

*26 de septiembre de 2022*

*Ernesto Eliezer Hernández Bernal*

*Ernesto Palma Molina*

*Jerónimo Deli Larios*

*Mariana Isabel Glatz Gutiérrez*

*Roberto Andrés Muller Ríos*

*Proyecto Calculadora*

**Índice**

[Descripción del problema 2](#_Toc115055417)

[Solución Diseñada. 2](#_Toc115055418)

[Pruebas. 4](#_Toc115055419)

[Limitaciones 6](#_Toc115055420)

[Mejoras y Conclusiones. 7](#_Toc115055421)

[Referencias. 8](#_Toc115055422)

[Apéndice. 9](#_Toc115055423)

**Descripción del problema.**

* Objetivo:

El objetivo principal del proyecto es crear una “Calculadora” funcional, que pueda realizar las operaciones más básicas.

* Requisitos:

Debe evaluar sumas, restas, divisiones y productos, así como la operación de la potencia con números negativos, números con punto decimal, números racionales, números irracionales y números enteros. El proyecto debe ser realizado en forma para ser entregado el jueves 22 de septiembre de 2022.

* Restriciones:

Se deberá entregar una interfaz gráfica que simule una calculadora y emule sus funciones. El proyecto debe ser realizado en el IDE de “Apache NetBeans” y debe contener métodos que usen pilas como su estructura principal de datos.

Debe cumplir con al menos 3 métodos generales: verificar valides de la expresión, convertir esa expresión a notación postfija y evaluar la expresión postfija para mostrar el resultado de las operaciones.

# **Solución Diseñada.**

Las principales soluciones que debemos contemplar para tener una calculadora funcional y que cumpla con los requisitos son 3.

1. Revisar la sintaxis de la expresión para que al evaluarla tenga coherencia en las operaciones y siga un orden lógico que permita ejecutar operaciones algebraicas.
2. Convertir la expresión ya revisada con anterioridad a su forma en Postfija teniendo en cuenta el orden de jerarquía de los operadores y paréntesis.
3. Dada la expresión en su forma Postfija, poder evaluar la expresión de una manera eficaz y que muestre el resultado adecuado.

Ahora, ¿Por qué es preferible convertir de notación Infija a notación Postfija?

“Las expresiones postfijas son, generalmente, más fáciles de evaluar que las expresiones infijas, porque no es necesario tener en cuenta ni reglas de precedencia, ni paréntesis. El orden de los valores y de los operadores en la expresión es suficiente para determinar el resultado. A menudo, los compiladores de los lenguajes de programación en los entornos de ejecución utilizan expresiones postfijas en sus cálculos internos, por esa misma razón. El proceso de evaluación de una expresión postfija puede enunciarse mediante una única regla simple: analizando la expresión de izquierda a derecha, hay que aplicar cada operación a los dos operandos inmediatamente precedentes y sustituir el operador por el resultado. Al final, lo que nos quedará será el resultado total de la expresión.” – John Lewis.

Para resolver los problemas anteriormente mencionados, necesitamos emplear tres clases y hacer uso de la estructura abstracta “Pilas” en ellas para que sea un código eficiente y bien estructurado.

En el apéndice, junto al código completo del proyecto, se presentan los diagramas UML que servirán de base para el diseño de las clases y se incluyen también los de la estructura Pilas como referencia.

**Pruebas.**

Se realizaron diversas pruebas para confirmar el correcto funcionamiento de nuestros códigos y nuestra GUI.

* Expresión con caracteres que no son dígitos
* Expresión con errores de sintaxis (balanceo de paréntesis, letras y números, expresiones erróneas.)
* Expresión con positivos y negativos
* Expresión con todos los operadores
* Expresión con diversas jerarquías de operaciones (paréntesis que cambiaban la jerarquía)
* Expresión con operadores juntos
* Expresión con un operador junto al símbolo menos que indicaba número negativo (así se decidió en equipo)
* Expresión con división entre cero y cero a la potencia cero

Durante el transcurso de las pruebas se notó la ausencia o ineficiencia de algunos métodos. Por ejemplo, en la clase EvaluacionSintaxis se tuvo que agregar varios métodos para considerar la mayor parte de casos posibles en los que había error de escritura. Ejemplo de lo anterior es anotar caracteres que no fueran números u operadores. También el caso “ ( ( 8 +6 ) ) ”, en el primer intento arrojó error debido a que consideraba el segundo ‘ ( ‘ como algo diferente a un operador y se tuvo que especificar el caso “ ( ( “ para seguir con la expresión. Otro caso fue que de manera ambiciosa hicimos válida una operación junto a ‘ – ‘ para indicar números negativos, es decir, el usuario no tenía necesidad de especificar con paréntesis los números negativos, lo que hace posible expresiones como “ 7 \* - 2 ”. Sin embargo, dejar esta expresión como válida arrojaba otros errores como “–()-()”. Otro caso notable fue el de indicar inválida la expresión “ 6 ( 8 ”, no puede haber paréntesis sin operador a lado que indique la operación a realizar. Pudimos ambicionar más e indicar que en esos casos tenía que multiplicar los números, pero entonces se tenía que especificar que fueran números o incluso “ ( 4 ) ( 5 ) ” lo tomara como válido y era reestructurar parte de la conversión a postfija y la parte de la evaluación ¿Cómo se podría saber si el caso “ ( 8 ) ( 9 ) ( 7 ) ” es una triple multiplicación o son 3 números juntos que en postfija harán otras operaciones?. En teoría, por el análisis previo de la expresión, podríamos considerar que nos indica una triple multiplicación, pero, como en los métodos dejamos paréntesis para agrupar los números, era aumentar las condiciones y se perdería la limpieza del código que llevamos, que cabe aclarar, se nos ocurrió tarde esa implementación.

