# Proyecto 2 Algoritmos y Estructuras de Datos 1

1st Joseph Coronado Alvarado
*Ingenieria en Computadores*
*Instituto Tecnológico De Costa Rica*
Cartago, Costa Rica
josephcoronado11@estudiantec.cr

2nd Josthin Soto Sanchez
*Ingenieria en Computadores*
*Instituto Tecnológico De Costa Rica*
Catargo, Costa Rica
josthin.soto.s@estudiantec.cr

***Abstract*—Esta es la documentacion externa del segundo proyecto programado de el curso algortimos y estructuras de datos 1, el cual busca documentar el deserrollo del proyecto y ejemplificar como se desarrolló el mismo.**

## I. Descripción del Problema

El proyecto se basa en la construcción de una calculadora utilizando arboles de expresion binarios dentro de un servidor la cual acepte opereraciones de una longitud n, tales como (+, -, *, /, **) y que también sea capaz de realizar operaciones lógicas como (and, or, not, xor) igualmente de una longitud n. Todo lo anterior mediante una aplicación programada en Java la cual posea la cualidad de aceptar expresiones mediante el reconocimiento de una expresión matematica impresa.

## II. Diagramas de Clase

A continuación se mostrarán los diagramas de clase, para las clases que se utilizaron en el proyecto

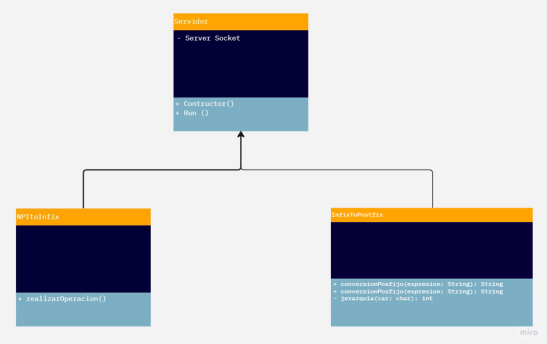


Fig. 1. Imagen del Diagrama de Clase del Servidor


Fig. 2. Imagen del Diagrama de Clase de la Calculadora

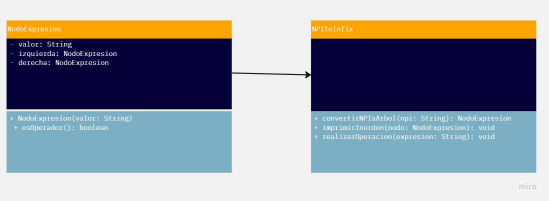


Fig. 3. Imagen del Diagrama de Clase del Nodo de Expresion

## III. Descripción de las Estructuras de Datos Desarrolladas

### A. Servidor

La clase Servidor es la clase principal de la aplicación. Se encarga de gestionar la comunicación con los clientes a través de sockets y de procesar los mensajes que recibe. Tiene un atributo serverSocket que representa el socket del servidor. El método constructor() inicia el servidor en el puerto 9999 y crea un hilo para gestionar las conexiones entrantes. El método run() se ejecuta en un hilo separado y maneja las solicitudes de los clientes. También contiene un método realizarOperacionMatematica() que convierte y evalúa expresiones matemáticas. La clase InfixToPostfix se utiliza para convertir expresiones matemáticas de notación infija a notación postfija (polaca inversa). La clase NPItoInfix esta relacionada con la conversión de notación polaca inversa (NPI) a notación infija.

### B. Calculadora

La clase Calculadora es la clase principal de la aplicación y extiende Application, que es una clase base de JavaFX para aplicaciones gráficas. En el método start(), se configura la interfaz de la aplicación, crea la ventana principal y agrega elementos gráficos, como imágenes, cajas de texto y botones. En el método parametrosCalculadora(), se definen los elementos gráficos, como la imagen de fondo, la caja de texto y el botón para realizar cálculos. Los métodos convertirExpresionAPolaco() y enviarMensajeServidor() se utilizan para procesar y enviar expresiones matemáticas al servidor. La aplicación utiliza contenedores Pane para organizar elementos gráficos. Se crea una escena de JavaFX para mostrar la interfaz de la calculadora. La interfaz gráfica incluye una imagen de fondo, una caja de texto (TextField) y un botón para calcular expresiones matemáticas. La clase InfixToPostfix parece ser una

parte de la lógica de la aplicación que se utiliza para convertir expresiones matemáticas de notación infija a notación postfija (polaca inversa). Los métodos enviarMensajeServidor() crean una conexión de socket con un servidor en localhost en el puerto 9999 y envían mensajes al servidor. El servidor es el responsable de realizar cálculos matemáticos basados en las solicitudes del cliente.

### *C. Nodo de Expresión*

La clase NodoExpresion representa un nodo en el árbol de expresión. Tiene tres atributos: valor para almacenar el valor del nodo, izquierda para el hijo izquierdo y derecha para el hijo derecho. El método esOperador() verifica si el valor del nodo es un operador matemático (como +, -, *, /). La clase NPItoInfix se utiliza para convertir una expresión en Notación Polaca Inversa (NPI) en un árbol de expresión y para imprimir la expresión en notación estándar (infix). El método convertirNPIaArbol(String npi) toma una expresión en NPI como entrada y devuelve un árbol de expresión representado como un nodo raíz. El método imprimirInorden(NodoExpresion nodo) imprime la expresión en notación estándar (infix) utilizando un recorrido inorden del árbol de expresión. El método realizarOperacion(String expresion) se encarga de realizar la conversión y la impresión del árbol. Los métodos convertirNPIaArbol() y imprimirInorden() son esenciales para la conversión y la impresión de la expresión matemática. Los métodos toman en cuenta la notación polaca inversa y manejan la construcción y visualización de la expresión en notación estándar.

## IV. Problemas encontrados en forma de bug de GitHub

Por algún motivo, el árbol de expresión si bien hace cálculos correctos, tiene problemas para procesar números de más de 2 digitos como el 33. Por ejemplo, la expresión 5/33+2 lo interpreta como ((3/3) + 2), el 5 desaparece completamente, por ende desordenado el orden de las operaciones y el resultado de las mismas.

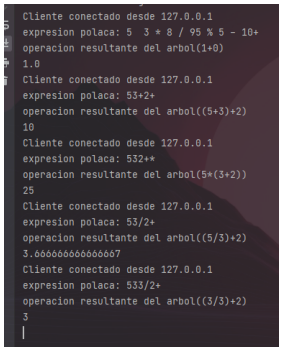


Fig. 4. Imagen del Bug que se presentó con el Arbol de expresión