***TP PROGRAMACIÓN***



***NOMBRE DE LA MATERIA:*** Laboratorio de Algoritmos orientado a Objetos

***TEMA:*** Pila – Arraylist – Colas – Listas.

***APELLIDO Y NOMBRE DEL ALUMNO:*** Lautaro Sorribas, Sol Oliveti Fernández, Marco Montarner.

***CUATRIMESTRE:*** 1er Cuatrimestre

***COLEGIO – CURSO:*** Instituto La Salette – 5to Computación

***PROFESOR:*** María Belén Hipólito

***FECHA DE ENTREGA DEL TRABAJO:*** 10/6/2024

***Introducción:***

En este informe se presenta el desarrollo de una calculadora en Java que permite realizar cálculos combinados ingresados por el usuario. La calculadora soporta operaciones básicas como suma, resta, multiplicación, división, potencia y raíz cuadrada. Además, la calculadora convierte las expresiones ingresadas en notación infija a notación polaca inversa (RPN) y evalúa la expresión resultante. El objetivo principal de este trabajo es proporcionar una herramienta que permita manejar expresiones matemáticas complejas de manera eficiente, utilizando arraylist, pilas, colas o listas.

***Desarrollo del trabajo:***

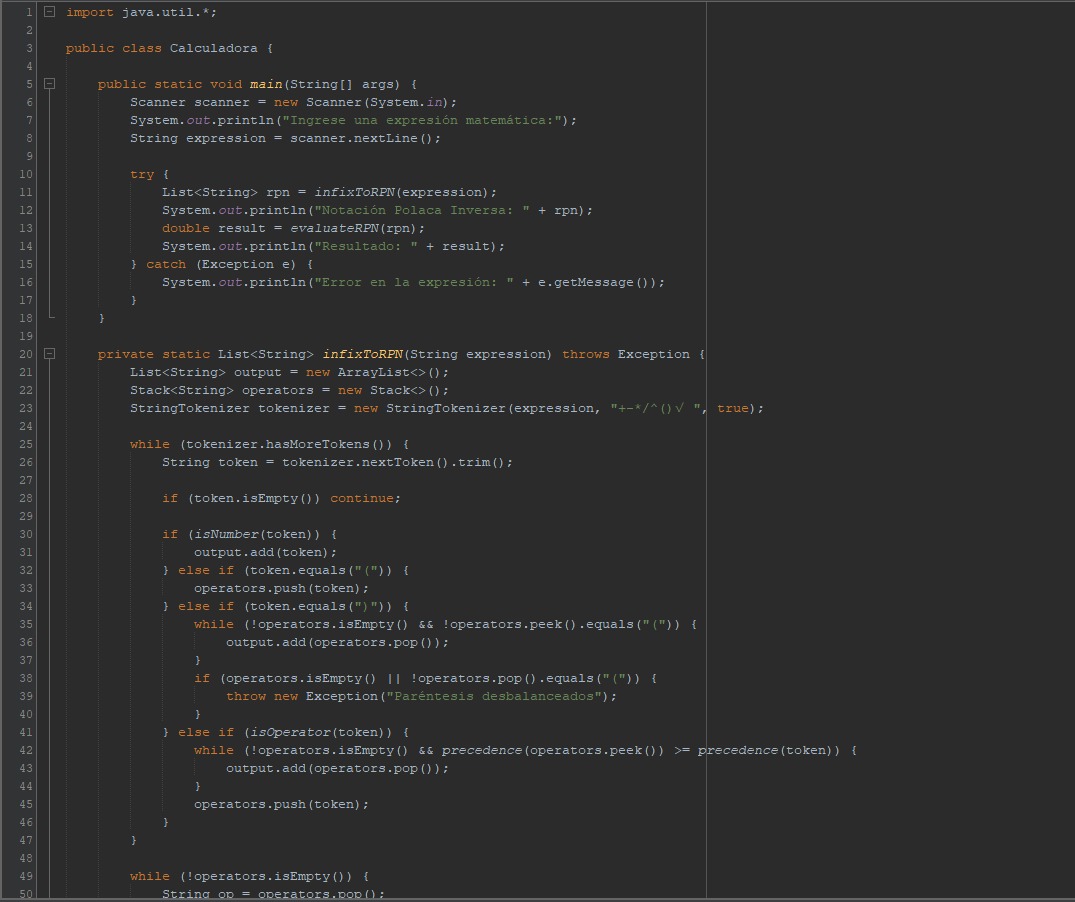
El desarrollo de la calculadora se realizó en varios pasos, detallados a continuación:

***1. Importación de paquetes:***

import java.util.\*;