En las otras dos clases lo más notable fue precisamente agrupar números enteros mayores a un dígito, pues usamos métodos que analizan caracter por caracter y reaccionan con base en ese carácter ¿Cómo indicar que es 23 o 345.343? Por eso recurrimos a los paréntesis de tal forma que un operador nos indique el tamaño del número previo a ese operador, por ejemplo “ 34.56 - 5664” al convertirse en postfija quedaría “ ( 34.56 ) ( 5664) - ” lo que muestra en parte el porqué no nos aventuramos a señalar una multiplicar con “ ( ) ( ) ”. Cabe destacar que dejamos hasta la evaluación de Postfija la señalización de las operaciones “x / 0“ y “0 ^ 0” como erróneas, pues es ahí donde nos da los posibles resultados inválidos. Por practicidad, lanzamos una excepción con el mensaje “Error” en la interfaz hasta que es ejecutada. Un ejemplo de lo anterior es “ (2 – 2 ) ^ ( 5 / 5 – 1 )” que al analizar el texto no tiene errores de sintaxis, pero al ser evaluada queda “ 0 ^ 0 ” que no es válido. Un último caso notable fue que admitiera diferentes maneras de expresar operaciones de resta o diferencia entre paréntesis y arrojara resultados correctos. Por ejemplo, “-(3+4) - (6+7)”, “-(8)-(7)+(6)”, “-(8-7+6)”, “-(8)-(7)+(6)”, “((-8)+(-7)+(6))”, ”(((-8)))-(((7)))+((6)) ”

**Limitaciones.**

La Calculadora no está desarrollada para revolver expresiones matemáticas avanzadas:

* Calculo de funciones
* Calculo de derivadas o integrales
* Calculo de raíz cuadrada
* Calculo de raíz cubica
* Calculo de funciones trigonométricas
* Generar estadísticas
* Exhibir gráficos

Además, no contemplamos por cuestión de tiempo y especificaciones del proyecto: multiplicación indicada como “ ( 2 ) ( 3 ) ”, valuación de números complejos, indicar el error específico de la sintaxis o de la operación matemática, y, probablemente, algún caso que no se nos haya ocurrido durante el lapso permitido.

Otra limitación fueron las pruebas Junit, en las que no pudimos contemplar todos los casos pues podrían extenderse tanto como quisiéramos. También, en la clase que convierte a postfija no había manera de ocupar adecuadamente las pruebas, pues cualquier expresión la modificaba, incluso con paréntesis mal balanceados o caracteres diferentes y, analizando el contexto, tiene sentido, pues así la diseñamos, con plena confianza en el método anterior que evalúa que la sintaxis de la expresión sea correcta. Incluso, aunque quisiéramos que el método de conversión postfija indicara si la expresión postfija es correcta tendríamos que evaluarla para saber si arroja un resultado válido o tendríamos que usar una comparación con un molde o la expresión postfija correcta, por ejemplo, “ 2 - 3 + 7 ” convertido quedaría “(2) (3) – (7) + “ y comparamos con “ (número)(número)operador (número)operador” pero si tuviéramos “(número)(número)(número)operador operador “ también sería una expresión válida; en todo caso, lo que podríamos hacer es garantizar un mínimo correcto, en el que al inicio de postfija haya 2 números juntos y al final un operador, es decir, “ (número)(número) . . . operador ”.

**Mejoras y Conclusiones.**

Además de las mejoras obvias, como calcular operaciones más complejas, el proyecto podría mejorarse en el apartado de los errores que arroja al usuario. Por ejemplo, mostrar el error específico en la sintaxis de la expresión o el resultado erróneo. También podríamos mejorar en señalar como válidas las operaciones ‘+-‘, ‘/-’,’\*-’,’^-’

Nuestra experiencia en general fue buena. Como cualquier proyecto en equipo, tiene su grado de complejidad el acoplarse a los ritmos y formas de trabajo de sus integrantes. A pesar de ello, logramos identificar las habilidades de cada uno de nosotros, y establecimos roles de trabajo para reunir con más rapidez y eficiencia los objetivos planteados.

Siempre es un reto colaborar con personas nuevas: cada una tiene ideas, conocimientos y habilidades distintas que al combinarse permiten una mejoría en los proyectos, siempre y cuando haya comunicación y organización.

