INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Proyecto I: Calculadora

22 de febrero de 2021

Semestre primavera 2021

Benjamín Goñi Hernández
Fernando Medina López
Marcos Medina Lopera
María Francisca Flores Rosas
Mariana Luna Rocha
Rebeca Estephania Angulo Rojas

Estructuras de Datos (002)

Profesora Silvia del Carmen Guardati

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Índice

Introducción

Se requiere crear una calculadora, que a través de una interfaz gráfica, pueda realizar diversas operaciones aritméticas con números reales. Las operaciones que se desean realizar son: suma, resta, multiplicación y división; será capaz de realizar estas operaciones con números largos, con punto decimal, con signo negativo o que contengan paréntesis para indicar la jerarquía de operaciones.

La interfaz debe de contar con botones para las cuatro operaciones mencionadas anteriormente y los dos paréntesis (izquierdo y derecho), también debe tener un botón para cada uno de los diez dígitos 0-9, uno para el punto decimal, uno para indicar "cambio de signo" (para poder expresar valores negativos), y uno con el símbolo "=" para que el programa comience a evaluar la expresión que el usuario haya indicado. Además, la calculadora deberá tener dos botones para "limpiar el contenido" de la ventana. El primero, limpiará el contenido completo que se ingresó permitiendo que el usuario ingrese una nueva expresión a evaluar y, el segundo, limpiará el último carácter ingresado para permitir que el usuario corrija algún error que pudo haber cometido.

Para estos efectos, el botón con el símbolo "=" deberá realizar 3 distintas tareas para asegurar el correcto funcionamiento de la calculadora:

- Verificar que la expresión ingresada sea correcta, asegurándose de revisar que los paréntesis estén balanceados, que no haya varios operadores seguidos y, no haya puntos seguidos.
- Convertir la expresión ingresada de notación infija a postfija.
- Evaluar la expresión general y arrojar el resultado sin errores.

Finalmente, este programa está diseñado para ser ejecutado en el IDE Netbeans, apegándose a las restricciones indicadas en el manual de usuario.

Objetivo

El objetivo de este proyecto es realizar una calculadora que funcione de la manera más eficiente, haciendo uso de lo aprendido en el curso sobre estructuras de datos y algorítmica. Otro objetivo que se persigue es conjuntar las habilidades del equipo de trabajo para desarrollar el software, siguiendo un orden para los fines antes mencionados.

Diseño

Descripción del diseño

Para resolver el problema planteado anteriormente se decidió programar clases diferentes en Java que conforman el software.

Primero, se creó la clase "Caratula" para poder modelar la interfaz gráfica de la calculadora, incluyendo la programación necesaria de los botones y escuchadores. Posteriormente, se creó la clase "Resuelve", donde se codificaron los diversos métodos necesarios para el funcionamiento de la calculadora. Finalmente, se realizó la documentación Javadoc y las pruebas unitarias de las clases que permitieron evaluar la funcionalidad del programa.

A continuación, se describirán las clases de apoyo usadas y se mostrarán los diagramas UML de las mismas.

Clases PilaADT y PilaA: estas clases se realizaron junto con la profesora para definir el comportamiento de la estructura de datos: pila. Se utilizan como apoyo en los métodos de la clase "Resuelve".

Clase ExcepcionColeccionVacia: esta clase se trabajó, de igual manera, con la profesora; define lo que sucede en caso de encontrar una pila vacía mediante el uso de un try-catch.

Diagramas UML

RevisorParentesis
−cadenaRevisar: String
+RevisorParentesis () +RevisorParentesis (String) +analisis (): boolean

Resuelve
+Resuelve () +revisaErrores (String): boolean +agregaCad (String): String [] +operador (String): String +getPrioridad (String): int +jerarquiaOperaciones (String, String): boolean +convierteAPosfijo (String []): String +resolverPosfijo (String []): double

Caratula
−TextInput: JTextArea −TextResultado: JTextField −btn: JButton −btn2: JButton −btn3: JButton −btn4: JButton −btn5: JButton −btn6: JButton −btn7: JButton −btn8: JButton −btn9: JButton −btnBorraTodo: JButton −btnBorraUno: JButton −btnCero: JButton −btnDecimal: JButton −btnDiv: JButton −btnIgual: JButton −btnMas: JButton −btnMasMenos: JButton −btnMenos: JButton −btnParentesisDer: JButton −btnParentesisIzq: JButton −btnPor: JButton −jLabel1: JLabel −jScrollPane1: JScrollPane −jScrollPane2: JScrollPane −jTextArea1: JTextArea
+Caratula () - initComponents () - btn9ActionPeformed (ActionEvent evt) - btn1ActionPeformed (ActionEvent evt) - btn2ActionPeformed (ActionEvent evt) - btn3ActionPeformed (ActionEvent evt) - btn4ActionPeformed (ActionEvent evt) - btn5ActionPeformed (ActionEvent evt) - btn6ActionPeformed (ActionEvent evt) - btn7ActionPeformed (ActionEvent evt) - btn8ActionPeformed (ActionEvent evt) - btnCeroActionPeformed (ActionEvent evt) - btnDecimalActionPeformed (ActionEvent evt) - btnBorraUnoActionPeformed (ActionEvent evt) - btnBorraTodoActionPeformed (ActionEvent evt) - btnMasActionPeformed (ActionEvent evt) - btnMenosActionPeformed (ActionEvent evt) - btnPorActionPeformed (ActionEvent evt) - btnDivActionPeformed (ActionEvent evt) - btnIgualActionPeformed (ActionEvent evt) - btnMasMenosActionPeformed (ActionEvent evt) - btnParentesisIzqActionPeformed (ActionEvent evt) - btnParentesisDerPeformed (ActionEvent evt) - btnTextResultadoActionPeformed (ActionEvent evt)