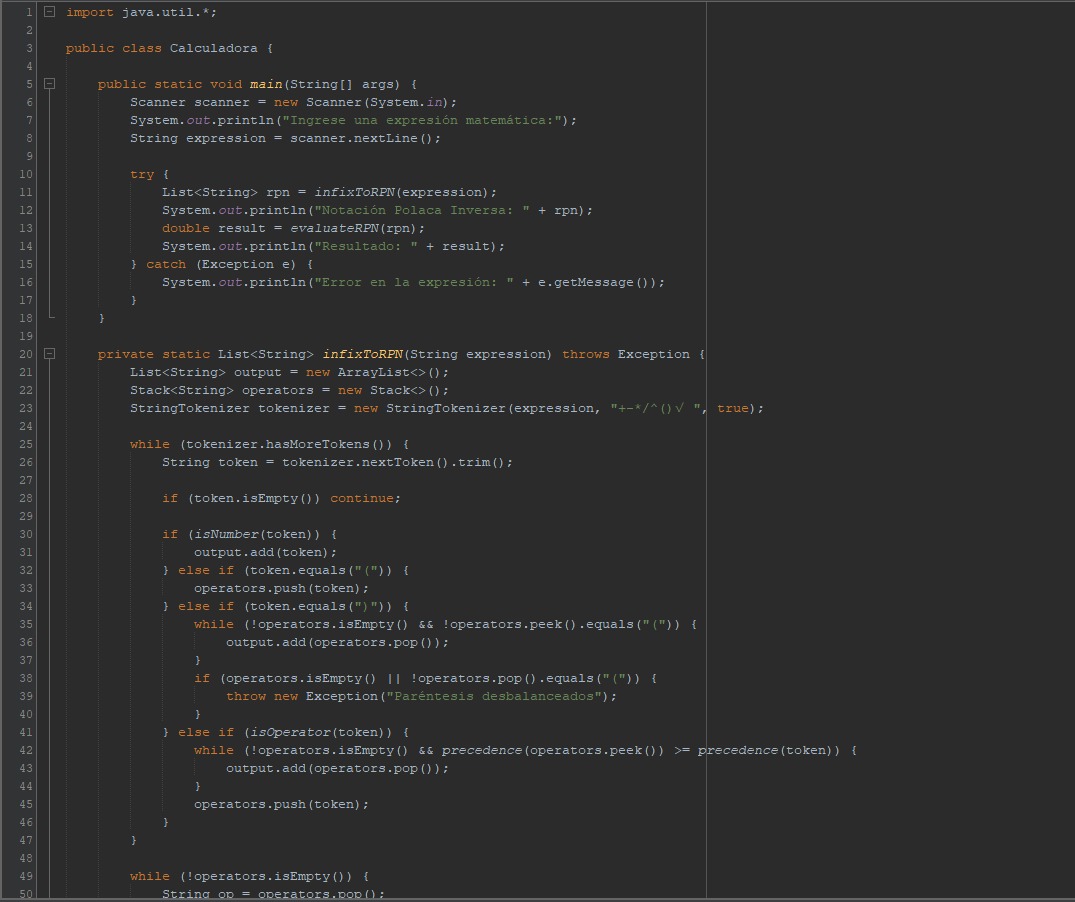
Se importa la clase Scanner del paquete java.util, que se utiliza para leer la entrada del usuario. La clase List y ArrayList del paquete java.util, que se utilizan para almacenar y manipular listas de elementos. La clase Stack del paquete java.util, que se utiliza para implementar una pila. La clase StringTokenizer del paquete java.util, que se utiliza para dividir una cadena en tokens.

***2. Definición de la clase Calculadora:***



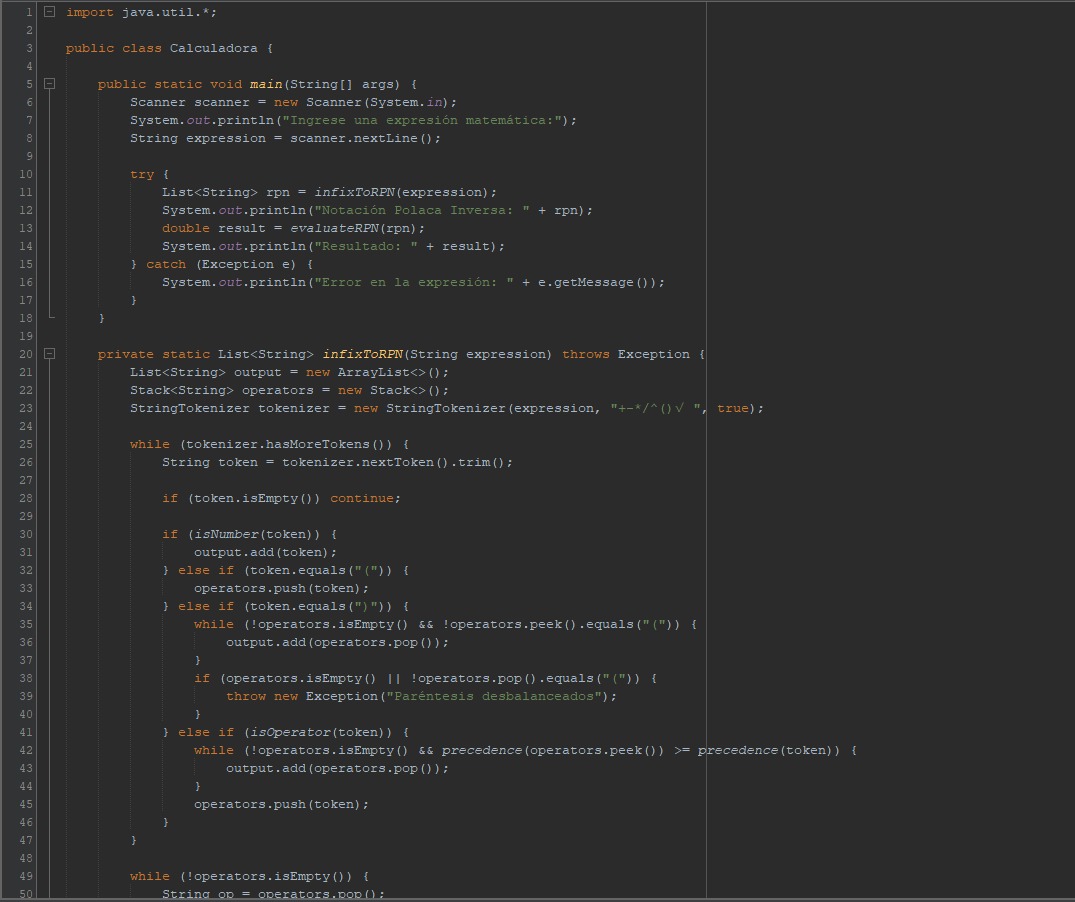
Se define una clase llamada Calculadora.