# **Referencias.**

* [Guardati Buemo, Silvia del Carmen](javascript:open_window(%22https://hammurabi.itam.mx:443/F/S77FIJ46188NV57YVS18JNLHCVB5L5JDQGU4P43FNTNQEIE9Q3-94115?func=service&doc_number=000264283&line_number=0009&service_type=TAG%22);). (2015). *Estructuras de datos básicas: programación orientada a objetos con Java*. 1ª ed. México, D. F. Alfaomega Grupo Editor.
* John Lewis, Joseph Chase. (2006). *Estructuras de datos con Java: diseño de estructuras y algoritmos.* 2a ed. Madrid: Pearson Addison Wesley.

# **Imágenes.**

* Recuperado: <https://www.google.com/url?sa=i&url=http%3A%2F%2Fescolar.itam.mx%2Fcentro_tesis%2Fcentro_trabajo_logo.php&psig=AOvVaw2ADbA_HoOGQospeYlmxs72&ust=1664323782185000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCJCa_bTXs_oCFQAAAAAdAAAAABAD>

# **Apéndice.**

***Javadoc***

***Proyecto-Calculadora/Calculadora/dist/javadoc/index.html***

***Diagramas***

|  |
| --- |
| <<interface>>  PilaADT |
| +pop(): T  +push (T dato): void  +peek(): T  +isEmpty(): boolean |

|  |
| --- |
| PilaA |
| -pila[]: T  -tope: int |
| +pop(): T  +push (T dato): void  +peek(): T  +isEmpty(): boolean |

|  |
| --- |
| EvaluacionSintaxis |
| +EvaluacionSintaxis()  +noCaracteresNoAceptados(String formula):boolean  +parentesisBalanceados(String formula): boolean  +esNumero(char c): boolean  +esOperador(char c): boolean  +esOperadorSinMenos(char c): boolean  +noDobleOperador(String formula):boolean  +noDobleDecimal(String formula): boolean  + noOperadorAntesDeParentesis (String formula): boolean  +expresionValida(String formula): boolean |

|  |
| --- |
| Posfijo |
| +Posfijo()  + pesoOp(Charactern):  + cadena(String expresion): String |

|  |
| --- |
| EvaluacionPostfija |
| +EvaluacionPostfija()  +evaluaPost(String cadena): double  +operador(char cadena):boolean  +parentesis(char cadena): boolean |

Ya que tenemos las clases en UML con sus métodos, es preciso también incluir las clases en UML de nuestra interfaz gráfica:

|  |
| --- |
| CalculadoraDemo |
|  |
| +main() |

|  |
| --- |
| CalculadoraGUI |
| +keyPanel  +textPanel  +equalPanel  +0-9button  +inputTF  +CalculadoraGUI()  +display()  +ClearListener()  +KeyListener()  +EqualListener() |

* Se podrá notar la diferencia entre nombres de métodos y parámetros, lo que resalta la diferencia en la forma de trabajar de cada miembro del equipo.

***Código***

1. EvaluacionSintaxis

*/\*\**

*\*Clase para evaluar que una sintaxis matematica este bien escrita*

*\* @author jeronimo*

*\*/*

**public** **class** **EvaluacionSintaxis** {

*/\*\**

*\* Constructor por omisión*

*\*/*

**public** EvaluacionSintaxis(){

}

*/\*\**

*\* Determina si la expresion tiene caracteres que no sean números, operadores, puntos o paréntesis*

*\**

*\* @param formula String de la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si la expresión no tiene caracteres no aceptados*

*\* <li>false: si la expresion contiene caracteres no aceptados*

*\* </ul>*

*\*/*

**public** **boolean** noCaracteresNoAceptados(String formula){

**boolean** res= **true**;

**int** i=0;

Character c;

**while**(i<formula.length() && res){

c= formula.charAt(i);

**if**(!esNumero(c)&&!esOperador(c)&& c!='(' && c!=')' && c!='.' || esOperadorSinMenos(formula.charAt(0))){

res=**false**;

}

i++;

}

**return** res;

}

*/\*\**

*\* Determina si una expresión matemática tiene los parámetros balanceados*

*\**

*\* @param formula String de la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si los paréntesis estan balanceados*

*\* <li>false: si los paréntesis no estan balanceados*

*\* </ul>*

*\*/*

**public** **boolean** parentesisBalanceados(String formula){

**boolean** res=**true**;

Character c;

PilaA<Character> pila= **new** PilaA();

**int** i=0;

**while**(i<formula.length() && res){

c= formula.charAt(i);

**if**(c=='('){

pila.push(c);

}

**else** if(c==')'){

**if**(pila.isEmpty()){

res=**false**;

}

**else**{

pila.pop();

}

}

i++;

}

**return** res && pila.isEmpty();

}

*/\*\**

*\* Método que nos permite determinar si un caracter es un número*

*\**

*\* @param c el caracter que queremos evaluar*

*\* @return <ul>*

*\*<li>true: si el character es un número*

*\*<li>false: si el character no es un número*

*\*</ul>*

*\*/*

**public** **boolean** esNumero(**char** c){

**boolean** res= **false**;

**if** (c=='1' || c=='2' || c=='3' || c=='4'|| c=='5' || c=='6' || c=='7' || c=='8' || c=='9' || c=='0'){

res= **true**;

}

**return** res;

}

*/\*\**

*\* Método que nos permite determinar si un character es un operador ('+','/','-','\*','^')*

