Contenido

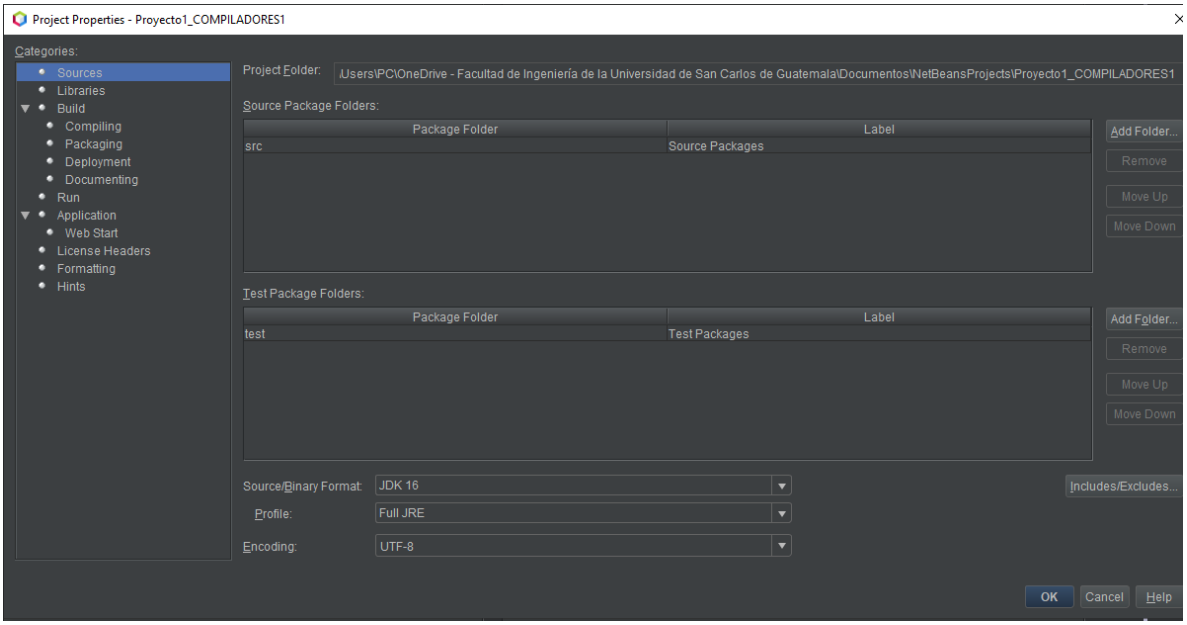
Especificaciones:

IDE utilizado: NetBeans.

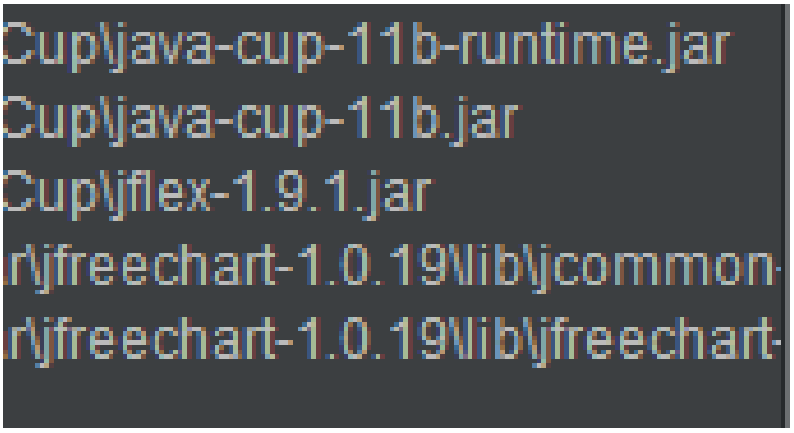
Lenguaje Utilizado: JAVA

Sistema Operativo: Windows 10 (64)

Acerca del Proyecto:



Librerías Utilizadas :



Analizadores:

