MANUAL TECNICO

PROYECTO 1

José Ricardo Menocal Kong
202000886 OLC1

MANUAL TECNICO

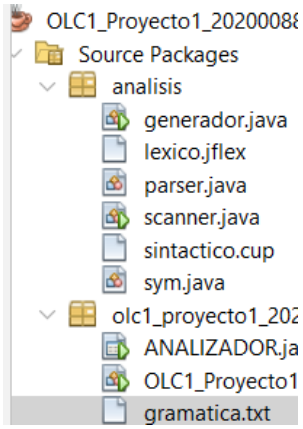
Informe sobre el programa

Manual dirigido únicamente y especialmente al técnico u programador que haga un estudio detallado del programa, fue desarrollado en el lenguaje java versión 17.0.2 2022-01-18 LTS, es un software que contiene distintas funciones que nos permite el análisis léxico y sintáctico para la realización de operaciones de un archivo de entrada tipo ".df".

Requerimientos del sistema

- Sistema operativo windos 7, windos 8, windos 10, windos 11 .
- Procesador pentium hasta procesadores de gama alta.
- 1Gb de ram
- Editor de texto utilizado visual Studio Code.

Estructura del programa (codigo)



Se desarrolló por paquetes, los cuales son:

1. ANALISIS:

En esta parte se integra los analizadores como lo es el analizador léxico, sintácticos, símbolos, y generador de estos de igual forma se explica por medio de comentarios el código.

A. Léxico

Se conforma por 3 partes:

1. Importaciones y código de usuario:

```
package analisis;

//importaciones si fuese necesario
import java_cup.runtime.Symbol;

%%

// codigo de usuario
%{
    String cadena="";
%}

%init{
    yyline = 1;
    yycolumn = 1;
%init}
```

2. Características de JFLEX:

```
//caracteristicas propias de jflex
%cup
%class scanner //nombre de la clase
%public // tipo de la clase
%line // conteo de lineas
%char // conteo de caracteres
%column // conteo de columnas
%full // reconocimiento de caracteres
%debug
%ignorecase // insensitive case

// creacion de estados si fuese necesario
%state CADENA
```