*\**

*\* @param c el caracter que queremos evaluar*

*\* @return <ul>*

*\*<li>true: si el character es '+','/','-','\*','^';*

*\*<li>false: si el character no es '+','/','-','\*','^';*

*\*</ul>*

*\*/*

**public** **boolean** esOperador(**char** c){

**boolean** res= **false**;

**if** (c=='\*' || c=='/' || c=='+' || c=='-' || c=='^'){

res= **true**;

}

**return** res;

}

*/\*\**

*\* Método que nos permite determinar si un character es un operador sin incluir al menos ('+','/','\*','^')*

*\**

*\* @param c el caracter que queremos evaluar*

*\* @return <ul>*

*\*<li>true: si el character es '+','/','\*','^';*

*\*<li>false: si el character no es '+','/','\*','^';*

*\*</ul>*

*\*/*

**public** **boolean** esOperadorSinMenos(**char** c){

**boolean** res= **false**;

**if** (c=='\*' || c=='/' || c=='+' ||c=='^'){

res= **true**;

}

**return** res;

}

*/\*\**

*\* Método para revisar que no existan dos operados juntos en una expresión*

*\**

*\* @param formula String que representa la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si no hay dos operadores juntos en la expresión*

*\* <li>false: si hay dos operados juntos en la expresión*

*\* </ul>*

*\*/*

**public** **boolean** noDobleOperador(String formula){

**boolean** res= **true**;

Character c;

Character c2;

*//nunca debe llegar a formula.length ya que checamos dos valores, uno mayor al contador*

**int** i=0;

**while**(i<formula.length()-1 && res){

c= formula.charAt(i);

c2= formula.charAt(i+1);

**if**(esOperadorSinMenos(c) && esOperadorSinMenos(c2)){

res= **false**;

}

i++;

}

**return** res;

}

*/\*\**

*\* Método para revisar que no existan dobles decimales*

*\**

*\* @param formula String que representa la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si no hay doble decimal*

*\* <li>false: si hay doble decimal*

*\* </ul>*

*\*/*

**public** **boolean** noDobleDecimal(String formula){

**boolean** res= **true**;

Character c;

PilaA<Character> pila= **new** PilaA();

**int** i=0;

**while**(i<formula.length() && res){

c= formula.charAt(i);

**if**(c=='.'){

**if**(pila.isEmpty()){

pila.push(c);

}

**else**{

res=**false**;

}

}

**else**{

**if**(esOperador(c)&& !pila.isEmpty()){

pila.pop();

}

}

i++;

}

**return** res;

}

*/\*\**

*\* Método para revisar que no existan paréntesis sin operador*

*\**

*\* @param formula String que representa la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si no hay paréntesis sin operador*

*\* <li>false: si hay paréntesis sin operador*

*\* </ul>*

*\*/*

**public** **boolean** noOperadorAntesDeParentesis(String formula){

**boolean** res= **true**;

Character c;

Character c2;

*//empezamos en 1 ya que queremos revisar el del contador y el anterior*

**int** i=1;

**while**(i<formula.length() && res){

c= formula.charAt(i);

c2= formula.charAt(i-1);

**if**(c=='(' && !esOperador(c2) && c2!='('){

res= **false**;

}

i++;

}

**return** res;

}

*/\*\**

*\* Método para evaluar que la expresión este bien escrita*

*\**

*\* @param formula String que representa la expresión que queremos evaluar*

*\* @return <ul>*

*\* <li>true: si esta bien escrita*

*\* <li>false: si no esta bien escrita*

*\* </ul>*

*\*/*

**public** **boolean** expresionValida(String formula){

**boolean** res= **false**;

**if**(parentesisBalanceados(formula) && noDobleOperador(formula) && noDobleDecimal(formula) && noOperadorAntesDeParentesis(formula) && noCaracteresNoAceptados(formula)){

res=**true**;

}

**return** res;

}

}

1. Posfijo

*/\*\**

*\* Definición de clase Postfijo que convierte expresión en postfija*

*\* @author m-gla*

*\*/*

**public** **class** **Posfijo** {

*/\*\**

*\* Constructor por omisión*

*\*/*

**public** Posfijo(){

}

*/\*\**

*\* Método para asignar el peso del los carácteres de los operadores.*

*\* @param n*

*\* @return*

*\*/*

**public** **int** pesoOp(Character n){

**int** valor = 0;

**if**(**null** != n)**switch** (n) {

**case** '+':

**case** '-':

valor = 1;

**break**;

**case** '\*':

**case** '/':

valor = 2;

**break**;

**case** '^':

valor = 3;

**break**;

**default**:

**break**;

}

**return** valor;

}

*/\*\**

*\* Método que regresa cadena de caracteres en formato postfija*

*\* @param expresion*

*\* @return String*

*\*/*

**public** String cadena(String expresion){

String posfija = "";

PilaADT <Character> pila = **new** PilaA();

**int** i = 0;

Character c;

**if**(!Character.isDigit(expresion.charAt(0)))

expresion="0"+expresion;

expresion=expresion.replace("(-", "(0-");

expresion=expresion.replace(")-", ")+0-");

**while**(i < expresion.length()){

c = expresion.charAt(i);