“léxico.jflex”: Utilizado para analizar lexicamente el código statpy.

```
"&&"   { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "AND", yytext().toString()); System.out.println("AND: "+yytext()); return new Symbol(sym.AND, yyline, yycolumn, yytext());}
"||"   { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "OR", yytext().toString()); System.out.println("OR: "+yytext()); return new Symbol(sym.OR, yyline, yycolumn, yytext());}
"/"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "DIVISION", yytext().toString()); System.out.println("DIVISION: "+yytext()); return new Symbol(sym.DIVISION, yyline, yycolumn, yytext());}
"!="   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "DISTINTO", yytext().toString()); System.out.println("DISTINTO: "+yytext()); return new Symbol(sym.DISTINTO, yyline, yycolumn, yytext());}
"^"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "POTENCIA", yytext().toString()); System.out.println("POTENCIA: "+yytext()); return new Symbol(sym.POTENCIA, yyline, yycolumn, yytext());}
"?"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "CBOOL", yytext().toString()); System.out.println("CBOOL: "+yytext()); return new Symbol(sym.CBOOL, yyline, yycolumn, yytext());}
"%"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PORCENTAJE", yytext().toString()); System.out.println("PORCENTAJE: "+yytext()); return new Symbol(sym.PORCENTAJE, yyline, yycolumn, yytex
","    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PCOMA", yytext().toString()); System.out.println("PCOMA: "+yytext()); return new Symbol(sym.PCOMA, yyline, yycolumn, yytext());}
"!"   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "NOT", yytext().toString()); System.out.println("NOT: "+yytext()); return new Symbol(sym.NOT, yyline, yycolumn, yytext());}
"<"  { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "MENORK", yytext().toString()); System.out.println("MENORK: "+yytext()); return new Symbol(sym.MENORK, yyline, yycolumn, yytext());}
">"  { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "MAYORK", yytext().toString()); System.out.println("MAYORK: "+yytext()); return new Symbol(sym.MAYORK, yyline, yycolumn, yytext());}
"<="  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "MENORIK", yytext().toString()); System.out.println("MENORIK: "+yytext()); return new Symbol(sym.MENORIK, yyline, yycolumn, yytext());}
"="   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "EQUALS", yytext().toString()); System.out.println("EQUALS: "+yytext()); return new Symbol(sym.EQUALS, yyline, yycolumn, yytext());}
">="   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "MAYORIK", yytext().toString()); System.out.println("MAYORIK: "+yytext()); return new Symbol(sym.MAYORIK, yyline, yycolumn, yytext());}
"["  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "CORCHETEA", yytext().toString()); System.out.println("CORCHETEA: "+yytext()); return new Symbol(sym.CORCHETEA, yyline, yycolumn, yytext());}
"]"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "CORCHETEC", yytext().toString());System.out.println("CORCHETEC: "+yytext()); return new Symbol(sym.CORCHETEC, yyline, yycolumn, yytext());}

"main"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_MAIN", yytext().toString()); System.out.println("MAIN: "+yytext()); return new Symbol(sym.MAIN, yyline, yycolumn, yytext());}
"char"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_CHAR", yytext().toString());System.out.println("CHAR_TYPE: "+yytext()); return new Symbol(sym.CHAR_TYPE, yyline, yycolumn, yytext());}
"int"      {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_INT", yytext().toString()); System.out.println("INT_TYPE: "+yytext()); return new Symbol(sym.INT_TYPE, yyline, yycolumn, yytext());}
"bool"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_BOOL", yytext().toString());System.out.println("BOOL_TYPE: "+yytext()); return new Symbol(sym.BOOL_TYPE, yyline, yycolumn, yytext());}
"double"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_DOUBLE", yytext().toString());System.out.println("DOUBLE_TYPE: "+yytext()); return new Symbol(sym.DOUBLE_TYPE, yyline, yycolumn,
"string"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_STRING", yytext().toString());System.out.println("STRING_TYPE: "+yytext()); return new Symbol(sym.STRING_TYPE, yyline, yycolumn, yyte
"void"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_VOID", yytext().toString());System.out.println("VOID: "+yytext()); return new Symbol(sym.VOID, yyline, yycolumn, yytext());}
"break"   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_BREAK", yytext().toString());System.out.println("BREAK: "+yytext()); return new Symbol(sym.BREAK, yyline, yycolumn, yytext());}
"case"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_CASE", yytext().toString());System.out.println("CASE: "+yytext()); return new Symbol(sym.CASE, yyline, yycolumn, yytext());}