***3. Método main:***



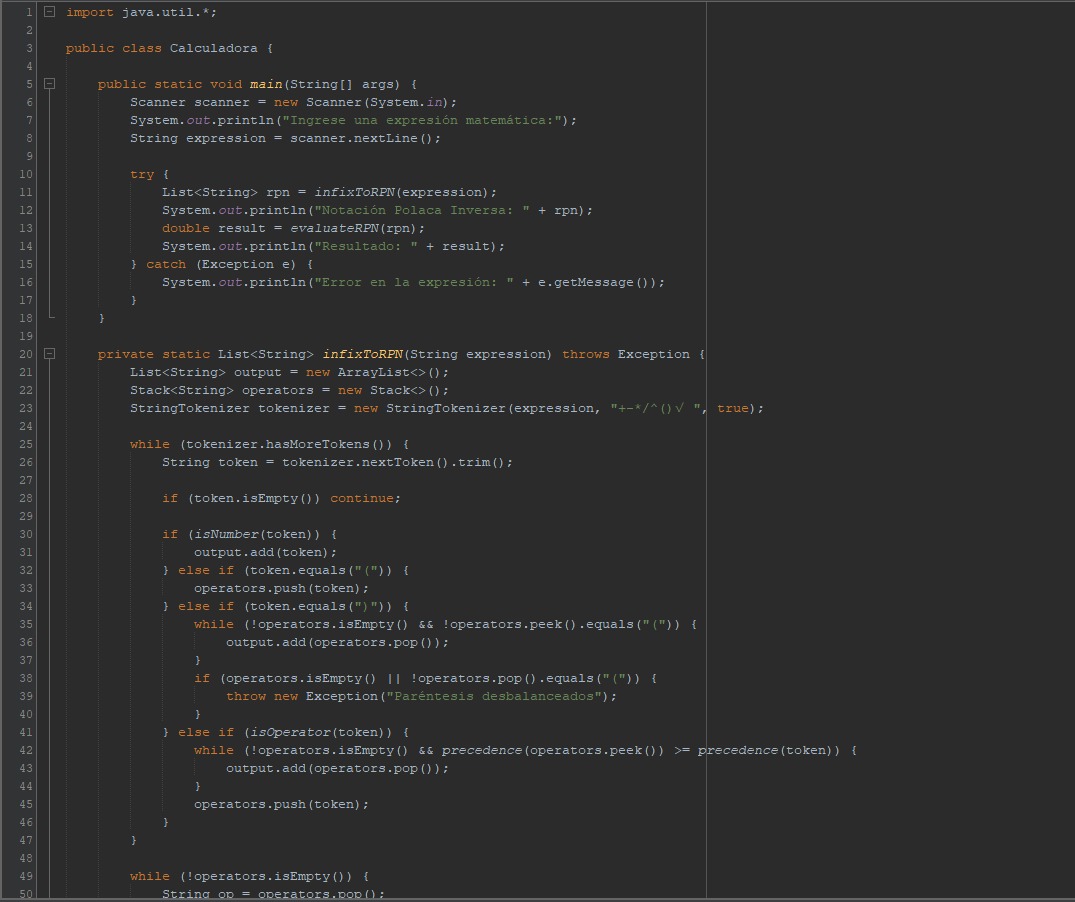
Se define el método principal main, que es el punto de entrada del programa. Se crea una instancia de Scanner para leer la entrada del usuario desde la consola.

Lectura de la Expresión:



Se utiliza la clase Scanner de Java para leer la expresión matemática ingresada por el usuario y almacenarla en la variable expression. Este paso es fundamental ya que la expresión en notación infija será la base para la conversión y evaluación posterior.

***4. Manejo de Errores:***



Se envuelve el código en un bloque try-catch para manejar posibles excepciones que puedan ocurrir durante la ejecución del programa. Si se produce una excepción, se muestra un mensaje de error que indica el problema.

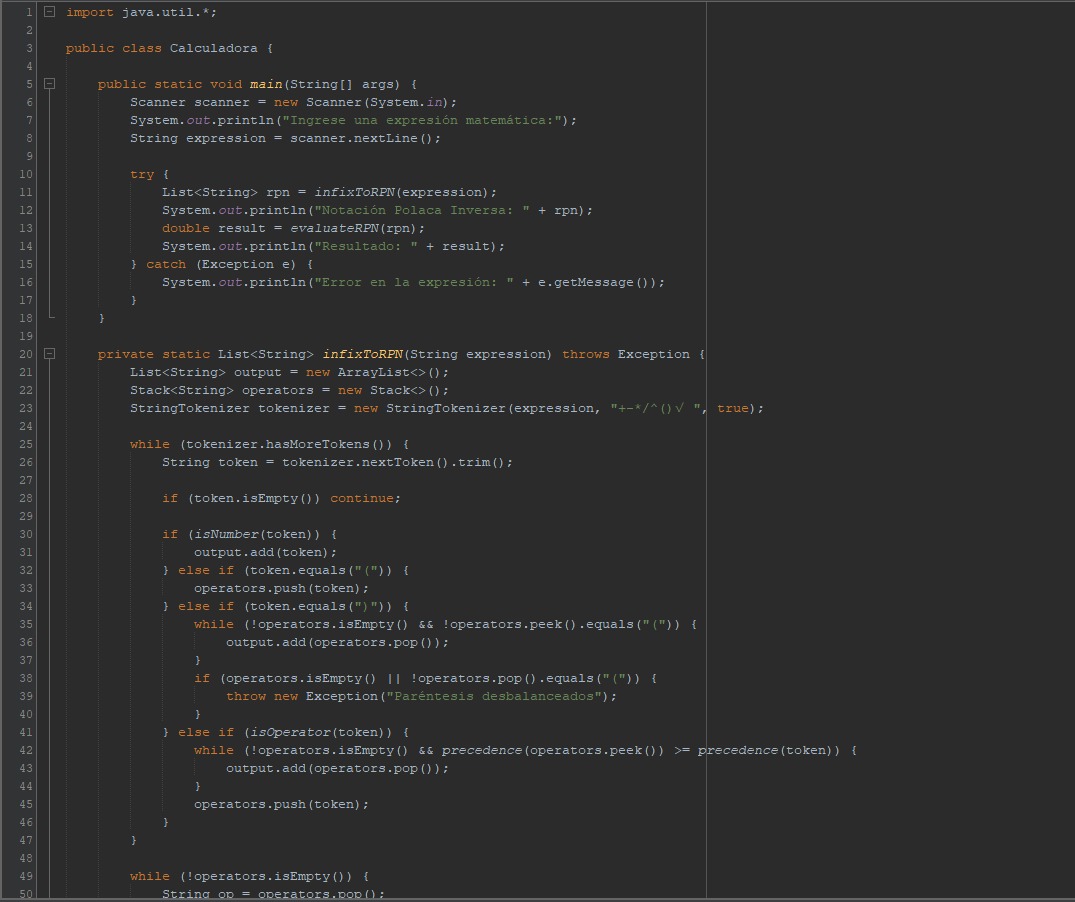
***5. Conversión de Notación Infija a Notación Polaca Inversa (RPN):***

