.push("+");
        p2.push(1);
        p2.push(1);

        PilaA expResult = p2;
        PilaA result = Funcionalidades.conviertePostFijo(Expresion);
        assertEquals(expResult, result);

    }

    /**
     * Test of Resuelve method, of class Funcionalidades.
     */
    @org.junit.jupiter.api.Test
    public void testResuelve() {
        System.out.println("Resuelve");

        PilaA pila = new PilaA();
        pila.push("+");
        pila.push(1);
        pila.push(1);
        double expResult = 2.0;
        double result = Funcionalidades.Resuelve(pila);
        assertEquals(expResult, result, 0.0);

    }

}
```