Códigos:

A continuación, se describirán brevemente los algoritmos principales de la calculadora.

- Algoritmo revisaErrores: este algoritmo está encargado de revisar que la cadena ingresada por el usuario esté libre de cualquier error sintáctico. Se ingresa una cadena al método y se revisa que:
    - Los paréntesis estén balanceados.
    - No se empiece o finalice la expresión con un operador.
    - No haya dos operadores seguidos.
    - Después o antes de un paréntesis, no haya un número.

Y finalmente, después de revisar todo lo explicado, arroja un resultado booleano.

- Algoritmo agregaCad: este algoritmo se encarga de llenar un arreglo con la cadena deseada. Este método permite una fácil manipulación de cadenas mediante el uso de la clase String Tokenizer.
- Algoritmos operador, getPrioridad y jerarquiaOperaciones: estos tres métodos se encargan de identificar la jerarquía de operaciones de una expresión. Arroja un resultado booleano.
- Algoritmo convierteAPosfijo: se encarga de convertir una cadena de tipo infijo a postfijo con ayuda de pilas. Recibe un arreglo de tipo String y regresa una cadena.
- Algoritmo resolverPosfijo: se encarga de resolver la expresión antes convertida a postfijo. Igualmente, este algoritmo usa como apoyo pilas. Recibe un arreglo de tipo String y regresa un double como resultado final que verá el usuario.

Para ver la composición de los códigos, consulte el anexo.

Pruebas

Para probar los diferentes métodos del programa, hicimos diversas pruebas unitarias que nos permitieron saber si la implementación del método era adecuada.

Descripción de las pruebas realizadas:

Para probar la solución se realizó una sola clase de prueba (ResuelveTest), en la que se incluían todos los métodos que le dan funcionalidad a la calculadora; esta clase fue generada como prueba unitaria JUnit. Además, se realizaron otras pequeñas pruebas en el main de la clase Caratula para revisar que el funcionamiento de la calculadora fuera excelente.

testOperador

```
@Test
public void testOperador() {
    System.out.println("operador");
    String cad = "5";
    String expResult = "num";
    String result = Resuelve.operador(cad);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}
```

Prueba exitosa: 5 sí es un número.

testGetPrioridad

```
@Test
public void testGetPrioridad() {
    System.out.println("getPrioridad");
    String c = "";
    int expResult = 0;
    int result = Resuelve.getPrioridad(c);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}
```

Prueba exitosa: vacío tiene una prioridad 0.

testConvierteAPostfijo

```
@Test
public void testConvierteAPostfijo() {
    System.out.println("convierteAPostfijo");//( 6 + 1.5 )  * 5 + 5.5 / 5 + 6
    String[] cad1 = new String[14];
    cad1[0] ="(";
    cad1[1] ="6";
    cad1[2] ="+";
    cad1[3] ="1.5";
    cad1[4] =")";
    cad1[5] ="*";
    cad1[6] ="5";
    cad1[7] ="+";
    cad1[8] ="5.5";
    cad1[9] ="/";
    cad1[10] ="5";
    cad1[11] ="+";
    cad1[12] ="6";


    String expResult = " 6 1.5 + 5 * 5.5 5 / + 6 + ";
    String result = Resuelve.convierteAPostfijo(cad1);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
```

Prueba exitosa: la expresión sí fue convertida a postfija.

testJerarquiaOperaciones

```
@Test
public void testJerarquiaOperaciones() {
    System.out.println("jerarquiaOperaciones");
    String cadena = "+";
    String elemPila = "*";
    boolean expResult = false;
    boolean result = Resuelve.jerarquiaOperaciones(cadena, elemPila);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}
```

Prueba exitosa: + tiene mayor jerarquía que *.

testAgregaCad

```
    /**
     * Test of agregarCad method, of class Resuelve.
     */
//     @Test
//     public void testAgregarCad() {
//         System.out.println("agregarCad");
//         String cadena = "";
//         String[] expResult = null;
//         String[] result = Resuelve.agregarCad(cadena);
//         assertArrayEquals(expResult, result);
//         // TODO review the generated test code and remove the default call to fail.
//
//     }
```

Prueba exitosa: el resultado sería un mapa de memoria y por ese motivo está comentado, pero es exitoso el resultado del método.

testRevisaErrores

```
@Test
public void testRevisaErrores() {
    System.out.println("revisaErrores");
    String input = " ( 6 + 1.5 )  * 5 + 5.5 / 5 + 6";
    boolean expResult = true;
    boolean result = Resuelve.revisaErrores(input);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.

}
```

Prueba exitosa: La expresión probada es correcta, por lo tanto true.