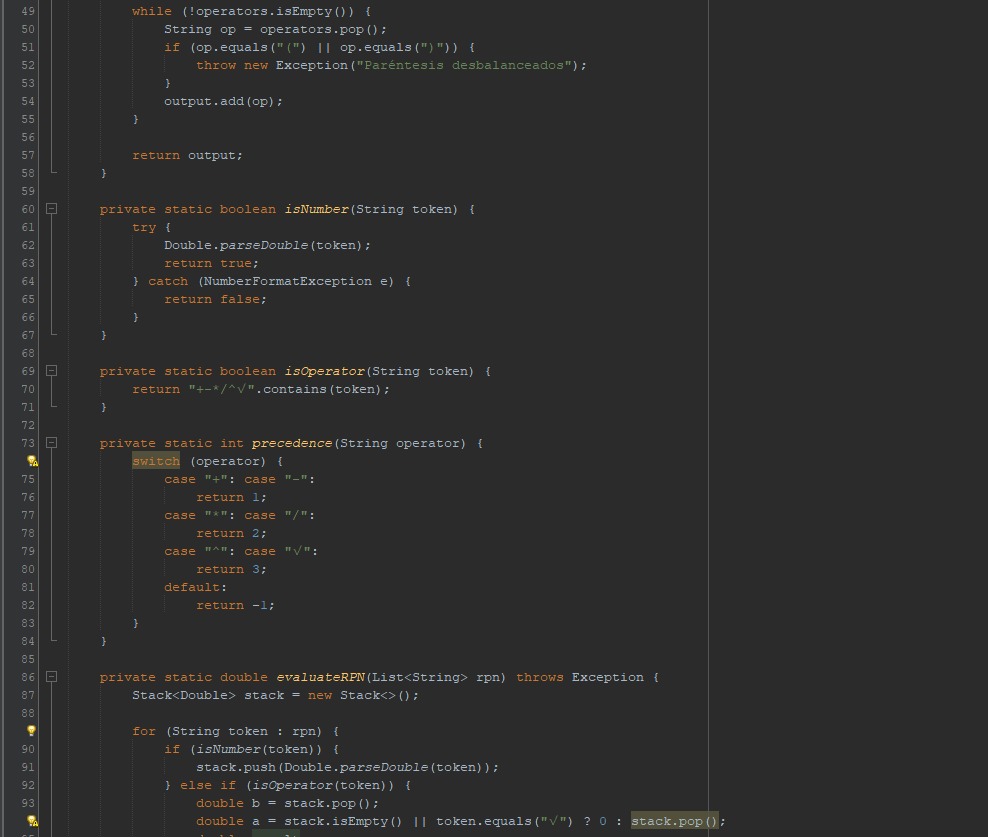
Para convertir la expresión de notación infija a RPN, se implementó el algoritmo de Shunting Yard, desarrollado por Edsger Dijkstra. Este algoritmo utiliza una pila para manejar los operadores y paréntesis, y una lista para la salida en RPN. Divide la expresión en tokens y reorganiza los tokens en RPN. El proceso incluye:

Tokenización: Se separa la expresión en tokens que pueden ser números, operadores o paréntesis. Un token es una unidad básica de significado en una cadena de texto. En el contexto de nuestra calculadora, los tokens son números, operadores y paréntesis que juntos forman expresiones matemáticas que pueden ser procesadas y evaluadas mediante algoritmos.

Manejo de Operadores: Se utilizan reglas de precedencia para ordenar los operadores de manera adecuada.

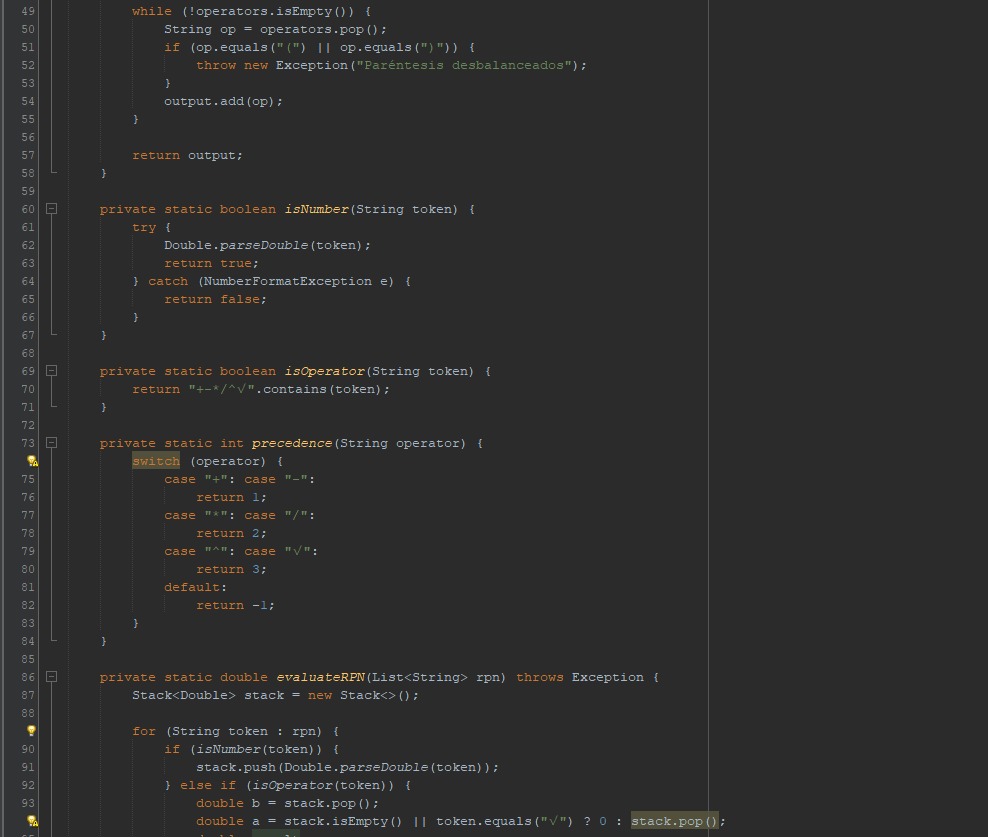
Manejo de Paréntesis: Se asegura que los paréntesis se procesen correctamente para mantener la estructura de la expresión.