**if**((i == 0 && expresion.charAt(i) == '-') || Character.isDigit(c) || c == '.' || (c == '-' && !Character.isDigit(expresion.charAt(i-1)))){

posfija += '(';

posfija += c;

**int** j = i + 1;

**while**(j < expresion.length() && (Character.isDigit(expresion.charAt(j)) || expresion.charAt(j) == '.')){

posfija += expresion.charAt(j);

j++;

}posfija += ')';

i = j - 1;

}

**else** if(!pila.isEmpty()){

**if**(c == '('){

pila.push(c);

}**else** **if**(c == ')'){

**while**(pila.peek() != '('){

posfija += pila.pop();

}

pila.pop();

}**else** **if**(pila.peek() == '(' || pesoOp(c) > pesoOp(pila.peek())){

pila.push(c);

}**else** **if**(pesoOp(c) <= pesoOp(pila.peek())){

**while**(!pila.isEmpty() && pila.peek()!= '(' && pesoOp(pila.peek()) >= pesoOp(c))

posfija+= pila.pop();

pila.push(c);

}

}**else**{

pila.push(c);

}i++;

}**while**(!pila.isEmpty()){

posfija += pila.pop();

}

**return** posfija;

}

}

1. EvaluacionPostfija

*/\*\**

*\**

*\* @author Ernesto Palma*

*\* \* Definiciòn de la clase Evaluación de Postfija que usa pilas*

*\* @param <T>*

*\*/*

**public** **class** **EvaluacionPostfija** <T>{

*/\*\**

*\* Constructor por omisión*

*\*/*

**public** EvaluacionPostfija(){

}

*/\*\**

*\* Método que regresa el resultado de une expresión en postfija usando pilas*

*\* @param cadena*

*\* @return double*

*\*/*

**public** **double** evaluaPost(String cadena){

PilaADT<Double> pila=**new** PilaA();

**int** i=0;

**double** resul, numer;

**while**(i<cadena.length()){

**if**(**this**.operador(cadena.charAt(i))){

numer=pila.pop(); *// no pedimos condiciones, pues se garantiza postfija bien escrita la cual empieza con un número al menos*

**switch**(cadena.charAt(i)){ *// es más fácil usar casos que condicional if por ser varios*

**case**'+': resul=pila.pop() + numer; *// cada caso es una operación, aquí es una suma*

pila.push(resul);

**break**;

**case**'-': resul=pila.pop() - numer; *// resta*

pila.push(resul);

**break**;

**case**'\*': resul=pila.pop() \* numer; *// multiplicación*

pila.push(resul);

**break**;

**case**'/': **if**(numer == 0){

**throw** **new** RuntimeException("no se puede dividir entre cero");

}

**else**{

resul=pila.pop() / numer; *// división*

pila.push(resul);

}

**break**;

**case**'^': **if**(numer == 0 && pila.peek()==0){

**throw** **new** RuntimeException("no se puede hacer la operación ");

}

**else**{

resul=Math.pow(pila.pop(), numer); *// potencia*

pila.push(resul);

}

**break**;

}

}

**else** if(**this**.parentesis(cadena.charAt(i))){ *// identifica el paréntesis inicial*

StringBuilder sb=**new** StringBuilder();

**int** cont=0;

**while**(!**this**.parentesis(cadena.charAt(i+1))){ *// Agrupa lo que está dentro de los paréntesis como un número*

sb.append(cadena.charAt(i+1));

i++;

cont++;

}

pila.push(Double.parseDouble(sb.toString())); *// agrega el número a la pila*

i++;

}

i++;

}

**return** Math.round(pila.peek()\*10000)/10000d;

}

*/\*\**

*\* Método que indica si un caracter de una cadena de texto es un operador*

*\* @param cadena*

*\* @return boolean*

*\*/*

**public** **boolean** operador(**char** cadena){

**boolean** resp=**false**;

**if**(cadena == '+' | cadena == '-' | cadena == '\*' | cadena == '/' | cadena == '^')

resp=**true**;

**return** resp;

}

*/\*\**

*\* Método que indica si un caracter de una cadena de texto es un paréntesis*

*\* @param cadena*

*\* @return boolean*

*\*/*

**public** **boolean** parentesis(**char** cadena){

**boolean** resp=**false**;

**if**(cadena == '(' | cadena== ')')

resp = **true**;

**return** resp;

}

}

1. interfaceCalculadora

*/\*\**

*\**

*\* @author m-gla*

*\*/*

**public** **class** **interfaceCalculadora** **extends** javax.swing.JFrame {

*/\*\**

*\* Creates new form interfaceCalculadora*

*\*/*

**public** interfaceCalculadora() {

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() {

unobutton = **new** javax.swing.JButton();

dosbutton = **new** javax.swing.JButton();

tresbutton = **new** javax.swing.JButton();

cuatrobutton = **new** javax.swing.JButton();

cincobutton = **new** javax.swing.JButton();

seisbutton = **new** javax.swing.JButton();

sietebutton = **new** javax.swing.JButton();

ochobutton = **new** javax.swing.JButton();

nuevebutton = **new** javax.swing.JButton();

cerobutton = **new** javax.swing.JButton();

puntobtton = **new** javax.swing.JButton();

igualbtton = **new** javax.swing.JButton();