Análisis de resultados:

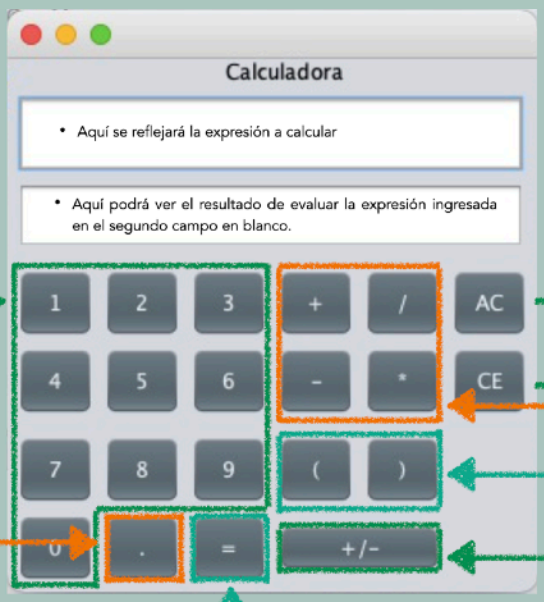
Durante las pruebas, se evaluaron todos los métodos con diferentes expresiones para considerar todos los casos posibles. Inicialmente se tuvieron algunos errores, pero al avanzar en las pruebas se logró corregirlos eficientemente.

Manual de usuario

Este proyecto representa una manera familiar de lo que conocemos como calculadora, esto permite que la interacción del usuario con la interfaz sea sencilla. Para hacer un uso adecuado de esta calculadora, se requiere lo siguiente:

1. Se deberá de indicar las operaciones que quiere realizar SÓLO mediante el uso de los botones de la interfaz. Evite ingresar una expresión con el uso del teclado.
2. Cuide no cometer errores de sintaxis como ingresar dos operadores seguidos, dos puntos seguidos, paréntesis mal colocados, o finalizar e iniciar la expresión con un operador.
3. Recuerde que la división entre cero no podrá ser resuelta; en caso de intentarlo, verá un mensaje de "Infinity" como resultado.
4. Para el uso de números negativos utilice el botón "+/-" ya que en caso de usar el botón "-" la expresión no podrá ser evaluada.
    a. Al usar "+/-" aparecerá automáticamente un paréntesis que evitará la ambigüedad sintáctica en la expresión; es decir, al presionar el botón verá que un paréntesis se abre y **usted deberá cerrarlo** al terminar de expresar el valor deseado.
        i. Por ejemplo: -3+6, en la calculadora equivale a (-3)+6
    b. Si usted desea hacer una operación dentro de un paréntesis con números negativos, deberá incluir dentro de un paréntesis toda la operación y cerrar el respectivo del botón "+/-".
        i. Por ejemplo: 5 * (-3+6), en la calculadora equivale a 5*((-3)+6)
5. En el momento que usted desee evaluar la expresión, debe de dar click en el botón "=".

A continuación, podrá ver una breve descripción de cómo funciona cada botón en la interfaz.



- **Botones numéricos:** ingresa números a la expresión. Únicamente debe de dar click en ellos.
- **Botón punto decimal:** indica si un número con el cual quiere operar contiene punto decimal; debe de dar click una sola vez en el botón.
- **Botón igual:** al terminar de escribir la expresión, debe de dar click una vez e inmediatamente verá el resultado en pantalla.
- **Botón CE:** elimina lo último que haya ingresado en su expresión cada vez que le dé click; ya sea un operador, un número, un paréntesis, etc. Una vez eliminado algo, ya no podrá recuperarlo.
- **Botón AC:** elimina todo lo que se ingresó en la calculadora. Una vez que le dé click, no podrá recuperar la expresión.
- **Botones de paréntesis:** ingresa paréntesis e indica el orden en el que desea que se evalúe la expresión.
- **Botón número negativo:** si da click una sola vez en el botón indicará que el número con el cual quiere operar, es negativo.
- **Botones de operadores:** indica la operación que desea hacer. Podrá ver en la pantalla de la calculadora inmediatamente cómo se va formando la expresión.

Conclusiones

En conclusión, se puede decir que los resultados del proyecto se obtuvieron de manera satisfactoria; a pesar de las dificultades y errores que se presentaron, se logró cumplir con todos los objetivos propuestos. Además, la elaboración del proyecto nos dio la oportunidad de potenciar las diversas habilidades de los miembros del equipo, logramos disponer de las herramientas suficientes en la modalidad virtual, como GitHub, y aplicamos los elementos aprendidos hasta el momento del curso de Estructuras de Datos. Esta experiencia fue enriquecedora en aspectos tanto académicos como personales.