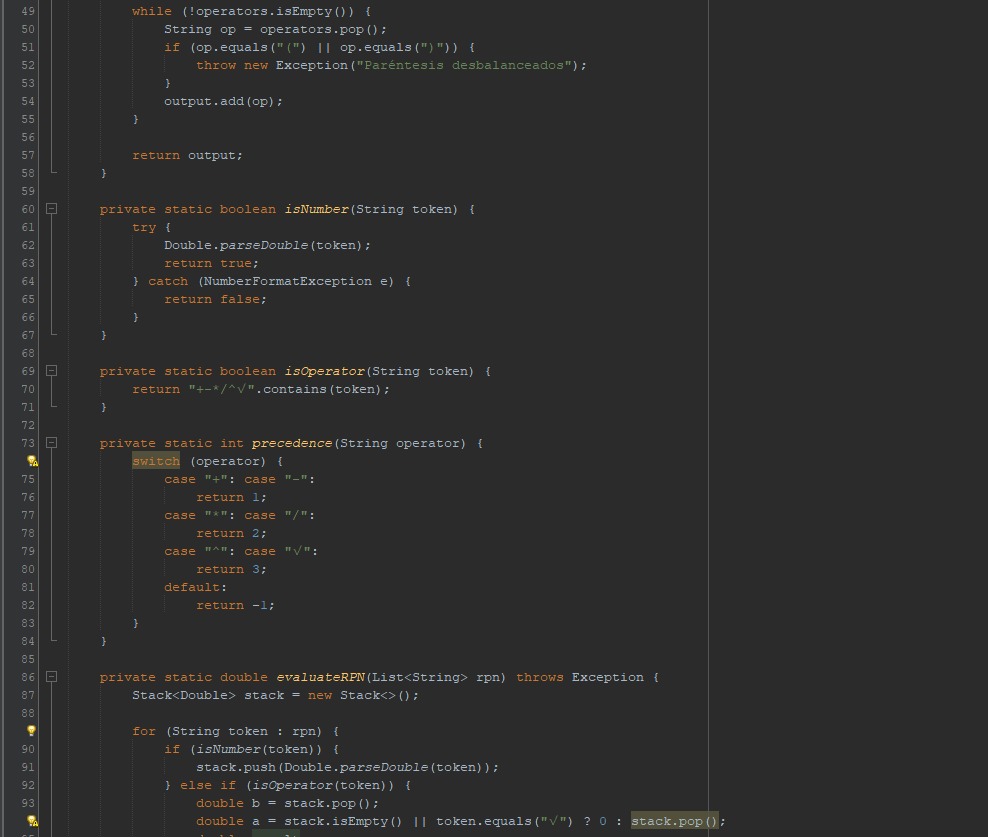
***6. Método isNumber:***

Este método verifica si un token es un número. Utiliza Double.parseDouble para intentar convertir el token a un número. Si la conversión es exitosa, devuelve true; de lo contrario, devuelve false.



