

Tp calculadora científica

Grupo: Tomás Palumbo, Juan Freire, Facundo Santos



10 de junio de 2024

ILS 5TO COMPUTACIÓN

Profesora: María Belén Hipólito

# Informe del Proyecto de Calculadora Científica en Java

Índice

[Informe del Proyecto de Calculadora Científica en Java 1](#_Toc168590759)

[Introducción 1](#_Toc168590760)

[Desarrollo del Trabajo 1](#_Toc168590761)

[Estructura del Código 1](#_Toc168590762)

[Método main 2](#_Toc168590763)

[Inicialización y Bienvenida: 2](#_Toc168590764)

[Recepción de la Entrada: 2](#_Toc168590765)

[Procesamiento de la Operación: 2](#_Toc168590766)

[Método pasarPolacaInversa 3](#_Toc168590767)

[Inicialización: 4](#_Toc168590768)

[Recorrido de la Expresión: 4](#_Toc168590769)

[Vaciar la Pila de Operadores: 4](#_Toc168590770)

[Método evaluarExpresion 5](#_Toc168590771)

[Inicialización: 5](#_Toc168590772)

[Evaluación de la Expresión RPN: 6](#_Toc168590773)

[Resultado Final: 6](#_Toc168590774)

[Métodos Auxiliares 7](#_Toc168590775)

[Método para verificar si es un signo: 7](#_Toc168590776)

[Método para verificar si es un número: 7](#_Toc168590777)

[Método para trabajar con la prioridad de cada operador: 8](#_Toc168590778)

[Conclusión 8](#_Toc168590779)

# Introducción

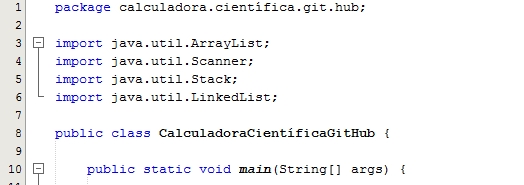
El proyecto desarrollado es una calculadora científica implementada en Java que permite a los usuarios ingresar expresiones matemáticas y obtener el resultado de manera precisa. Esta calculadora soporta operaciones aritméticas básicas como suma, resta, multiplicación, y división, así como operaciones avanzadas como potenciación y radicación. El objetivo de este proyecto es proporcionar una herramienta que convierta las expresiones matemáticas infijas a notación polaca inversa (RPN, por sus siglas en inglés) y luego evalúe la expresión resultante.

# Desarrollo del Trabajo

## Estructura del Código

El código se compone de una clase principal llamada **CalculadoraCientíficaGitHub** que contiene varios métodos encargados de diferentes tareas, como la conversión de expresiones infijas a notación polaca inversa y la evaluación de dichas expresiones. A continuación, se detalla el funcionamiento de cada parte del código:

Agregamos primero todas las librerías necesarias e iniciamos el main.



## Método main

El método principal (**main**) inicia solicitando al usuario que ingrese una operación matemática. El usuario puede ingresar la operación en formato infijo y finalizar la entrada con el signo igual (**=**). También se incluye la opción de apagar la calculadora ingresando la palabra "off".

### Inicialización y Bienvenida:

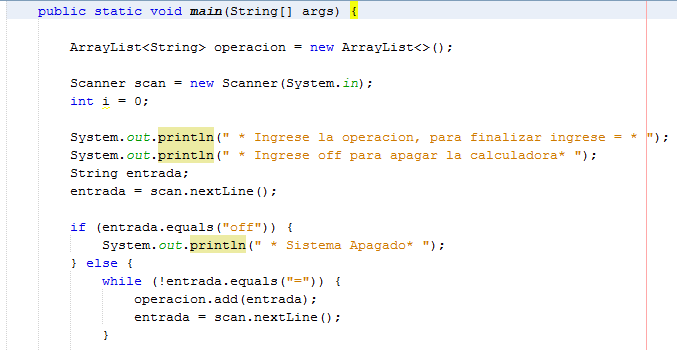
* + Inicializa un **ArrayList** para almacenar la operación.
  + Usa un **Scanner** para leer la entrada del usuario.
  + Muestra mensajes iniciales para guiar al usuario.