3. Declaración de tokens

```
TKINTE = "!"
TKARRO = "@"
TKCI = "["
TKCD = "]"
TKPI = "("
TKPD = ")"
TKASIGNACIONI = "<-"
TKASIGNACIOND = "->"
TKDDOBLEP = "::"
TKDOBLEP = ":"
TKCOMA = ","
TKPCOMA = ";"
TKBLANCOS = [\ \r\t\f\n]+
TKNUMEROS = [0-9]+("."[0-9]+)?


//palabras reservadas
PROGRAM = "PROGRAM"
END = "END;"
VAR= "VAR"
ARR = "ARR:"
SUM = "SUM"
RES = "RES"
MUL = "MUL"
DIV = "DIV"
MODULO = "MOD"
MEDIA = "Media"
MEDIANA = "Mediana"
MOD = "Moda"
VARIANZA = "Varianza"
MAX = "Max"
MIN = "Min"
CONSOLE_PRINT = "console::print="
```

```
MIN = "Min"
CONSOLE_PRINT = "console::print="
CONSOLE_COLUMN = "console::column= "
DOUBLE = "DOUBLE"
CHAR = "CHAR[]"


%%
<YYINITIAL> {TKINTE}     {return new Symbol(sym.TKINTE,yyline,yycolumn,yytext());}
<YYINITIAL> {TKARRO}     {return new Symbol(sym.TKARRO,yyline,yycolumn,yytext());}
<YYINITIAL> {TKCI}     {return new Symbol(sym.TKCI,yyline,yycolumn,yytext());}
<YYINITIAL> {TKCD}     {return new Symbol(sym.TKCD,yyline,yycolumn,yytext());}
<YYINITIAL> {TKPI}     {return new Symbol(sym.TKPI,yyline,yycolumn,yytext());}
<YYINITIAL> {TKPD}     {return new Symbol(sym.TKPD,yyline,yycolumn,yytext());}
<YYINITIAL> {TKASIGNACIONI}     {return new Symbol(sym.TKASIGNACIONI,yyline,yycolumn,yytext());}
<YYINITIAL> {TKASIGNACIOND}     {return new Symbol(sym.TKASIGNACIOND,yyline,yycolumn,yytext());}
<YYINITIAL> {TKDDOBLEP}     {return new Symbol(sym.TKDDOBLEP,yyline,yycolumn,yytext());}
<YYINITIAL> {TKDOBLEP}     {return new Symbol(sym.TKDOBLEP,yyline,yycolumn,yytext());}
<YYINITIAL> {TKCOMA}     {return new Symbol(sym.TKCOMA,yyline,yycolumn,yytext());}
<YYINITIAL> {TKPCOMA}     {return new Symbol(sym.TKPCOMA,yyline,yycolumn,yytext());}
<YYINITIAL> {TKNUMEROS}     {return new Symbol(sym.TKNUMEROS,yyline,yycolumn,yytext());}
<YYINITIAL> {TKBLANCOS}     {}


<YYINITIAL> {PROGRAM}     {return new Symbol(sym.PROGRAM,yyline,yycolumn,yytext());}
<YYINITIAL> {END}     {return new Symbol(sym.END,yyline,yycolumn,yytext());}
<YYINITIAL> {VAR}     {return new Symbol(sym.VAR,yyline,yycolumn,yytext());}
<YYINITIAL> {ARR}     {return new Symbol(sym.ARR,yyline,yycolumn,yytext());}
<YYINITIAL> {SUM}     {return new Symbol(sym.SUM,yyline,yycolumn,yytext());}
<YYINITIAL> {RES}     {return new Symbol(sym.RES,yyline,yycolumn,yytext());}
<YYINITIAL> {MUL}     {return new Symbol(sym.MUL,yyline,yycolumn,yytext());}
<YYINITIAL> {DIV}     {return new Symbol(sym.DIV,yyline,yycolumn,yytext());}
<YYINITIAL> {MODULO}     {return new Symbol(sym.MODULO,yyline,yycolumn,yytext());}
<YYINITIAL> {MEDIA}     {return new Symbol(sym.MEDIA,yyline,yycolumn,yytext());}
<YYINITIAL> {MEDIANA}     {return new Symbol(sym.MEDIANA,yyline,yycolumn,yytext());}
```

```
<YYINITIAL> {PROGRAM}     {return new Symbol(sym.PROGRAM,yyline,yycolumn,yytext());}
<YYINITIAL> {END}     {return new Symbol(sym.END,yyline,yycolumn,yytext());}
<YYINITIAL> {VAR}     {return new Symbol(sym.VAR,yyline,yycolumn,yytext());}
<YYINITIAL> {ARR}     {return new Symbol(sym.ARR,yyline,yycolumn,yytext());}
<YYINITIAL> {SUM}     {return new Symbol(sym.SUM,yyline,yycolumn,yytext());}
<YYINITIAL> {RES}     {return new Symbol(sym.RES,yyline,yycolumn,yytext());}
<YYINITIAL> {MUL}     {return new Symbol(sym.MUL,yyline,yycolumn,yytext());}
<YYINITIAL> {DIV}     {return new Symbol(sym.DIV,yyline,yycolumn,yytext());}
<YYINITIAL> {MODULO}     {return new Symbol(sym.MODULO,yyline,yycolumn,yytext());}
<YYINITIAL> {MEDIA}     {return new Symbol(sym.MEDIA,yyline,yycolumn,yytext());}
<YYINITIAL> {MEDIANA}     {return new Symbol(sym.MEDIANA,yyline,yycolumn,yytext());}
<YYINITIAL> {MOD}     {return new Symbol(sym.MOD,yyline,yycolumn,yytext());}
<YYINITIAL> {VARIANZA}     {return new Symbol(sym.VARIANZA,yyline,yycolumn,yytext());}
<YYINITIAL> {MAX}     {return new Symbol(sym.MAX,yyline,yycolumn,yytext());}
<YYINITIAL> {MIN}     {return new Symbol(sym.MIN,yyline,yycolumn,yytext());}
<YYINITIAL> {CONSOLE_PRINT}     {return new Symbol(sym.CONSOLE_PRINT,yyline,yycolumn,yytext());}
<YYINITIAL> {CONSOLE_COLUMN}     {return new Symbol(sym.CONSOLE_COLUMN,yyline,yycolumn,yytext());}
<YYINITIAL> {DOUBLE}     {return new Symbol(sym.DOUBLE,yyline,yycolumn,yytext());}
<YYINITIAL> {CHAR}     {return new Symbol(sym.CHAR,yyline,yycolumn,yytext());}
<YYINITIAL> [\"]     {yybegin(CADENA);cadena="";}

<CADENA>     {
    [\"]    {String tmp=cadena; cadena=""; yybegin(YYINITIAL); return new Symbol(sym.CADENA, yycolumn,yyline,tmp);}
    [^\"]   {cadena+=yytext();}
```