Lista de referencias:

- Franco García, A. (2000, enero). La clase StringTokenizer. Física con ordenador. http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/colecciones/stringtokenizer.htm

Clase RevisorParentesis

```
/**
 * <pre>
 *
 * Clase RevisorParentesis
 *
 * Esta clase revisa que las expresiones que se introduzcan en la
calculadora, sean correctas y estén balanceadas en cuanto a paréntesis
 *
 * </pre>
 * @author Benjamín Goñi, Fernando Medina, Marcos Medina, Francisca Flores
Rosas, Mariana Luna, Rebeca Angulo Rojas
 * @version 1.0
 */

public class RevisorParentesis {
    private String cadenaRevisar;

    public RevisorParentesis() {
    }

    public RevisorParentesis(String cadenaRevisar) {
        this.cadenaRevisar = cadenaRevisar;
    }

/**  Que los paréntesis estén balanceados significa que cada símbolo de
apertura
*    tiene un símbolo de cierre correspondiente y que los paréntesis están
*    apropiadamente anidados.
*
* Este método analiza las cadenas que se introducen
*
* @return <ul>
*          <li> true: los paréntesis sí están balanceados </li>
*          <li> false: los paréntesis no están balanceados </li>
*       </ul>
*/
    public boolean analisis()
    {
        PilaA <Character> palabra;
        palabra = new PilaA<Character>();

        boolean balanceados;
        balanceados = true;

        char simbolo;
        int indice;
        indice = 0;

        while(indice < cadenaRevisar.length() && balanceados == true)
        {
            simbolo = cadenaRevisar.charAt(indice);
            if( simbolo == '(')
                palabra.push(simbolo);
            else if( simbolo == ')')
                try{
                   palabra.pop();
                }catch(Exception e){
                    balanceados = false;
                }
            indice = indice + 1;
        }
        if(balanceados && palabra.isEmpty())
                return balanceados;
            else
                return false;
    }

}
```