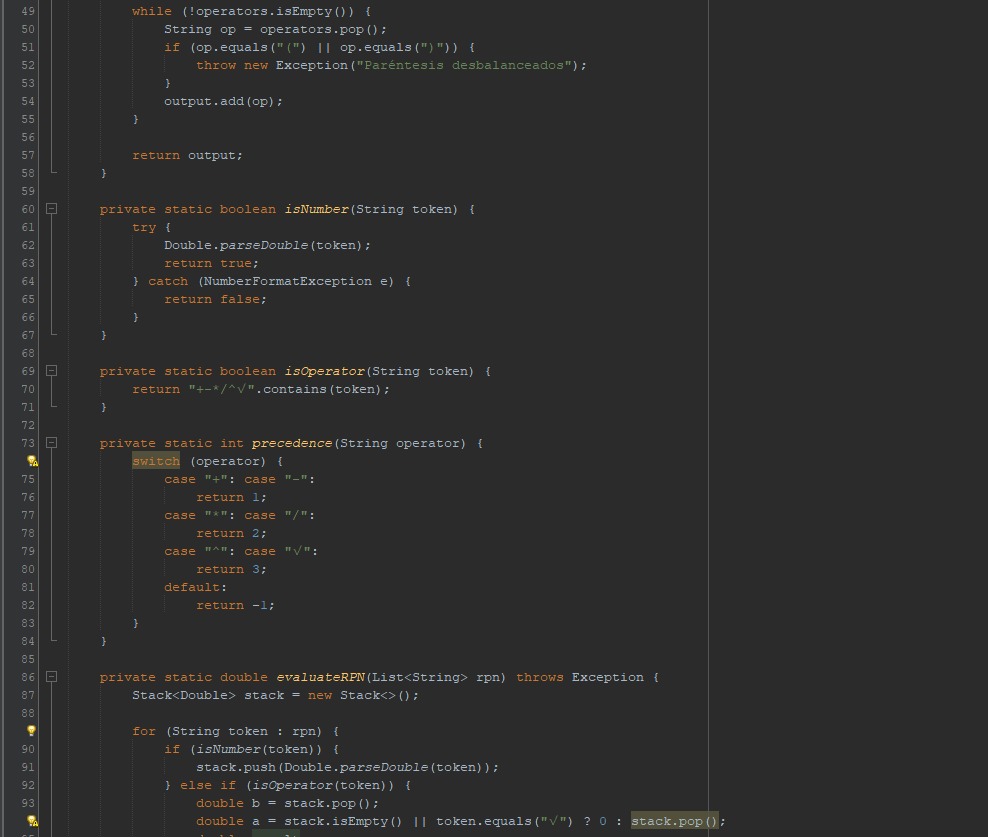
***7. Método isOperator:***

Este método verifica si un token es un operador matemático.Comprueba si el token está contenido en la cadena "+-\*/^√".



***8. Método precedence:***

Este método devuelve la precedencia de un operador. Asigna un valor numérico a cada operador para determinar su precedencia a través de un Switch.



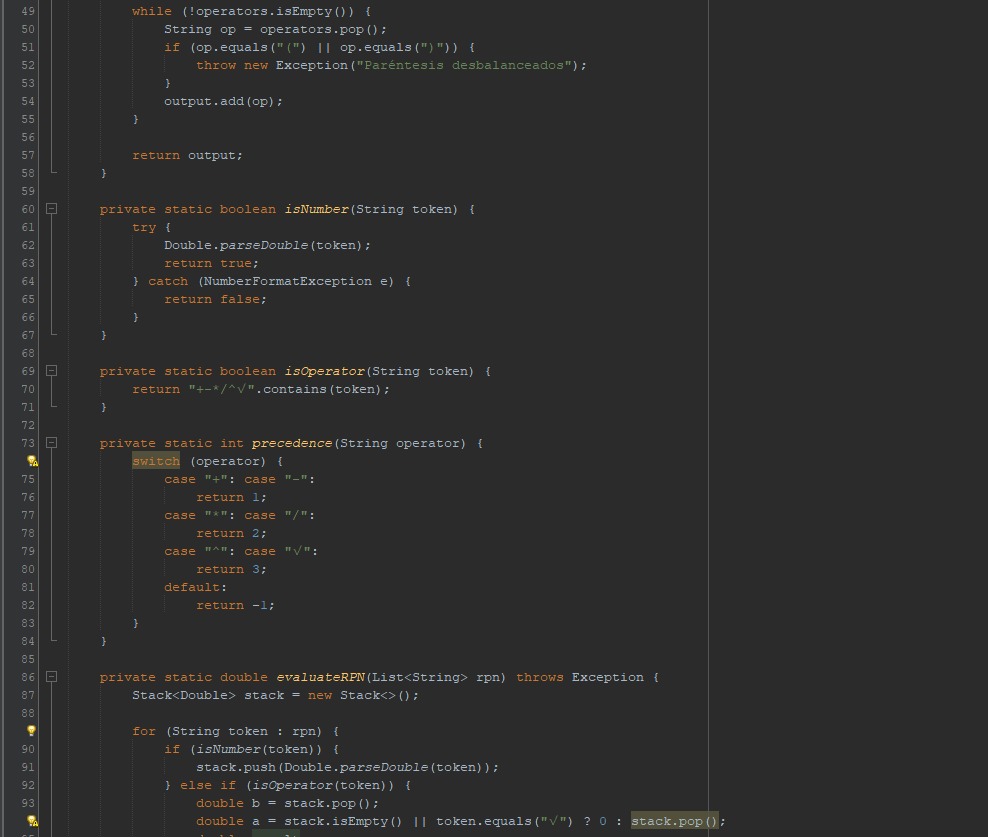
***9. Método evaluateRPN:***

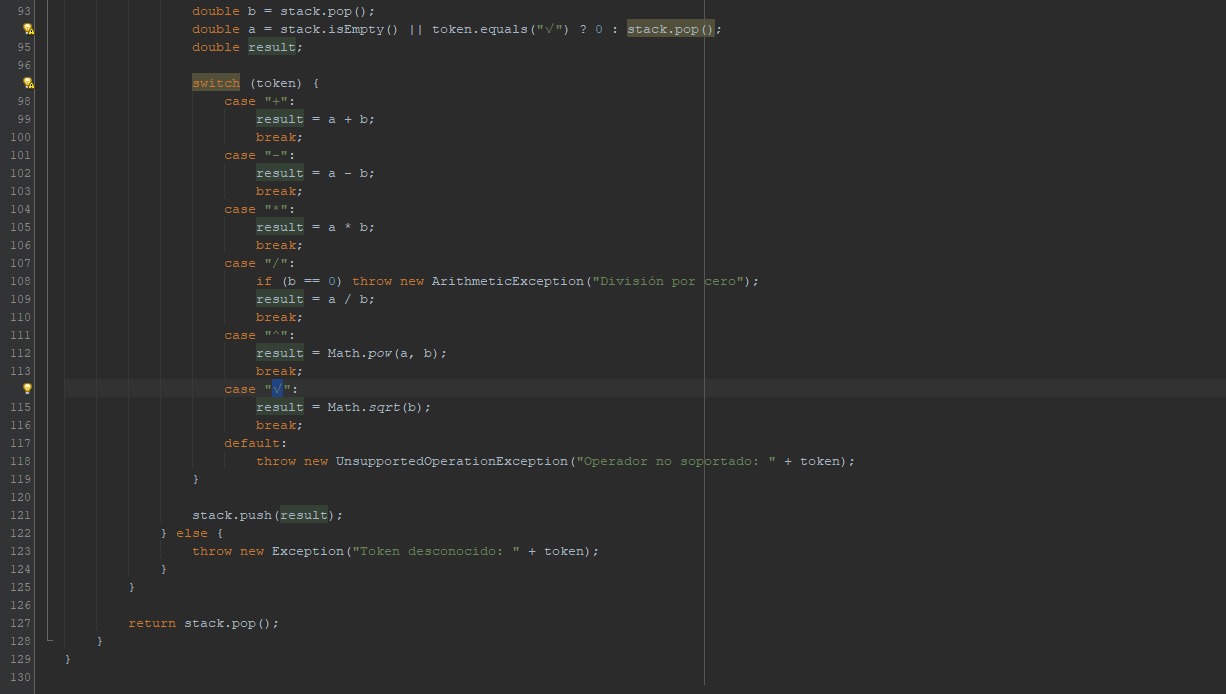
Este método evalúa la expresión matemática en notación polaca inversa. Utiliza una pila para realizar las operaciones. Itera sobre los tokens de la expresión y realiza las operaciones correspondientes.