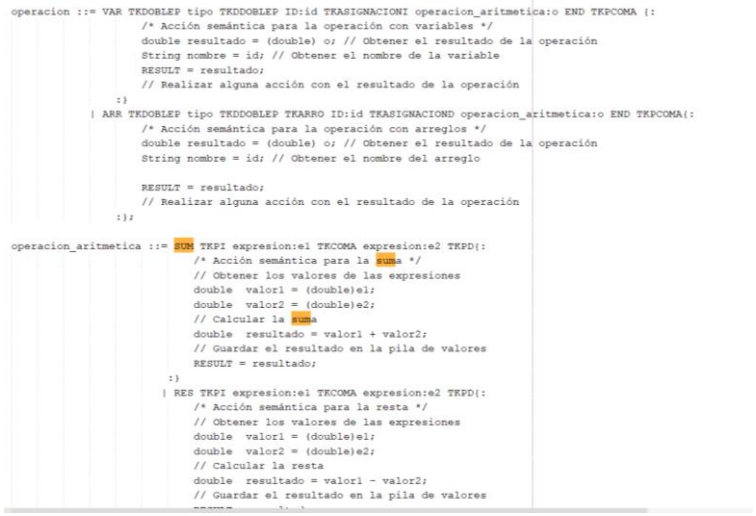
B. Sintáctico:

Acá podemos encontrar todo el orden que debe de seguir el programa para que el texto sea aceptado

```
package analisis;

// importaciones si fuese necesario
import java_cup.runtime.Symbol;

//import java.util.Math;

//parser (codigo accesible usando punto)  p.variable
parser code
{:
    public void syntax_error(Symbol s){
        System.out.println("Error sintactico en la linea " +
            (s.left) + " y columna " + (s.right) +
            ". No se esperaba el componente: " + (s.value) + ".");
    }

    public void unrecovered_syntax_error (Symbol s) throws java.lang.Exception{
        System.out.println("Error sintactico no recuperable en la linea " +
            (s.left) + " y columna " + (s.right) +
            ". No se esperaba el componente: " + (s.value) + ".");
    }
:}

// codigo interno de cup
action code
{:
    String codigoUsr="";
:}

/* Especificación de símbolos terminales */
terminal TKARRO, TKCI, TKCD, TKPI, TKPD, TKASIGNACIONI, TKCOMA, TKPCOMA, TKNUMEROS, PROGRAM, END, VAR, ARR, SUM, RES,
        MUL, DIV, MODULO, MEDIA, MEDIANA, MOD, VARIANZA, MAX, MIN, CONSOLE_PRINT, CONSOLE_COLUMN, DOUBLE, CHAR, CADENA,TKDDOBLEP,TKDOBLEP,TKASIGNACIOND,TKINT

non terminal programa, declaraciones, declaracion, instrucciones, tipo,  expresiones, expresion, lista_valores, valores, valor,
        instruccion, asignacion, operacion, estadistica, impresion_consola, impresion_arreglo, comentario,
```

```
:}

/* Especificación de símbolos terminales */
terminal TKARRO, TKCI, TKCD, TKPI, TKPD, TKASIGNACIONI, TKCOMA, TKPCOMA, TKNUMEROS, PROGRAM, END, VAR, ARR, SUM, RES,
        MUL, DIV, MODULO, MEDIA, MEDIANA, MOD, VARIANZA, MAX, MIN, CONSOLE_PRINT, CONSOLE_COLUMN, DOUBLE, CHAR, CADENA,TKDDOBLEP,TKDOBLEP,TKASIGNACIOND,TKINTE

non terminal programa, declaraciones, declaracion, instrucciones, tipo,  expresiones, expresion, lista_valores, valores, valor,
       instruccion, asignacion, operacion, estadistica, impresion_consola, impresion_arreglo, comentario,
       operacion_aritmetica, arreglo_double, arreglo;

non terminal String ID ;

/* Producciones */
start with programa;

programa ::= PROGRAM declaraciones instrucciones END PROGRAM {:  RESULT=codigoUsr;   :};

declaraciones ::= declaraciones declaracion
            | /* Producción vacia */;

declaracion ::= VAR TKDOBLEP tipo TKDDOBLEP ID:id TKASIGNACIONI arreglo_double:arr END TKPCOMA{:
                    /* Acción semántica para la declaración de variables */
                    double[] arreglo = (double[]) arr ; // Obtener el arreglo de valores
                    String nombre = id; // Obtener el nombre de la variable
                    // Realizar alguna acción con la variable declarada
                :}
              | ARR TKDOBLEP tipo TKDDOBLEP TKARRO ID:id TKASIGNACIONI lista_valores:arr  END TKPCOMA{:
                    /* Acción semántica para la declaración de arreglos */
                    double[] arreglo = (double[]) arr; // Obtener el arreglo de valores
                    String nombre = id; // Obtener el nombre del arreglo
                    // Realizar alguna acción con el arreglo declarado
                :};

tipo ::= DOUBLE
      | CHAR;
```

```
                    /* Acción semántica para la lista de valores en un arreglo */
                    RESULT = (double[]) v; // Convertir la lista de valores a un arreglo de tipo double
                :};

valores ::= valores:vs TKCOMA valor:v {:
                /* Acción semántica para la lista de valores */
                double[] arreglo = (double[]) vs; // Obtener la lista de valores anterior
                double valor = (double) v; // Obtener el nuevo valor
                // Crear un nuevo arreglo con el valor añadido
                double[] nuevoArreglo = new double[arreglo.length + 1];
                // Copiar los valores anteriores al nuevo arreglo
                System.arraycopy(arreglo, 0, nuevoArreglo, 0, arreglo.length);
                // Añadir el nuevo valor al final del nuevo arreglo
                nuevoArreglo[arreglo.length] = valor;
                RESULT = nuevoArreglo; // Actualizar el resultado con el nuevo arreglo
            :}
          | valor:vs {: RESULT = new double[]{(double)vs}; :};

valor ::= TKNUMEROS:n {: RESULT = Double.parseDouble((String)n); :}
        | CADENA
        | ID;

instrucciones ::= instrucciones instruccion
                | instruccion;

instruccion ::= asignacion
              | operacion
              | estadistica
              | impresion_consola
              | impresion_arreglo
              | comentario;


operacion ::= VAR TKDOBLEP tipo TKDDOBLEP ID:id TKASIGNACIONI operacion_aritmetica:o END TKPCOMA {:
```



```
```

```
        :)
        | MUL TKPI expresion:e1 TKCOMA expresion:e2 TKPD(:
            /* Acción semántica para la multiplicación */
            // Obtener los valores de las expresiones
            double  valor1 = (double)e1;
            double  valor2 = (double)e2;
            // Calcular la multiplicación
            double  resultado = valor1 * valor2;
            // Guardar el resultado en la pila de valores
            RESULT = resultado;
        :)
        | DIV TKPI expresion:e1 TKCOMA expresion:e2 TKPD(:
            /* Acción semántica para la división */
            // Obtener los valores de las expresiones
            double  valor1 = (double)e1;
            double  valor2 = (double)e2;
            // Verificar si se intenta dividir por cero
            if (valor2 == 0) {
                throw new RuntimeException("¡Error! División por cero.");
            }
            // Calcular la división
            double  resultado = valor1 / valor2;
            // Guardar el resultado en la pila de valores
            RESULT = resultado;
        :)
        | MODULO TKPI expresion:e1 TKCOMA expresion:e2 TKPD(:
            /* Acción semántica para el módulo */
            // Obtener los valores de las expresiones
            double  valor1 = (double)e1;
            double  valor2 = (double)e2;
            // Verificar si se intenta realizar el módulo por cero
            if (valor2 == 0) {
                throw new RuntimeException("¡Error! División por cero.");
            }
            // Calcular el módulo
```

```
                if (valor2 == 0) {
                    throw new RuntimeException("¡Error! División por cero.");
                }
                // Calcular el módulo
                double  resultado = valor1 % valor2;
                // Guardar el resultado en la pila de valores
                RESULT = resultado;
            :};

estadistica ::= MEDIA TKPI arreglo_double TKPD
              | MEDIANA TKPI arreglo_double TKPD
              | MOD TKPI arreglo_double TKPD
              | VARIANZA TKPI arreglo_double TKPD
              | MAX TKPI arreglo_double TKPD
              | MIN TKPI arreglo_double TKPD;

arreglo_double ::= lista_valores;

impresion_consola ::= CONSOLE_PRINT TKPI expresiones:e TKPD (:  codigoUsr+=e+"\n";   :);

expresiones ::= expresiones:es TKCOMA expresion:e (: RESULT = es + ", " + e; :)
              | expresion:es (: RESULT = es; :);

impresion_arreglo ::= CONSOLE_COLUMN TKPI CADENA TKARRO arreglo TKPD ;

arreglo ::= ID
          | lista_valores;

comentario ::= CADENA;

expresion ::= TKNUMEROS
            | CADENA
            | ID
```