Clase Resuelve

```
import java.util.ArrayList;
import java.util.*;
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 * <pre>
 *
 * Clase Resuelve
 *
 * Esta clase contiene los diversos métodos necesarios para el funcionamiento de la calculadora
 *
 * </pre>
 * @author Benjamín Goñi, Fernando Medina, Marcos Medina, Francisca Flores Rosas, Mariana Luna, Rebeca
Angulo Rojas
 * @version 1.0
 */
public class Resuelve {

    public Resuelve() {
    }
     /**
      * Este método analiza si existe algún error de sintaxis en la expresión que se introduce (antes
de que ésta se convierta a postfjo)
      *
      * @param input: recibe una cadena de String
      * @return <ul>
      *         <li> true: la expresión no tiene errores de sintaxis. </li>
      *         <li> false: Si el primer o ultimo caracter son operadores, no tiene operando: error </
li>
      *     </ul>
      */
     public static boolean revisaErrores(String input) {
        boolean balanceados;
        RevisorParentesis revisor;
        revisor = new RevisorParentesis(input);
        int f = input.length()-1;
        int i;
        int contoper;
        contoper = 0;
        int linoper;
        linoper = 0;

        balanceados = revisor.analisis();

        if(balanceados == true ){
            if(input.charAt(1) == '+' || input.charAt(1) == '-' || input.charAt(1) == '*' ||
input.charAt(1) == '/' || input.charAt(f - 1) == '+' || input.charAt(f - 1) == '-' || input.charAt(f -
1) == '*' || input.charAt(f - 1) == '/' ){
                balanceados = false;
            }
            if(balanceados == true){
                for(i = 0; i < f; i++){
                    if(input.charAt(i) == '+' || input.charAt(i) == '-' || input.charAt(i) == '*' ||
input.charAt(i) == '/'){
                        contoper++;
                    }
                }

                for(i = 0; i < f; i++){

                    if(input.charAt(i) == '+' || input.charAt(i) == '-' || input.charAt(i) == '*' ||
input.charAt(i) == '/'){
                        linoper++;
                        if(input.charAt(i) == ')' || input.charAt(i) == '(' ){
                            if(i != 1 && i != f - 1){
                                if(input.charAt(i) == ')' && Character.isDigit(input.charAt(i + 2))){
                                    balanceados = false;
                                }
                                else if(input.charAt(i) == '(' && Character.isDigit(input.charAt(i -
2))){

                                    balanceados = false;
                                }
                            }
                        }

                        if(linoper < contoper && balanceados){
                            if(input.charAt(i + 3) == '+' || input.charAt(i + 3) == '-' ||
input.charAt(i + 3) == '*' || input.charAt(i + 3) == '/' ){
                                balanceados = false;
                            }
                        }
                    }
                }
            }
        }
        return balanceados;
    }
```

```
     /**
      * Este método convierte una cadena de String revisada a un arreglo en el que cada casilla existe
un componente de la expresión, el StringTokenizer diferencia los componentes
      * @param cadena: recibe un String
      * @return <ul>
      *         <li> arr: la expresión separada por casillas. </li>
      *     </ul>
      */
    public static String[] agregarCad(String cadena){
        String[] arr = new String[cadena.length()];

        StringTokenizer tokens = new StringTokenizer(cadena," ");

        int i=0;
        while(tokens.hasMoreTokens()){
            String str=tokens.nextToken();
            arr[i] = str;
            i++;
        }
        return arr;

    }

     /**
      * Este método analiza los componentes y determina qué operador se utiliza
      *
      * @param cad: recibe un String
      *
      * @return <ul>
      *         <li> operadorActual </li>
      *     </ul>
      */
    public static String operador(String cad) { //método útil para hacer estas preguntas en otros
métodos
        String operadorActual;
        switch(cad) {
            case "+":
                    operadorActual = "+";
                    break;
            case "-":
                    operadorActual = "-";
                    break;
            case "*":
                    operadorActual = "*";
                    break;
            case "/":
                    operadorActual = "/";
                    break;
            case "(":
                    operadorActual = "(";
                    break;
            case ")":
                    operadorActual = ")";
                    break;
            default:
                    operadorActual = "num";
                    break;
               }
        return operadorActual;
    }

     /**
      * Este método analiza y asigna la prioridad de operadores en la expresión
      *
      * @param c: recibe un String
      *
      * @return <ul>
      *         <li> la prioridad del operador </li>
      *     </ul>
      */
    public static int getPrioridad(String c){
        int ans = 0;
        switch(c){
            case ("+"):
                ans = 1;
                break;
            case("-"):
                ans = 1;
                break;
            case("*"):
                ans = 2;
                break;
            case("/"):
                ans = 2;
                break;
        }
        return ans;
    }
```

```
     /**
      * Este método analiza la jerarquía tomando en cuenta la prioridad de cada operador que compone
      *
      * @param cadena: recibe un String
      * @param elemPila: recibe un String con elementos de la pila
      *
      * @return <ul>
      *         <li> false: el elemento de la pila dada que tenga menor jerarquía que el otro </li>
      *         <li> true: el elemento de la pila dada que tenga mayor jerarquía que el otro </li>
      *     </ul>
      */
    public static  boolean jerarquiaOperaciones(String cadena, String elemPila) {
        boolean mayorJerarquia;
        if( getPrioridad(operador(cadena)) <= getPrioridad(operador(elemPila))) {
            mayorJerarquia = false;
        }else {
                        mayorJerarquia = true;
                }
        return mayorJerarquia;
    }

     /**
      * Este método convierte a postfijo la expresión, analiza con una cadena para determinar la
jeraquía y divide las posibilidades en casos específicos para después resolverlos
      *
      * @param cad: recibe un arreglo de String
      *
      * @return <ul>
      *         <li> resultado: la expresión que desea evaluarse </li>
      *     </ul>
      */
    public static String convierteAPostfijo(String[] cad){
        String resultado;
        PilaA<String> pila;
        pila = new PilaA();
        int i = 0;
        resultado = " ";
        while(!(cad[i] == null)) {
            if(operador(cad[i]).equals("num")){
                resultado = resultado + cad[i];
                resultado += " ";
            }
            else{

                if(operador(cad[i]).equals("("))
                    pila.push(cad[i]);

                else if(operador(cad[i]).equals(")")) {
                    while(!pila.peek().equals("(")) {
                        resultado = resultado + pila.pop();
                        resultado += " ";
                    }
                    pila.pop();
                            }
                else {
                    if(!pila.isEmpty()) {
                        if(jerarquiaOperaciones(cad[i],pila.peek())) {
                                pila.push(cad[i]);
                        }
                        else {
                            resultado = resultado + pila.pop();
                            resultado += " ";
                            if(!pila.isEmpty()) {
                                while(!pila.isEmpty() && !jerarquiaOperaciones(cad[i],pila.peek())) {
                                        resultado = resultado + pila.pop();
                                        resultado += " ";
                                }
                            }
                            pila.push(cad[i]);
                        }
                    }
                    else {
                        pila.push(cad[i]);
                    }
                }
            }
            i++;
        }
        while(!pila.isEmpty()) {
            resultado = resultado + pila.pop();
            resultado += " ";
        }
        return resultado;
    }
```

```
   /**
      * Este método resuelve la expresión ya convertida a postfija
      *
      * @param cad: recibe un String
      * @return <ul>
      *             <li> operacion: el resultado final de la expresión </li>
      *         </ul>
      */
     public static double resolverPosfijo(String cad[]){
         PilaA<Double> pila;
         pila = new PilaA();
         double num1, num2, num;
         num = 0;
         double operacion = 0;
         int i = 0;
         while(!(cad[i] == null)){
             if(cad[i].equals("+")) {
                 num1 = pila.pop();
                 num2 = pila.pop();
                 operacion = num1 + num2;
                 pila.push( operacion);
             }
             else if(cad[i].equals("-")) {
                 num1 = pila.pop();
                 num2 = pila.pop();
                 operacion = num2 - num1;
                 pila.push(operacion);

             }
             else if(cad[i].equals("*")){
                 num1 = pila.pop();
                 num2 = pila.pop();
                 operacion = num1*num2;
                 pila.push(operacion);
             }

             else if(cad[i].equals("/")){
                 num1 = pila.pop();
                 num2 = pila.pop();

                 try{
                     operacion = num2/num1;
                 }
                 catch(Exception e ){
                     System.out.println("Math Error");

                 }
                 pila.push(operacion);
             }
             else {
                 num = Double.parseDouble(cad[i]);
                 pila.push(num);
             }
             i++;
         }
        return operacion;
     }
}
```