"catch"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_CATCH", yytext().toString());System.out.println("CATCH: "+yytext()); return new Symbol(sym.CATCH, yyline, yycolumn, yytext());}
"do"       {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_DO", yytext().toString());System.out.println("DO: "+yytext()); return new Symbol(sym.DO, yyline, yycolumn, yytext());}
"else"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_ELSE", yytext().toString());System.out.println("ELSE: "+yytext()); return new Symbol(sym.ELSE, yyline, yycolumn, yytext());}
"for"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_FOR", yytext().toString());System.out.println("FOR: "+yytext()); return new Symbol(sym.FOR, yyline, yycolumn, yytext());}
"if"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_IF", yytext().toString());System.out.println("IF: "+yytext()); return new Symbol(sym.IF, yyline, yycolumn, yytext());}
"switch"  { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_SWITCH", yytext().toString()); System.out.println("SWITCH: "+yytext()); return new Symbol(sym.SWITCH, yyline, yycolumn, yytext());}
"null"  {System.out.println("NULL: "+yytext()); return new Symbol(sym.NULL, yyline, yycolumn, yytext());}
"new" {System.out.println("NEW: "+yytext()); return new Symbol(sym.NEW, yyline, yycolumn, yytext());}
"public"  {System.out.println("PUBLIC: "+yytext()); return new Symbol(sym.PUBLIC, yyline, yycolumn, yytext());}
"return"   {System.out.println("RETURN: "+yytext()); return new Symbol(sym.RETURN, yyline, yycolumn, yytext());}
"while"   {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_WHILE", yytext().toString());System.out.println("WHILE: "+yytext()); return new Symbol(sym.WHILE, yyline, yycolumn, yytext());}
"private" {System.out.println("PRIVATE: "+yytext()); return new Symbol(sym.PRIVATE, yyline, yycolumn, yytext());}
"true"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_TRUE", yytext().toString());System.out.println("TRUE: "+yytext()); return new Symbol(sym.TRUE, yyline, yycolumn, yytext());}
"Console.Write"  {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "R_PRINT", yytext().toString());System.out.println("PRINT: "+yytext()); return new Symbol(sym.PRINT, yyline, yycolumn, yytext());}
```

```
%%

%{
public String errores="";
%}

%class Lexico
%public
%line
%column
%char
%cup
%unicode
%ignorecase

%{
%}

ESPACIO= [ \t\r\n]+
COMMENT_MULTILINE="/*"([^*]|(\*+[^*/]))+"*/"
COMMENT_SIMPLE="//".*
NUM = [0-9]+(\.[0-9]*)?
ESP = [ -/@{-`{-}]
ESCAPADOS = "\\\""|"\\\"|"\\n"
NO_ESCAPADOS = [^\"]
ID = [a-zA-Z_][a-zA-Z0-9_]*
CHAR = \' ([^\"]|"\\\"")+ \'
STRING = \" ([^\"]|"\\\"")+ \"

%%
":"     { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "NUM", yytext().toString()); System.out.println("DOS_PUNTOS: "+yytext()); return new Symbol(sym.DOS_PUNTOS, yyline, yycolumn, yytext()); }
","     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "COMA", yytext().toString()); System.out.println("COMA: "+yytext()); return new Symbol(sym.COMA, yyline, yycolumn, yytext());  }
"{"     { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "LLAVE_A", yytext().toString()); System.out.println("LLAVE_A: "+yytext()); return new Symbol(sym.LLAVE_A, yyline, yycolumn, yytext()); }
"}"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "LLAVE_C", yytext().toString());System.out.println("LLAVE_C: "+yytext()); return new Symbol(sym.LLAVE_C, yyline, yycolumn, yytext()); }
"("     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARA", yytext().toString());System.out.println("PARA: "+yytext()); return new Symbol(sym.PARA, yyline, yycolumn, yytext());  }
")"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARC", yytext().toString());System.out.println("PARC: "+yytext()); return new Symbol(sym.PARC, yyline, yycolumn, yytext());   }
"."     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARC", yytext().toString()); System.out.println("DOT: "+yytext()); return new Symbol(sym.DOT, yyline, yycolumn, yytext());   }
"+"     { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARC", yytext().toString()); System.out.println("MAS: "+yytext()); return new Symbol(sym.MAS, yyline, yycolumn, yytext());  }
"-"    {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARC", yytext().toString()); System.out.println("MENOS: "+yytext()); return new Symbol(sym.MENOS, yyline, yycolumn, yytext());  }
"*"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "PARC", yytext().toString()); System.out.println("MULTI: "+yytext()); return new Symbol(sym.MULTI, yyline, yycolumn, yytext());   }
"$"     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "DOLAR", yytext().toString()); System.out.println("DOLAR: "+yytext()); return new Symbol(sym.DOLAR, yyline, yycolumn, yytext()); }
```