porbutton = **new** javax.swing.JButton();

divisionbtton = **new** javax.swing.JButton();

sumabtton = **new** javax.swing.JButton();

menosbutton = **new** javax.swing.JButton();

jScrollPane1 = **new** javax.swing.JScrollPane();

expresiontxt = **new** javax.swing.JTextPane();

pizquierdobtton = **new** javax.swing.JButton();

pderechobtton = **new** javax.swing.JButton();

jScrollPane2 = **new** javax.swing.JScrollPane();

jTextPane2 = **new** javax.swing.JTextPane();

jLabel1 = **new** javax.swing.JLabel();

potenciabtton = **new** javax.swing.JButton();

borrarbtton = **new** javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

unobutton.setText("1");

unobutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

unobuttonActionPerformed(evt);

}

});

dosbutton.setText("2");

dosbutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

dosbuttonActionPerformed(evt);

}

});

tresbutton.setText("3");

tresbutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

tresbuttonActionPerformed(evt);

}

});

cuatrobutton.setText("4");

cuatrobutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

cuatrobuttonActionPerformed(evt);

}

});

cincobutton.setText("5");

cincobutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

cincobuttonActionPerformed(evt);

}

});

seisbutton.setText("6");

seisbutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

seisbuttonActionPerformed(evt);

}

});

sietebutton.setText("7");

sietebutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

sietebuttonActionPerformed(evt);

}

});

ochobutton.setText("8");

ochobutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

ochobuttonActionPerformed(evt);

}

});

nuevebutton.setText("9");

nuevebutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

nuevebuttonActionPerformed(evt);

}

});

cerobutton.setText("0");

cerobutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

cerobuttonActionPerformed(evt);

}

});

puntobtton.setText(".");

puntobtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

puntobttonActionPerformed(evt);

}

});

igualbtton.setBackground(**new** java.awt.Color(102, 102, 255));

igualbtton.setText("=");

igualbtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

igualbttonActionPerformed(evt);

}

});

porbutton.setBackground(**new** java.awt.Color(153, 153, 255));

porbutton.setText("x");

porbutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

porbuttonActionPerformed(evt);

}

});

divisionbtton.setBackground(**new** java.awt.Color(153, 153, 255));

divisionbtton.setText("÷");

divisionbtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

divisionbttonActionPerformed(evt);

}

});

sumabtton.setBackground(**new** java.awt.Color(153, 153, 255));

sumabtton.setText("+");

sumabtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

sumabttonActionPerformed(evt);

}

});

menosbutton.setBackground(**new** java.awt.Color(153, 153, 255));

menosbutton.setText("-");

menosbutton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

menosbuttonActionPerformed(evt);

}

});

jScrollPane1.setViewportView(expresiontxt);

pizquierdobtton.setBackground(**new** java.awt.Color(204, 204, 255));

pizquierdobtton.setText("(");

pizquierdobtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

pizquierdobttonActionPerformed(evt);

}

});

pderechobtton.setBackground(**new** java.awt.Color(204, 204, 255));

pderechobtton.setText(")");

pderechobtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

pderechobttonActionPerformed(evt);

}

});

jScrollPane2.setViewportView(jTextPane2);

jLabel1.setText("Resultados:");

potenciabtton.setBackground(**new** java.awt.Color(153, 153, 255));

potenciabtton.setText("^");

potenciabtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

potenciabttonActionPerformed(evt);

}

});

borrarbtton.setText("Borrar");

borrarbtton.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

borrarbttonActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(48, 48, 48)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**)

.addGroup(layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 198, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(borrarbtton))

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addComponent(unobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(dosbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(tresbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(cuatrobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(cincobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(seisbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

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

.addComponent(cerobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(sietebutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

.addGroup(layout.createSequentialGroup()

.addComponent(ochobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(nuevebutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(puntobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(igualbtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)))))

.addGap(18, 18, 18)

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

.addComponent(porbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(divisionbtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(menosbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(potenciabtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

.addComponent(sumabtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 90, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addComponent(pizquierdobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(pderechobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 97, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel1))))

.addContainerGap(51, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(20, Short.MAX\_VALUE)

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

.addComponent(borrarbtton, javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, **false**)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(unobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(dosbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(tresbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(divisionbtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(sumabtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(seisbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(cuatrobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(cincobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(porbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(pizquierdobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(pderechobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(ochobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(nuevebutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(sietebutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(menosbutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(cerobutton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(puntobtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(igualbtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(potenciabtton, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addGap(96, 96, 96)

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(36, 36, 36))

);

pack();

}*// </editor-fold>*

**private** **void** dosbuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"2");

}

**private** **void** cuatrobuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"4");

}

**private** **void** nuevebuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"9");

}

**private** **void** cerobuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"0");

}

**private** **void** puntobttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+".");

}

**private** **void** igualbttonActionPerformed(java.awt.event.ActionEvent evt) {

String texto,expresion,resul;

**double** resultado=0;

texto=**this**.expresiontxt.getText();

Posfijo pos = **new** Posfijo();

EvaluacionPostfija ev = **new** EvaluacionPostfija();

EvaluacionSintaxis s= **new** EvaluacionSintaxis();