Clase Caratula

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */


/**
 * <pre>
 *
 * Clase Caratula
 *
 * Esta clase define la interfaz gráfica de la calculadora, incluyendo la programación necesaria de
los botones y escuchadores
 *
 * </pre>
 * @author Benjamín Goñi, Fernando Medina, Marcos Medina, Francisca Flores Rosas, Mariana Luna, Rebeca
Angulo Rojas
 * @version 1.0
 */


import javax.swing.JOptionPane;
import java.util.*;
public class Caratula extends javax.swing.JFrame {

    /**
     * Creates new form Caratula
     */
    public Caratula() {
        initComponents();
        this.setLocationRelativeTo(null);
    }

    /**
     * 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() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        btn1 = new javax.swing.JButton();
        btn4 = new javax.swing.JButton();
        btn7 = new javax.swing.JButton();
        btn2 = new javax.swing.JButton();
        btn8 = new javax.swing.JButton();
        btn5 = new javax.swing.JButton();
        btn3 = new javax.swing.JButton();
        btn9 = new javax.swing.JButton();
        btn6 = new javax.swing.JButton();
        btnCero = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        btnDecimal = new javax.swing.JButton();
        btnBorraUno = new javax.swing.JButton();
        btnBorraTodo = new javax.swing.JButton();
        btnMas = new javax.swing.JButton();
        btnMenos = new javax.swing.JButton();
        btnPor = new javax.swing.JButton();
        btnDiv = new javax.swing.JButton();
        btnIgual = new javax.swing.JButton();
        btnMasMenos = new javax.swing.JButton();
        btnParentesisIzq = new javax.swing.JButton();
        btnParentesisDer = new javax.swing.JButton();
        TextResultado = new javax.swing.JTextField();
        jScrollPane2 = new javax.swing.JScrollPane();
        TextInput = new javax.swing.JTextArea();

        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(38, 50, 56));

        btn1.setBackground(new java.awt.Color(55, 71, 79));
        btn1.setForeground(new java.awt.Color(236, 239, 241));
        btn1.setText("1");
        btn1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btn1ActionPerformed(evt);
            }
        });
```

```
btn4.setBackground(new java.awt.Color(55, 71, 79));
btn4.setForeground(new java.awt.Color(236, 239, 241));
btn4.setText("4");
btn4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn4ActionPerformed(evt);
    }
});

btn7.setBackground(new java.awt.Color(55, 71, 79));
btn7.setForeground(new java.awt.Color(236, 239, 241));
btn7.setText("7");
btn7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn7ActionPerformed(evt);
    }
});

btn2.setBackground(new java.awt.Color(55, 71, 79));
btn2.setForeground(new java.awt.Color(236, 239, 241));
btn2.setText("2");
btn2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn2ActionPerformed(evt);
    }
});

btn8.setBackground(new java.awt.Color(55, 71, 79));
btn8.setForeground(new java.awt.Color(236, 239, 241));
btn8.setText("8");
btn8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn8ActionPerformed(evt);
    }
});

btn5.setBackground(new java.awt.Color(55, 71, 79));
btn5.setForeground(new java.awt.Color(236, 239, 241));
btn5.setText("5");
btn5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn5ActionPerformed(evt);
    }
});

btn3.setBackground(new java.awt.Color(55, 71, 79));
btn3.setForeground(new java.awt.Color(236, 239, 241));
btn3.setText("3");
btn3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn3ActionPerformed(evt);
    }
});

btn9.setBackground(new java.awt.Color(55, 71, 79));
btn9.setForeground(new java.awt.Color(236, 239, 241));
btn9.setText("9");
btn9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn9ActionPerformed(evt);
    }
});

btn6.setBackground(new java.awt.Color(55, 71, 79));
btn6.setForeground(new java.awt.Color(236, 239, 241));
btn6.setText("6");
btn6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn6ActionPerformed(evt);
    }
});

btnCero.setBackground(new java.awt.Color(55, 71, 79));
btnCero.setForeground(new java.awt.Color(236, 239, 241));
btnCero.setText("0");
btnCero.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCeroActionPerformed(evt);
    }
});

jLabel1.setText("Calculadora");

btnDecimal.setBackground(new java.awt.Color(55, 71, 79));
btnDecimal.setForeground(new java.awt.Color(236, 239, 241));
btnDecimal.setText(".");
btnDecimal.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDecimalActionPerformed(evt);
    }
});
```