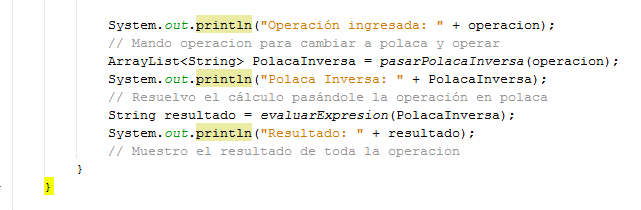
### Recepción de la Entrada:

* + Lee la primera entrada del usuario.
  + Si la entrada es "off", apaga el sistema mostrando un mensaje y finaliza el programa.
  + Si la entrada es diferente de "off", continua leyendo hasta encontrar el signo "=".

### Procesamiento de la Operación:

* + Muestra la operación ingresada.
  + Convierte la operación de notación infija a notación polaca inversa (RPN) usando **pasarPolacaInversa**.
  + Muestra la expresión convertida a RPN.
  + Evalúa la expresión en RPN usando **evaluarExpresion**.
  + Muestra el resultado final de la evaluación.





## Método pasarPolacaInversa

Este método convierte una expresión en notación infija a notación polaca inversa (RPN) utilizando el algoritmo de Shunting Yard. El algoritmo maneja la prioridad de los operadores y los paréntesis para asegurar que la expresión se convierta correctamente.

### Inicialización:

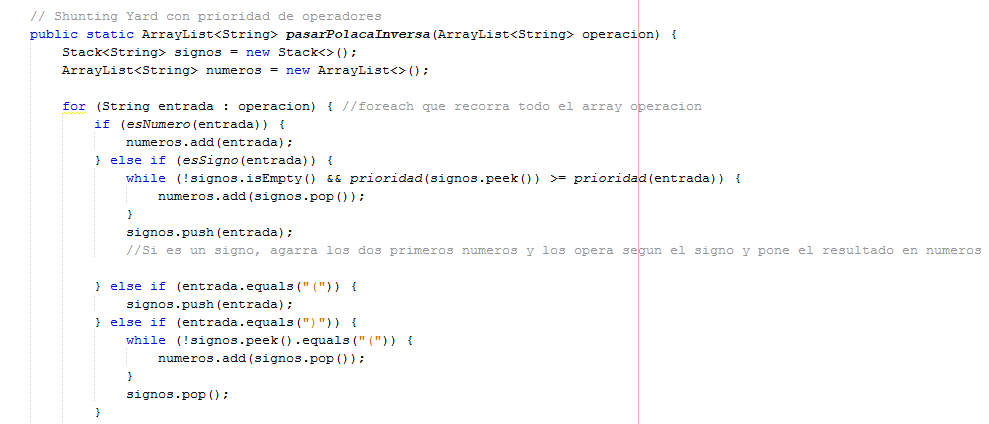
* + Usa una pila (**Stack**) para almacenar los operadores.
  + Usa un **ArrayList** para almacenar los números y la salida final en RPN.

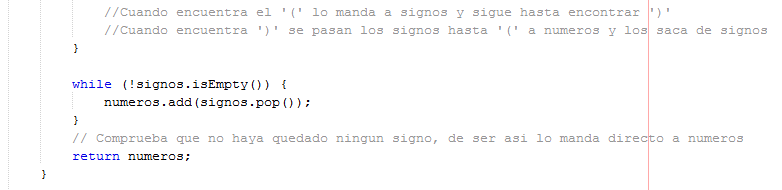
### Recorrido de la Expresión:

* + Recorre cada elemento de la lista de entrada.
  + Si el elemento es un número, lo añade directamente a la lista de salida (**numeros**).
  + Si es un operador, lo maneja según su precedencia, empujando o sacando operadores de la pila (**signos**).
  + Maneja los paréntesis de apertura y cierre para agrupar operaciones correctamente.

### Vaciar la Pila de Operadores:

* + Tras recorrer toda la entrada, vacía la pila de operadores añadiéndolos a la lista de salida.





## Método evaluarExpresion

El método **evaluarExpresion** toma una expresión en notación polaca inversa y la evalúa utilizando una pila (stack) para almacenar los números y aplicar los operadores en el orden correcto.

### Inicialización:

* + Usa una **LinkedList** para almacenar los números y resultados parciales durante la evaluación.