token de la expresión en RPN se procesa de la siguiente manera:

Números: Se añaden a la pila.

Operadores: Se aplican los operadores a los operandos correspondientes que se extraen de la pila, y el resultado se vuelve a insertar en la pila.





***10. Manejo de Excepciones:***

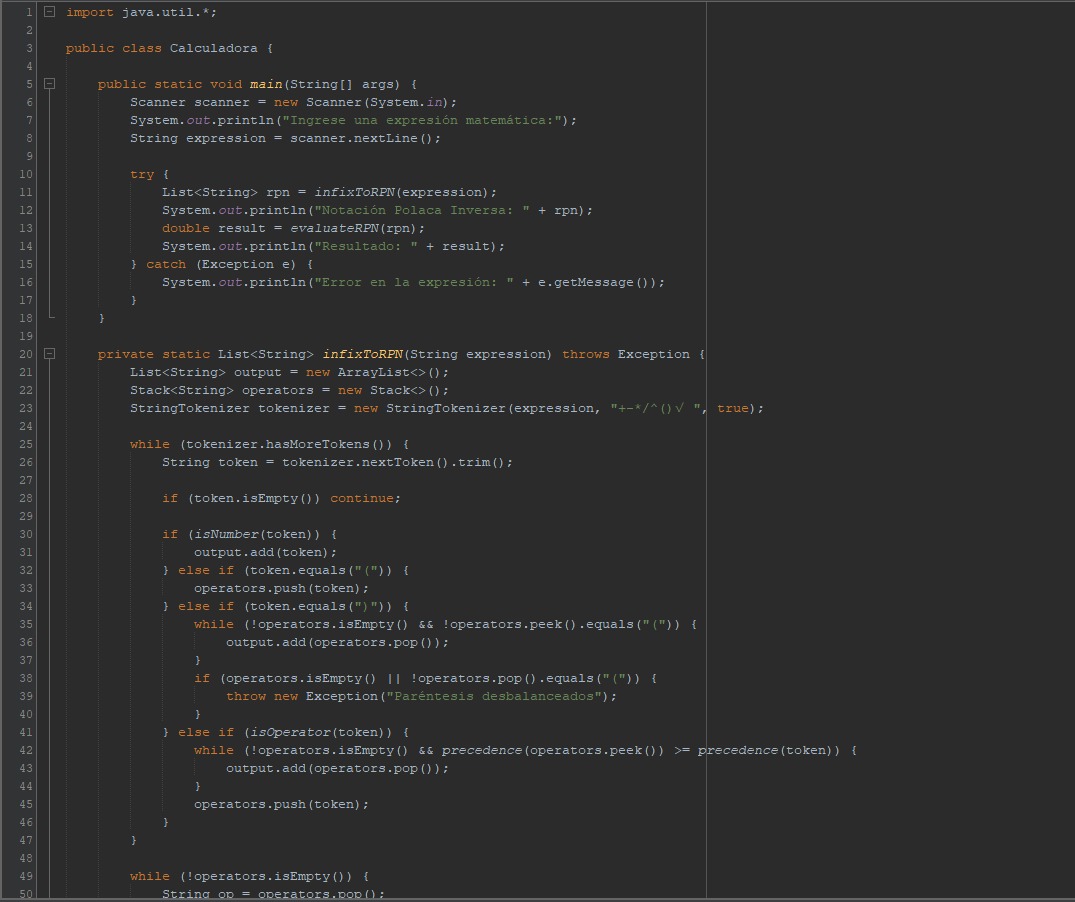
El código maneja varias excepciones, como divisiones por cero, desequilibrios de paréntesis y tokens desconocidos.