```
btnBorraUno.setBackground(new java.awt.Color(55, 71, 79));
btnBorraUno.setForeground(new java.awt.Color(236, 239, 241));
btnBorraUno.setText("CE");
btnBorraUno.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnBorraUnoActionPerformed(evt);
    }
});

btnBorraTodo.setBackground(new java.awt.Color(55, 71, 79));
btnBorraTodo.setForeground(new java.awt.Color(236, 239, 241));
btnBorraTodo.setText("AC");
btnBorraTodo.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnBorraTodoActionPerformed(evt);
    }
});

btnMas.setBackground(new java.awt.Color(55, 71, 79));
btnMas.setForeground(new java.awt.Color(236, 239, 241));
btnMas.setText("+");
btnMas.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMasActionPerformed(evt);
    }
});

btnMenos.setBackground(new java.awt.Color(55, 71, 79));
btnMenos.setForeground(new java.awt.Color(236, 239, 241));
btnMenos.setText("-");
btnMenos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMenosActionPerformed(evt);
    }
});

btnPor.setBackground(new java.awt.Color(55, 71, 79));
btnPor.setForeground(new java.awt.Color(236, 239, 241));
btnPor.setText("*");
btnPor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPorActionPerformed(evt);
    }
});

btnDiv.setBackground(new java.awt.Color(55, 71, 79));
btnDiv.setForeground(new java.awt.Color(236, 239, 241));
btnDiv.setText("/");
btnDiv.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDivActionPerformed(evt);
    }
});

btnIgual.setBackground(new java.awt.Color(55, 71, 79));
btnIgual.setForeground(new java.awt.Color(236, 239, 241));
btnIgual.setText("=");
btnIgual.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnIgualActionPerformed(evt);
    }
});

btnMasMenos.setBackground(new java.awt.Color(55, 71, 79));
btnMasMenos.setForeground(new java.awt.Color(236, 239, 241));
btnMasMenos.setText("+/-");
btnMasMenos.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMasMenosActionPerformed(evt);
    }
});

btnParentesisIzq.setBackground(new java.awt.Color(55, 71, 79));
btnParentesisIzq.setForeground(new java.awt.Color(236, 239, 241));
btnParentesisIzq.setText("(");
btnParentesisIzq.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnParentesisIzqActionPerformed(evt);
    }
});

btnParentesisDer.setBackground(new java.awt.Color(55, 71, 79));
btnParentesisDer.setForeground(new java.awt.Color(236, 239, 241));
btnParentesisDer.setText(")");
btnParentesisDer.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnParentesisDerActionPerformed(evt);
    }
});

TextResultado.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
TextResultado.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        TextResultadoActionPerformed(evt);
    }
});
```

```
        TextInput.setColumns(20);
        TextInput.setRows(5);
        jScrollPane2.setViewportView(TextInput);

        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(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(btn1, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btn2, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btn3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btnMas, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btnDiv, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btnBorraTodo, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING, false)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(btnCero, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnDecimal, javax.swing.GroupLayout.PREFERRED_SIZE,
44, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnIgual, javax.swing.GroupLayout.PREFERRED_SIZE,
44, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnMasMenos, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnParentesisIzq,
javax.swing.GroupLayout.PREFERRED_SIZE, 44, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnParentesisDer,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(btn4, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btn5, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btn6, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnMenos, javax.swing.GroupLayout.PREFERRED_SIZE,
44, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                                    .addComponent(btnPor, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(btnBorraUno, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(TextResultado, javax.swing.GroupLayout.PREFERRED_SIZE, 292,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(0, 0, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 294,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel1)
                .addGap(24, 24, 24)
                .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(TextResultado, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(btn1, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btn2, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btn3, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnMas, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnDiv, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnBorraTodo, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(btn4, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btn5, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btn6, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnMenos, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnBorraUno, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnPor, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
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.BASELINE)
                        .addComponent(btn7, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(btn8, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(btn9, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(btnParentesisIzq, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnParentesisDer, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(btnCero, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnDecimal, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnIgual, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnMasMenos, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(6, 6, 6))
        );

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

    private void btn9ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn9ActionPerformed
        TextInput.setText(TextInput.getText()+"9");
    }//GEN-LAST:event_btn9ActionPerformed

    private void btn1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn1ActionPerformed
        TextInput.setText(TextInput.getText()+"1");
    }//GEN-LAST:event_btn1ActionPerformed

    private void btn2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn2ActionPerformed
        TextInput.setText(TextInput.getText()+"2");
    }//GEN-LAST:event_btn2ActionPerformed

    private void btn3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn3ActionPerformed
        TextInput.setText(TextInput.getText()+"3");
    }//GEN-LAST:event_btn3ActionPerformed

    private void btn4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn4ActionPerformed
        TextInput.setText(TextInput.getText()+"4");
    }//GEN-LAST:event_btn4ActionPerformed

    private void btn5ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn5ActionPerformed
        TextInput.setText(TextInput.getText()+"5");
    }//GEN-LAST:event_btn5ActionPerformed

    private void btn6ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn6ActionPerformed
        TextInput.setText(TextInput.getText()+"6");
    }//GEN-LAST:event_btn6ActionPerformed
```

