



2 de Junio del 2018

PROYECTO U1

LENGUAJES Y AUTOMATAS II | Juan Pablo Rosas Baldazo

Iván Alejandro Padilla Esparza | #15480063
INGENIERIA EN SISTEMAS COMPUTACIONALES

Introducción

Este proyecto tiene como objetivo realizar un programa que nos ayude a la evaluación de expresiones que incluyan operandos números y operadores matemáticos, como la suma, resta, multiplicación, división, etc. Este también tiene como objetivo que al momento de ingresar dicha operación nos pueda dar como resultado los 3 ordenes, las cuales son: Prefija, Posfija e Infija, todo esto con la finalidad de poder ver como es el orden correcto de cada expresión.

Descripción:

Se realizo un árbol binario en el cual consta que cada nodo tiene 2 hijos, ya sean números, letras y operadores matemáticos.

El objetivo de este programa es poder imprimir las 3 expresiones (Posfija, Prefija e Infija) mediante una operación capturada por el mismo usuario.

Pseudocodigo

```
package Arbol;

import java.util.Scanner;

public class Nodo {

    public static void main(String[] args) {

        print ("Digite expresión que desea evaluar");

        String infija, infija1 = Expresión ingresada;

        print ("Resultado y ordenes de la expresión");

    }

    public static double evaluar (String infija, String infija1){

        String prefija= conversión expresión a prefija;

        String posfija = conversión expresión a posfija;

        print ("La expresión posfija es: " + posfija);

        print ("La expresión infija es: " + infija);

        print ("La expresión prefija es: " + prefija);

        return Resultado de la expresión;

    }

    private static String convertirpre(String infija) {

        pila = caracteres de la expresión;
```

```
for (recorrido de la pila){  
    char letra = separar carácter por carácter;  
    if (si el carácter no es operador) {  
        if (si la pila está vacía) {  
            (entonces) se apila carácter en la pila;  
        } en caso contrario {  
            int pe = prioridad en la expresión del operando;  
            int pp = prioridad en pila del siguiente operando;  
            if (pe > pp){  
                apilar operando en pila;  
            } en caso contrario {  
                prefija += se desapila el operando de la pila;  
                se apila el operando en la pila e ingresa a prefija;  
            }  
        }  
    } en caso contrario {  
        prefija += el operador se ingresa a prefija;  
    }  
}  
mientras (la pila no esté vacía) {  
    prefija += se desapilan los operandos y se ingresan a prefija;  
}  
devuelve el orden de prefija;  
}
```

```

private convertirpos (String infija) {
    pila = número de caracteres de la expresión;
    for (recorrido de la pila) {
        char letra = separar carácter por carácter;
        if (si el carácter es operador) {
            if (si la pila está vacía) {
                (entonces) se apila carácter en la pila;
            } en caso contrario {
                int pe = prioridad en la expresión del operador;
                int pp = prioridad en pila del operador;
                if (pe > pp){
                    apilar operador en pila;
                } en caso contrario {
                    posfija = se desopila el operador de la pila;
                    se apila el operador en la pila e ingresa a posfija;
                }
            }
        } en caso contrario {
            posfija = el operando se ingresa a posfija;
        }
    }
    mientras (la pila no esté vacía) {
        posfija = se desapilan los operadores y se ingresan a posfija;
    }

    return devuelve el orden de posfija;
}

private static int prioridadEnExpresion (char operador) {

```

```

    if (operador == '^') devuelve ID# 4;
    if (operador == '*' || operador == '/') devuelve ID# 2;
    if (operador == '+' || operador == '-') devuelve ID# 1;
    if (operador == '(') devuelve ID# 5;
    return 0;
}

private static int prioridadEnPila (char operador) {
    if (operador == '^') devuelve ID# 3;
    if (operador == '*' || operador == '/') devuelve ID# 2;
    if (operador == '+' || operador == '-') devuelve ID# 1;
    if (operador == '(') return 0;
    return 0;
}

private static double evaluar (String posfija) {
    tamaño pila = cantidad de caracteres de la expresión;
    for (recorrido de la pila) {
        char letra = separar carácter por carácter;
        if (si el carácter no es operador) {
            double num = se apila operando en la pila;
            se apila numero en la pila;
        } en caso contrario {
            double num2 = se desapila el operando;

            double num1 = se desapila el operando;
            double num3 = resultado de la operación;
            apila resultado de la operación en la pila;
        }
    }
}

return devuelve resultado de la expresión;