**if**(!s.expresionValida(texto)){

**this**.jTextPane2.setText("Error de sintaxis");

}

**else**{

expresion=pos.cadena(texto);

**try**{

resultado=ev.evaluaPost(expresion);

**this**.jTextPane2.setText(Double.toString(resultado));

}**catch**(Exception e){

**this**.jTextPane2.setText("Error: resultado indefinido");

}

}

}

**private** **void** porbuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"\*");

}

**private** **void** divisionbttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"/");

}

**private** **void** sumabttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"+");

}

**private** **void** menosbuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"-");

}

**private** **void** pizquierdobttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"(");

}

**private** **void** pderechobttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+")");

}

**private** **void** potenciabttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"^");

}

**private** **void** unobuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"1");

}

**private** **void** seisbuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"6");

}

**private** **void** tresbuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"3");

}

**private** **void** cincobuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"5");

}

**private** **void** sietebuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"7");

}

**private** **void** ochobuttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText(**this**.expresiontxt.getText()+"8");

}

**private** **void** borrarbttonActionPerformed(java.awt.event.ActionEvent evt) {

**this**.expresiontxt.setText("");

}

*/\*\**

*\* @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(interfaceCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);

} **catch** (InstantiationException ex) {

java.util.logging.Logger.getLogger(interfaceCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);

} **catch** (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(interfaceCalculadora.class.getName()).log(java.util.logging.Level.SEVERE, **null**, ex);

} **catch** (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(interfaceCalculadora.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** interfaceCalculadora().setVisible(**true**);

}

});

}

*// Variables declaration - do not modify*

**private** javax.swing.JButton borrarbtton;

**private** javax.swing.JButton cerobutton;

**private** javax.swing.JButton cincobutton;

**private** javax.swing.JButton cuatrobutton;

**private** javax.swing.JButton divisionbtton;

**private** javax.swing.JButton dosbutton;

**private** javax.swing.JTextPane expresiontxt;

**private** javax.swing.JButton igualbtton;

**private** javax.swing.JLabel jLabel1;

**private** javax.swing.JScrollPane jScrollPane1;

**private** javax.swing.JScrollPane jScrollPane2;

**private** javax.swing.JTextPane jTextPane2;

**private** javax.swing.JButton menosbutton;

**private** javax.swing.JButton nuevebutton;

**private** javax.swing.JButton ochobutton;

**private** javax.swing.JButton pderechobtton;

**private** javax.swing.JButton pizquierdobtton;

**private** javax.swing.JButton porbutton;

**private** javax.swing.JButton potenciabtton;

**private** javax.swing.JButton puntobtton;

**private** javax.swing.JButton seisbutton;

**private** javax.swing.JButton sietebutton;

**private** javax.swing.JButton sumabtton;

**private** javax.swing.JButton tresbutton;

**private** javax.swing.JButton unobutton;

*// End of variables declaration*

}

1. Pruebas

*/\*\**

*\**

*\* @author Ernesto Palma*

*\*/*

**public** **class** **EvaluacionPostfijaTest** {

*/\*\**

*\* Constructor por omisión*

*\*/*

**public** EvaluacionPostfijaTest() {

}

@BeforeAll

**public** **static** **void** setUpClass() {

}

@AfterAll

**public** **static** **void** tearDownClass() {

}

@BeforeEach

**public** **void** setUp() {

}

@AfterEach

**public** **void** tearDown() {

}

*/\*\**

*\* Test of evaluaPost method, of class EvaluacionPostfija.*

*\*/*

@Test

**public** **void** testEvaluaPost() {

System.out.println("evaluaPost");

String cadena = "56\*89/74-(89+58)-85+(56-85)"; *//expresion larga y con diferentes jerarquias*

String cadena2="(7)+(5)\*(4)/(6)"; *// valuacion de varios parentesis*

EvaluacionPostfija instance = **new** EvaluacionPostfija();

**double** expResult = -193.6486;

**double** expResult2 = 10.3332;

**double** result = instance.evaluaPost(cadena);

**double** result2 = instance.evaluaPost(cadena2);

assertEquals(expResult, result, 0.2);

assertEquals(expResult2, result2, 0.0);

}

*/\*\**

*\* Test of operador method, of class EvaluacionPostfija.*

*\*/*

@Test

**public** **void** testOperador() {

System.out.println("operador");

**char** cadena = '{';

**char** cadena2 = '^';

EvaluacionPostfija instance = **new** EvaluacionPostfija();

**boolean** expResult = **false**; *//caso falso*

**boolean** expResult2 = **true**; *// caso verdadero*

**boolean** result = instance.operador(cadena);

**boolean** result2 = instance.operador(cadena2);

assertEquals(expResult, result);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of parentesis method, of class EvaluacionPostfija.*

*\*/*

@Test

**public** **void** testParentesis() {

System.out.println("parentesis");

**char** cadena = '4';

**char** cadena2 = '(';

EvaluacionPostfija instance = **new** EvaluacionPostfija();

**boolean** expResult = **false**; *//caso falso*

**boolean** expResult2 = **true**; *// caso verdadero*

**boolean** result = instance.parentesis(cadena);

**boolean** result2 = instance.parentesis(cadena2);

assertEquals(expResult, result);

assertEquals(expResult2, result2);