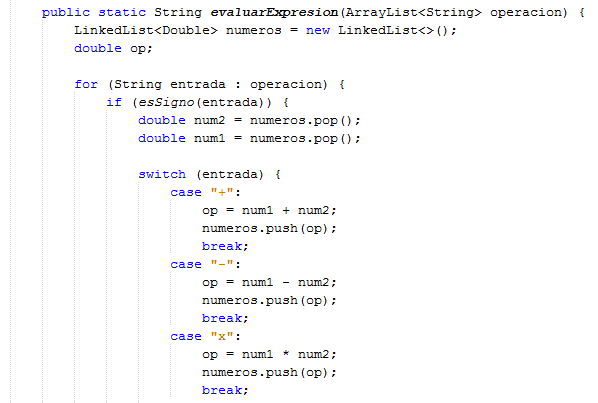
### Evaluación de la Expresión RPN:

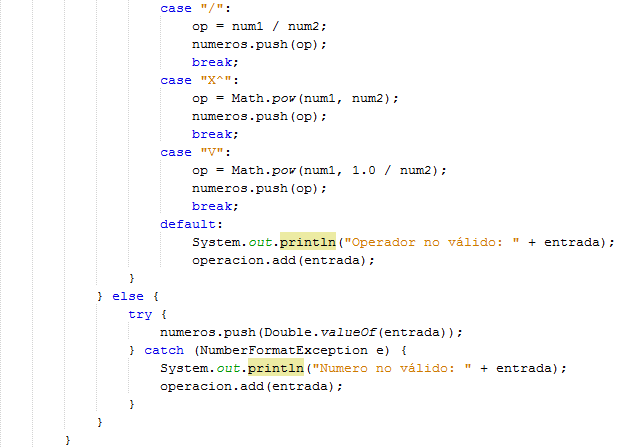
* + Recorre cada elemento de la lista en notación polaca inversa.
  + Si el elemento es un operador, toma los dos números más recientes de la pila, aplica el operador y empuja el resultado de nuevo en la pila.
  + Si el elemento es un número, lo convierte y lo empuja en la pila.

### 

### Resultado Final:

* + Después de procesar todos los elementos, el único valor restante en la pila es el resultado de la expresión.





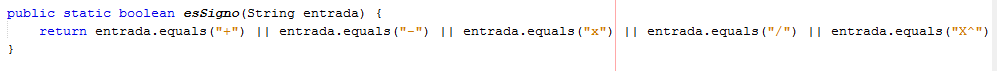


## Métodos Auxiliares

Estos métodos ayudan a verificar si una entrada es un número, un signo, y a determinar la prioridad de los operadores.

### Método para verificar si es un signo:

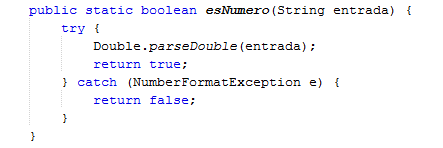
Devuelve lo que sea igual a cada signo preguntando con él .equals.



### 

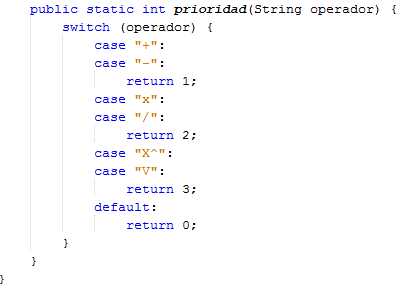
### Método para verificar si es un número:

Utilizamos una nueva función que aprendimos que sirve para detectar errores, se llama Try-Catch. Esto nos sirve para averiguar si lo ingresado es un número, ya que lo intenta convertir a double, y si no se puede pasar, capta el error.



### Método para trabajar con la prioridad de cada operador:

Devuelve un número por cada signo y se usa para comprobar cual signo tiene más prioridad en función de este método.



# 

# Conclusión

La implementación de una calculadora científica en Java que convierte expresiones infijas a notación polaca inversa y luego evalúa estas expresiones demuestra el uso efectivo de estructuras de datos como pilas y listas enlazadas, así como algoritmos de manipulación de expresiones matemáticas. El proyecto no solo facilita el cálculo de expresiones complejas, sino que también sirve como un ejemplo educativo de cómo aplicar algoritmos clásicos en la programación. Este proyecto puede ser extendido para soportar más funciones matemáticas y mejorar la interfaz de usuario para una mejor experiencia.

Este informe proporciona una visión completa del desarrollo del proyecto, desde la introducción del problema hasta la implementación y la conclusión, explicando cada parte del código con detalle y ofreciendo espacios visuales para una mejor comprensión del funcionamiento del programa.