```

```

    }

    private static boolean esOperador (char letra) {

        Si es alguno de los operadores (letra == '*' || letra == '/' || letra == '+' ||
            letra == '-' || letra == '(' || letra == ')' || letra == '^') {

            responde verdadero;
        }

        En caso contrario responde falso;
    }

    private static double operacion (char letra, double num1, double num2) {

        si el operando (letra == '*') regresa una multiplicación;
        si el operando (letra == '/') regresa una división;
        si el operando (letra == '+') regresa una suma;
        si el operando (letra == '-') regresa una resta;
        si el operando (letra == '^') regresa una potencia;

        return 0;

    }
}

```

Arbol.java

```

package Arbol;

public class Arbol {

    se crea variable int n, tope;
    se crea objeto pila [];

    public Arbol (int n) {

        esta n es = tamaño de la pila;

        tope = 0;

        pila = tiene un tamaño = n;

    }

    public boolean estaVacia(){

```

```
        si esta vacía regresa un 0;
    }
    public boolean estaLlena(){
        si está llena regresa el número de elementos en la pila;
    }
    public boolean apilar(recibe un dato){
        if(la pila esta llena){
            regresa un falso;
        }
        en caso contrario
            posicion de la pila actual = dato;
            posicion de la pila avanza 1;
            regresa un verdadero;
        }
        public Object desapilar(){
            if(la pila esta vacía) {
                regresa un valor nulo;
            }
            posición de la pila retrocede 1;
            regresa posición de la pila;
        }
        public Object elementoTope(){
            regresa posición anterior de la pila;
        }
    }
```

Resultado

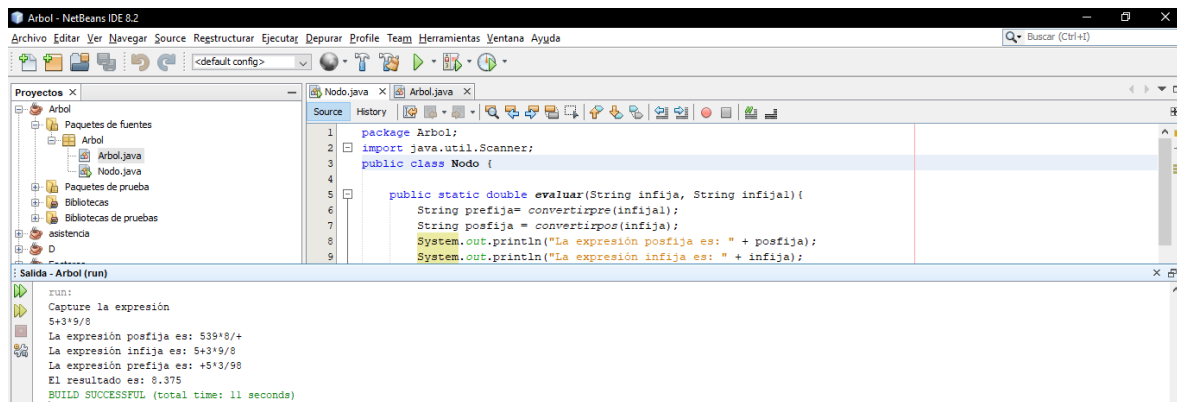
Con este programa se obtuvo en pantalla la impresión de la operación ingresada por el usuario en sus respectivos ordenes Posfija, Infija y Prefija. Se capturo la siguiente operación y tenemos como resultado lo que se muestra a continuación:

$$5+3*9/8$$

La expresión posfija es: 539*8/+

La expresión infija es: 5+3*9/8

La expresión prefija es: +5*3/98

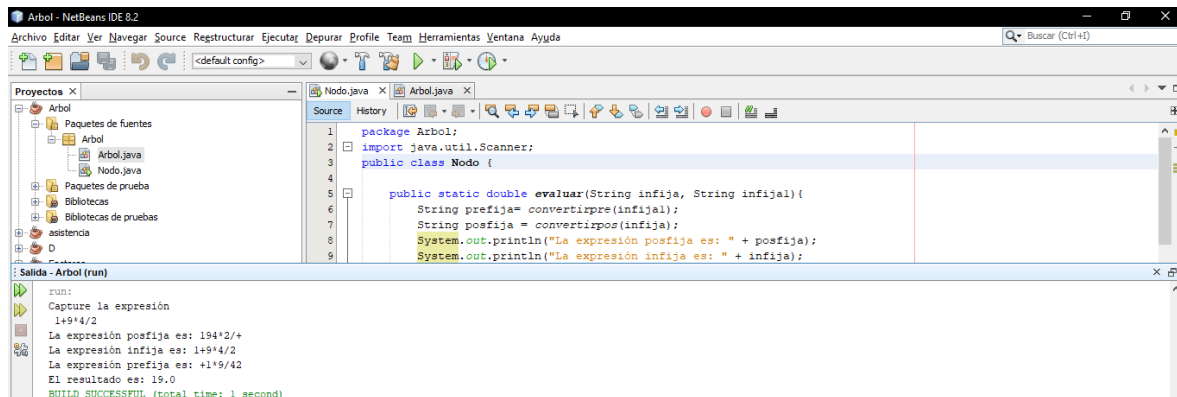


The screenshot shows the NetBeans IDE interface. The 'Proyectos' (Projects) window on the left shows a project named 'Arbol' with sub-packages 'Paquetes de fuentes', 'Paquetes de prueba', and 'Bibliotecas'. The 'Arbol.java' file is open in the editor. The code in 'Arbol.java' is as follows:

```
1 package Arbol;
2 import java.util.Scanner;
3 public class Arbol {
4
5     public static double evaluar(String infija, String infija){
6         String prefija= convertirpre(infija);
7         String posfija = convertirpos(infija);
8         System.out.println("La expresión posfija es: " + posfija);
9         System.out.println("La expresión infija es: " + infija);
10    }
```

The 'Salida - Arbol (run)' window at the bottom shows the output of the program:

```
run:
Capture la expresión
5+3*9/8
La expresión posfija es: 539*8/+
La expresión infija es: 5+3*9/8
La expresión prefija es: +5*3/98
El resultado es: 8.375
BUILD SUCCESSFUL (total time: 11 seconds)
```



The screenshot shows the NetBeans IDE interface with the same project 'Arbol'. The code in 'Arbol.java' is the same as in the previous screenshot. The 'Salida - Arbol (run)' window at the bottom shows the output of the program for a different input:

```
run:
Capture la expresión
1+9*4/2
La expresión posfija es: 194*2/+
La expresión infija es: 1+9*4/2
La expresión prefija es: +1*9/42
El resultado es: 19.0
BUILD SUCCESSFUL (total time: 1 second)
```

Conclusión

Es un buen programa para comenzar el curso de autómatas II, ya que al momento de que nosotros ingresamos una expresión nos muestra cómo sería el orden/recorrido de la misma operación, en este caso utilizamos 2 expresiones, las cuales fueron: 5+3*9/8 y 1+9*4/2, y como se puede apreciar la salida del programa nos muestra 3 expresiones de la operación, que es Posfija, Infija y Prefija, y así mismo el resultado final de la misma.

El programa es fácil de realizar con la ayuda de algunos videos o libros, ya que en algunos sitios web muestran trabajos similares al solicitado en este proyecto.

Bibliografía

Juan Carlos Zuluaga (2015), Evaluador de Funciones en Java (1/3), 02-Junio-2018, Sitios Web:

<https://www.youtube.com/watch?v=xKhA9W1fUZU&t=2016s>

Víctor Viera Balanta (2012), Estructura de Datos-Expresiones Posfijas, 02-Junio-2018, Sitio web:

<https://www.youtube.com/watch?v=wciAZR1BnSY>

Cristian Henao (2014), Prefijo y Posfijo en Java, 02-Junio-2018, Sitio web:

<https://www.youtube.com/watch?v=eNLInCujKwc>