```
    private void btn7ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn7ActionPerformed
        TextInput.setText(TextInput.getText()+"7");
    }//GEN-LAST:event_btn7ActionPerformed

    private void btn8ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btn8ActionPerformed
        TextInput.setText(TextInput.getText()+"8");
    }//GEN-LAST:event_btn8ActionPerformed

    private void btnCeroActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnCeroActionPerformed
        TextInput.setText(TextInput.getText()+"0");
    }//GEN-LAST:event_btnCeroActionPerformed
//punto
    private void btnDecimalActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnDecimalActionPerformed
        String s= TextInput.getText();
        if(s.length()==1 && s.equals("-"))
            TextInput.setText("-0.");
        if(s.length()<=0)
            TextInput.setText("0.");
        else{
            switch (s.charAt(s.length()-1)) {
                case ' ':
                    TextInput.setText(TextInput.getText()+"0.");
                    break;
                case '-':
                    TextInput.setText(TextInput.getText()+"-0.");
                    break;
                default:
                    TextInput.setText(TextInput.getText()+".");
                    break;
            }
        }

    }//GEN-LAST:event_btnDecimalActionPerformed

    private void btnBorraUnoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnBorraUnoActionPerformed
        String s= TextInput.getText();
        if(s.length()>0){
            s=s.substring(0, s.length()-1);
            TextInput.setText(s);
        }
    }//GEN-LAST:event_btnBorraUnoActionPerformed

    private void btnBorraTodoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnBorraTodoActionPerformed
        TextInput.setText("");
        TextResultado.setText("");
    }//GEN-LAST:event_btnBorraTodoActionPerformed

    private void btnMasActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnMasActionPerformed
        TextInput.setText(TextInput.getText()+" + ");
    }//GEN-LAST:event_btnMasActionPerformed

    private void btnMenosActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnMenosActionPerformed
        TextInput.setText(TextInput.getText()+" - ");
    }//GEN-LAST:event_btnMenosActionPerformed

    private void btnPorActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnPorActionPerformed
        TextInput.setText(TextInput.getText()+" * ");
    }//GEN-LAST:event_btnPorActionPerformed

    private void btnDivActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnDivActionPerformed
        TextInput.setText(TextInput.getText()+" / ");
    }//GEN-LAST:event_btnDivActionPerformed

    private void btnIgualActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnIgualActionPerformed
        String cadena = TextInput.getText();
        String posf;
        double resp;
            if(Resuelve.revisaErrores(cadena)){

                posf = Resuelve.convierteAPostfijo(Resuelve.agregarCad(cadena));

                resp = Resuelve.resolverPosfijo(Resuelve.agregarCad(posf));

                TextResultado.setText("" + resp);
            }
            else{
                TextResultado.setText("Syntax Error");
            }
    }//GEN-LAST:event_btnIgualActionPerformed
```

```
    private void btnMasMenosActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnMasMenosActionPerformed

        char x;
        if(TextInput.getText().length()<=0)
            TextInput.setText("( -");
        else if (TextInput.getText().equals("( -")){
            TextInput.setText("");
        }
        else{
            x=TextInput.getText().charAt(TextInput.getText().length()-1);
            if(x != '-')
                TextInput.setText(TextInput.getText()+"( -");
            else
                TextInput.setText(TextInput.getText().substring(0, TextInput.getText().length()-3));
        }
    }//GEN-LAST:event_btnMasMenosActionPerformed


    private void btnParentesisIzqActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnParentesisIzqActionPerformed
        TextInput.setText(TextInput.getText()+" ( ");
    }//GEN-LAST:event_btnParentesisIzqActionPerformed

    private void btnParentesisDerActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnParentesisDerActionPerformed
        TextInput.setText(TextInput.getText()+" ) ");
    }//GEN-LAST:event_btnParentesisDerActionPerformed

    private void TextResultadoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_TextResultadoActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_TextResultadoActionPerformed


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

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

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

java.util.logging.Logger.getLogger(Caratula.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
        }
        //</editor-fold>
        //</editor-fold>

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

    }
```

```
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JTextArea TextInput;
    private javax.swing.JTextField TextResultado;
    private javax.swing.JButton btn1;
    private javax.swing.JButton btn2;
    private javax.swing.JButton btn3;
    private javax.swing.JButton btn4;
    private javax.swing.JButton btn5;
    private javax.swing.JButton btn6;
    private javax.swing.JButton btn7;
    private javax.swing.JButton btn8;
    private javax.swing.JButton btn9;
    private javax.swing.JButton btnBorraTodo;
    private javax.swing.JButton btnBorraUno;
    private javax.swing.JButton btnCero;
    private javax.swing.JButton btnDecimal;
    private javax.swing.JButton btnDiv;
    private javax.swing.JButton btnIgual;
    private javax.swing.JButton btnMas;
    private javax.swing.JButton btnMasMenos;
    private javax.swing.JButton btnMenos;
    private javax.swing.JButton btnParentesisDer;
    private javax.swing.JButton btnParentesisIzq;
    private javax.swing.JButton btnPor;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration//GEN-END:variables
}
```