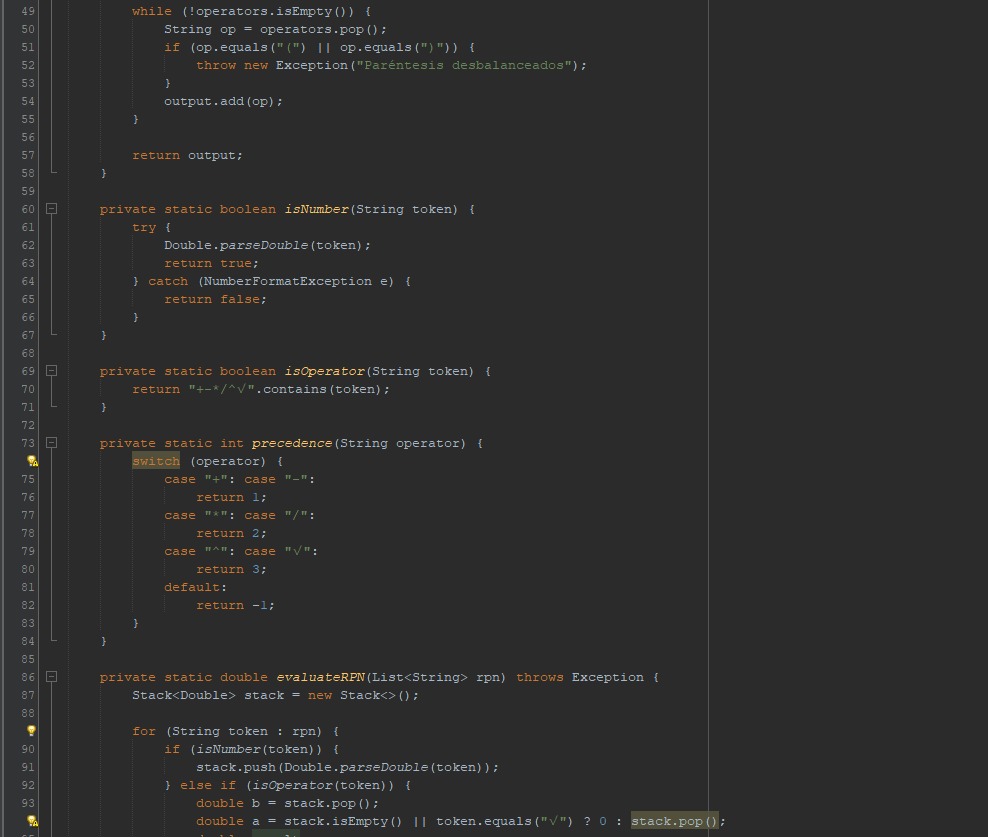
***Resumen:***

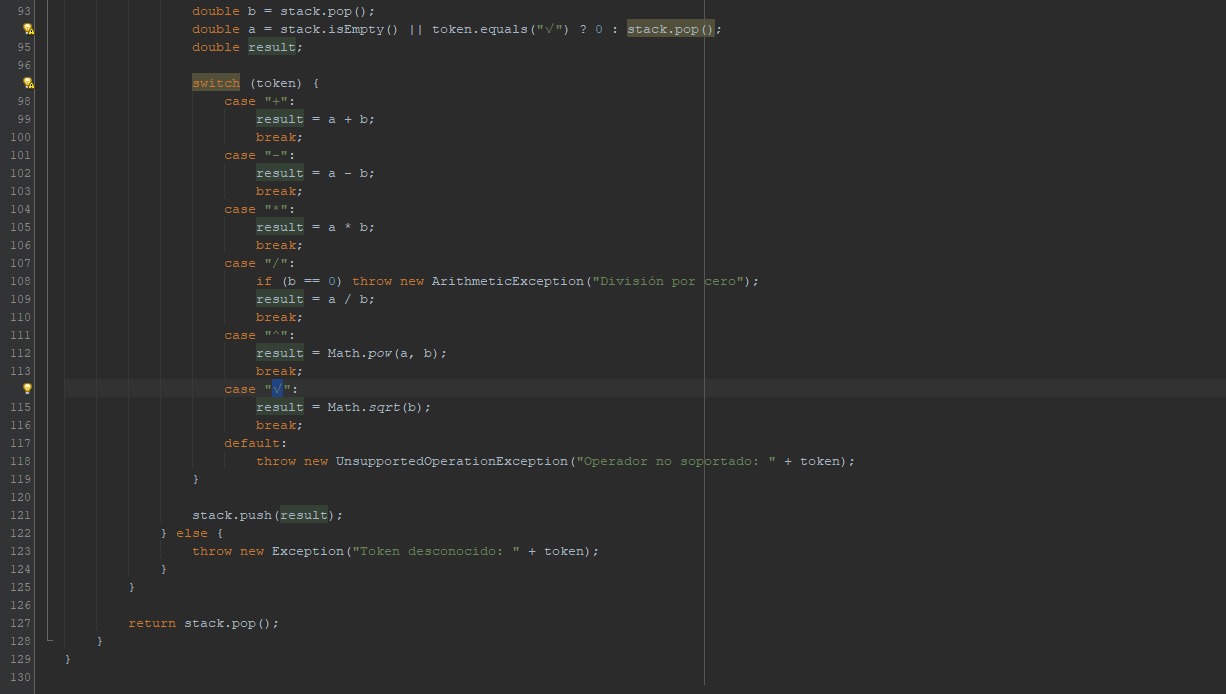
El programa comienza solicitando al usuario que ingrese una expresión matemática. Luego, utiliza el método infixToRPN para convertir la expresión en notación infija a notación polaca inversa, usa una lista para la salida y una pila para los operadores. Este método utiliza un algoritmo de evaluación de expresiones matemáticas para realizar la conversión. Si se encuentra algún error durante este proceso, se lanza una excepción y se muestra un mensaje de error. Una vez que se ha obtenido la notación polaca inversa, se utiliza el método evaluateRPN para evaluar la expresión y obtener el resultado. Este método utiliza una pila para realizar los cálculos necesarios. Se recorre cada token de la notación polaca inversa y se realiza la operación correspondiente. Si se encuentra algún error durante este proceso, se lanza una excepción y se muestra un mensaje de error.

Finalmente, se muestra en la consola la notación polaca inversa obtenida y el resultado de la expresión. El código utiliza varias funciones auxiliares para verificar si un token es un número (isNumber), un operador (isOperator), obtener la precedencia de un operador (precedence), y realizar las operaciones aritméticas necesarias.

***Archivamos fotos del proyecto:***







***Conclusión:***

El desarrollo de esta calculadora en Java nos permitió aprender y aplicar conceptos fundamentales de estructuras de datos y algoritmos, como pilas y el algoritmo de Shunting Yard. La capacidad de convertir expresiones de notación infija a notación polaca inversa y evaluar estas expresiones proporciona una herramienta robusta para manejar cálculos matemáticos complejos. La implementación demuestra cómo la teoría de la computación puede aplicarse para resolver problemas prácticos de manera eficiente y efectiva. Esta calculadora es un ejemplo claro de cómo la programación puede facilitar el procesamiento y la resolución de expresiones matemáticas complejas.