C. Generador:

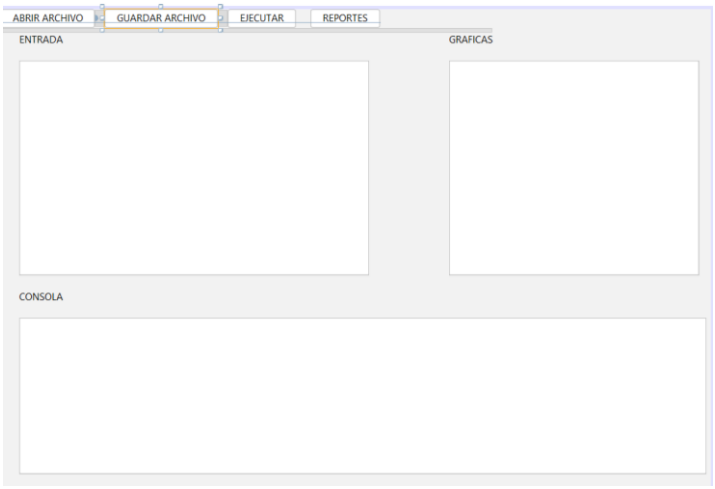
En este caso esta clase, crea y guarda lo que son los símbolos del analizador , también realiza el parser del mismo.

```
public class generador {

    public static void main(String[] args) {
        generarCompilador();
    }

    public static void generarCompilador() {
        try {
            String ruta = "src/analisis/";
            /*
            ruta del archivo jflex
            -d -> ruta donde generar archivo de salida
            ruta de salida
             */

            String Flex[] = {ruta + "lexico.jflex", "-d", ruta};
            jflex.Main.generate(Flex);
            /*
            -destdir indica la ruta donde se generara la salida
            ruta de salida
            -parser indican el nombre del archivo
            parser
            ruta del archivo cup
             */

            String Cup[] = {"-destdir", ruta, "-parser", "parser", ruta + "sintactico.cup"};
            java_cup.Main.main(Cup);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2. Olc1_proyecto1_202000886

Este paquete almacena varias cosas, lo cuales son La interfaz gráfica del programa y de la clase para poder visualizar la interfaz.

A. Interfaz Grafica



Funcionalidad boton EJECUTAR:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String texto = ENTRADATXT.getText();
    try {
        scanner s = new scanner(new BufferedReader(new StringReader(texto)));
        parser p = new parser(s);
        var result = p.parse();

        SALIDATXT.setText((String) result.value);
    } catch (Exception e) {
        System.out.println(e);
    }               // TODO add your handling code here:
```

Con esto ya claro el programador o persona que quiere hacer uso de este programa lo puede hacer, este solo es una pequeña explicación de la funcionalidad y estructura de este programa, este programa puede ser más optimo y puede mejor para versiones futuras.