}

}

*/\*\**

*\**

*\* @author rmull*

*\*/*

**public** **class** **PosfijoTest** {

**public** PosfijoTest() {

}

@BeforeAll

**public** **static** **void** setUpClass() {

}

@AfterAll

**public** **static** **void** tearDownClass() {

}

@BeforeEach

**public** **void** setUp() {

}

@AfterEach

**public** **void** tearDown() {

}

*/\*\**

*\* Test of pesoOp method, of class Posfijo.*

*\*/*

@Test

**public** **void** testPesoOp() {

System.out.println("pesoOp");

Character n = '+';

Posfijo instance = **new** Posfijo();

**int** expResult = 1;

**int** result = instance.pesoOp(n);

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 cadena method, of class Posfijo.*

*\*/*

@Test

**public** **void** testCadena() {

System.out.println("cadena");

String expresion = "4+5";

String expresion2 = "";

Posfijo instance = **new** Posfijo();

String expResult = "(4)(5)+";*//Es un caso sencillo*

String result = instance.cadena(expresion);

assertEquals(expResult, result);

*// TODO review the generated test code and remove the default call to fail.*

fail("The test case is a prototype.");

}

}

*/\*\**

*\**

*\* @author rmull*

*\*/*

**public** **class** **EvaluacionSintaxisTest** {

**public** EvaluacionSintaxisTest() {

}

@BeforeAll

**public** **static** **void** setUpClass() {

}

@AfterAll

**public** **static** **void** tearDownClass() {

}

@BeforeEach

**public** **void** setUp() {

}

@AfterEach

**public** **void** tearDown() {

}

*/\*\**

*\* Test of noCaracteresNoAceptados method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testNoCaracteresNoAceptados() {

System.out.println("noCaracteresNoAceptados");

String formula1 = "(16)+5\*8^2-4/7";*//Expresión únicamente con caracteres aceptados*

String formula2 = "18-8+m+6/4";*//Agrego una letra para que marque false*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.noCaracteresNoAceptados(formula1);

**boolean** result2 = instance.noCaracteresNoAceptados(formula2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of parentesisBalanceados method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testParentesisBalanceados() {

System.out.println("parentesisBalanceados");

String formula1 = "(4+(5\*3(4/2)^2))";*//Expresión con parentesis bien*

String formula2 = ")4+(5\*3(4/2)^2))";*//Expresión con un parentesis erroneo*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.parentesisBalanceados(formula1);

**boolean** result2 = instance.parentesisBalanceados(formula2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of esNumero method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testEsNumero() {

System.out.println("esNumero");

**char** c1 = '5';*//Prueba con un numero*

**char** c2 = '-';*//Proeba con un operador*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.esNumero(c1);

**boolean** result2 = instance.esNumero(c2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of esOperador method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testEsOperador() {

System.out.println("esOperador");

**char** c1 = '/';*//Prueba con un operador*

**char** c2 = '9';*//Prueba con un numero*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2= **false**;

**boolean** result1 = instance.esOperador(c1);

**boolean** result2 = instance.esOperador(c2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of esOperadorSinMenos method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testEsOperadorSinMenos() {

System.out.println("esOperadorSinMenos");

**char** c1 = '/';*//Prueba con un operador cualquiera*

**char** c2 = '-';*//Prueba con un menos*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.esOperadorSinMenos(c1);

**boolean** result2 = instance.esOperadorSinMenos(c2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of noDobleOperador method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testNoDobleOperador() {

System.out.println("noDobleOperador");

String formula1 = "4+8/6\*24";*//Prueba con operadores separados*

String formula2 = "166/19\*/7";*//Prueba con operadores juntos*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.noDobleOperador(formula1);

**boolean** result2 = instance.noDobleOperador(formula2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of noDobleDecimal method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testNoDobleDecimal() {

System.out.println("noDobleDecimal");

String formula1 = "47.57+23.86";*//Prueba con decimales bien*

String formula2 = "96.46/3.3.3";*//Prueba con doble decimal*

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.noDobleDecimal(formula1);

**boolean** result2 = instance.noDobleDecimal(formula2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of noOperadorAntesDeParentesis method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testNoOperadorAntesDeParentesis() {

System.out.println("noOperadorAntesDeParentesis");

String formula = "(4+(5\*3(4/2)^2))";

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **false**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.noOperadorAntesDeParentesis(formula);

**boolean** result2 = instance.noOperadorAntesDeParentesis(formula);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

*/\*\**

*\* Test of expresionValida method, of class EvaluacionSintaxis.*

*\*/*

@Test

**public** **void** testExpresionValida() {

System.out.println("expresionValida");

String formula1 = "85\*66-(47\*85)+(48/24)^4+43";

String formula2 = "85\*66..64-(47\*85)++((48m/24)";

EvaluacionSintaxis instance = **new** EvaluacionSintaxis();

**boolean** expResult1 = **true**;

**boolean** expResult2 = **false**;

**boolean** result1 = instance.expresionValida(formula1);

**boolean** result2 = instance.expresionValida(formula2);

assertEquals(expResult1, result1);

assertEquals(expResult2, result2);

}

}