Sintactico.cup :

Archivo utilizado para analizar sintácticamente el código statpy.

Gramatica recursividad por la izquierda.

Importantes:

```
// ------> Precedencia
precedence left MAYORK, MENORK, MAYORIK, MENORIK, EQUALS,DISTINTO,NOT;
precedence left OR;
precedence left AND;
precedence left MENOS;
precedence left MAS;
precedence left MULTI;
precedence left MAS;
precedence left MULTI;
precedence left PORCENTAJE, DIVISION;
precedence left POTENCIA;
```

Result:

Debe ser una variable tipo LinkedList.

```
inicio ::= VOID MAIN PARA PARC LLAVE_A lista_instr:lista LLAVE_C
                                {:

                                 LinkedList<String> lista1 = new LinkedList<>();
                                   lista1.add("def main (): ");
                                  Traductor.Traduc.contador++;
                                   lista1.addAll(Traductor.Traduc.tabulaciones(lista));
                                   Traductor.Traduc.contador--;
                                    lista1.add("if_name_= \" _main_ \": \n \t \t main()" );
                                    Traductor.Traduc.traduccion = lista1;
                                :}
;
```

```
// ---------- Paquete e importaciones ----------
package Analizadores;

import java_cup.runtime.*;
import java.util.LinkedList;


//-------> Codigo para el parser
//-------> Declaracion de variables, funciones y funciones de error

parser code
{:
    public static String errores_s = "";
    public static String errores_2s = "";
    public void syntax_error(Symbol s)
    {
        System.err.println("Error Sintactico: "+ s.value + " - Fila: " + s.right + " - Columna: " + s.left   );
        Errores.DataErrores.addError(" Sintactico  " , s.right, s.left, s.value.toString());
        errores_s += "<tr><td>Sintáctico</td><td> Error sintactico encontrado: \""+s.value+"\"</td><td>"+(s.left+1)+"</td><td>"+(s.right+1)+"</td></tr>\n";
    }

    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception
    {
        Errores.DataErrores.addError(" Sintactico sin recuperacion " , s.right, s.left, s.value.toString());
         errores_s += "<tr><td>Sintáctico</td><td> Error sintactico encontrado: \""+s.value+"\"</td><td>"+(s.left+1)+"</td><td>"+(s.right+1)+"</td></tr>\n";
        System.err.println("Error Sintactico: "+ s.value + " - Fila: " + s.right + " - Columna: " + s.left + ". Sin recuperacion." );
    }
String TituloBarras ;
String TituloX;
String TituloY;
String TituloPie;
Object valorId;
:}


//-------> Codigo para las acciones gramaticales (no tocar)
action code
{: :}
```

Generar archivos analizadores “.java”:

Clic derecho, Run File.

```
public static void main(String[] args) {
    // TODO code application logic heretring[] args) {
    // TODO code applicati
    try{
        String ruta = "src/Analizadores/";
        String opcFlex[ ] = {ruta + "lexico.jflex","-d",ruta};
        jflex.Main.generate( argv: opcFlex);
        System.out.println( x: "Lexico Listo");
        String opcCUP[ ] = {"-destdir",ruta,"-parser","Parser",ruta+"sintactico.cup"};
        System.out.println( x: "Cup: ");
        java_cup.Main.main( argv: opcCUP);
    } catch (Exception e){
        System.out.println( x: "No se ha podido generar los analizadores");
        e.printStackTrace();
    }
}
}
```

Almacenar errores:

```java
public class DataErrores {
        public static LinkedList<Error> errores = new LinkedList<>();

public static void addError(String tipo, int fila, int columna, String valor) {
    Error error = new Error(tipo, fila, columna, valor);
    errores.add( e: error);
}

    public static void showErrores() {
        System.out.println( x: "Lista de Errores:");
        for (Error error : errores) {
            System.out.println("Tipo: " + error.getTipo());
            System.out.println("Fila: " + error.getFila());
            System.out.println("Columna: " + error.getColumna());
            System.out.println("Valor: " + error.getValor());
            System.out.println( x: "-------------------------");
        }
    }



}
```

Almacenar Símbolos:

Clase utilizada para almacenar los símbolos según su:

- Token
- Lexema
- Fila
- Columna

```
public class DataSimbolos {
    public static LinkedList<Simbolo> Simbolos = new LinkedList<>();

    public static void addSimbolo(int fila, int columna, String token, String lexema) {
            System.out.println("Fila: " + fila);
            System.out.println("Columna: " + columna);
            System.out.println("Token: " + token);
            System.out.println("Lexema: " + lexema);
            System.out.println("--------------------");
        Simbolo simbolo = new Simbolo(fila, columna, token, lexema);
        Simbolos.add(simbolo);
    }
        public static void show() {
          System.out.println("Símbolos:");

          for (Simbolo simbolo : Simbolos) {
            System.out.println("Fila: " + simbolo.getFila());
            System.out.println("Columna: " + simbolo.getColumna());
            System.out.println("Token: " + simbolo.getToken());
            System.out.println("Lexema: " + simbolo.getLexema());
            System.out.println("--------------------");
        }


    }
```

Se almacenan directamente en los archivos “.jflex” de la siguiente manera:

```
"."     { Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "NUM", yytext().toString());
","     {Tabla_Simbolos.DataSimbolos.addSimbolo(yyline ,yycolumn, "COMA", yytext().toString());
```

Tabulaciones y Traducción:

```
public class Traduc {
    public static HashMap variables = new HashMap();
    public static int contador = 0;
    public static LinkedList<String> traduccion = new LinkedList<>();

  public static LinkedList<String>  tabulaciones(LinkedList<String> lista){
    String tabs = "";
    for (int i = 0; i < contador; i++) {
        tabs = "\t"+tabs;
    }

    for (int i = 0; i < lista.size(); i++) {
        lista.set( index: i, tabs+lista.get( index: i));
    }

    return lista;
  }
    public static String mostrar (){
    System.out.println( x: "Contenido de la LinkedList:");
    String texto = "";
    for (String elemento : traduccion) {
        texto += elemento + "\n";
        //System.out.println(texto);
    }

      return texto;
  }
    public static LinkedList<String>  elif (LinkedList<String> lista){
          if (!lista.isEmpty()) {
              String primerElemento = lista.get( index: 0);
              String nuevoPrimerElemento = primerElemento.replaceFirst( regex: "if",  replacement: "elif ");
              lista.set( index: 0,  element: nuevoPrimerElemento);
          }
    return lista;
  }
```

En el archivo “sintactico.cup” se hace uso de una Linkedlist en la cual se va añadiendo directamente la traducción, incrementado el contador para agregar sus tabulaciones adecuadas para Python y disminuyendo.

```
switch ::= SWITCH PARA ID PARC LLAVE_A cases:val   DEFAULT DOS_PUNTOS lista_instr:inst2 LLAVE_C
{:
    LinkedList<String> lista = new LinkedList<>();
    lista.add("def switch (case,valor)  \n switcher { " );
   Traductor.Traduc.contador++;
    lista.addAll(Traductor.Traduc.tabulaciones((LinkedList)val));
    Traductor.Traduc.contador--;

   LinkedList<String> lista2 = new LinkedList<>();
   lista2.add("default:");
   Traductor.Traduc.contador++;
    lista2.addAll(Traductor.Traduc.tabulaciones(inst2));
    Traductor.Traduc.contador--;
   Traductor.Traduc.contador++;
    lista.addAll(Traductor.Traduc.tabulaciones((LinkedList)lista2));
    Traductor.Traduc.contador--;

    lista.add("}");
    RESULT = lista;
:}
;
```

Analizar archivos “json” “

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    String codigo = txta_editor.getText();
        try{
            AnalizadoresJson.Lexico scanner = new AnalizadoresJson.Lexico(new StringReader( s: codigo));
             AnalizadoresJson.ParserJson AnalizadorJ = new AnalizadoresJson.ParserJson( s: scanner);

            AnalizadorJ.parse();

            data.dataJson.agregar();
            data.dataJson.Sho();

        }catch (Exception e) {
          }
        //Codigo omitido

}
```

Analizar archivos “sp”

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
            String codigo = txta_editor.getText();
    try {
        Lexico scanner = new Lexico (new StringReader( s: codigo));
        AnalizadorS = new Parser( s: scanner);
        AnalizadorS.parse();
        if(!AnalizadorS.errores_s.equals( anObject: "")|| !scanner.errores.equals( anObject: "")){
            System.out.println( x: "Errroreees");
            String reporte_errores = "<!doctype html>\n"
            + "<html lang=\"en\">\n"
            + "  <head>\n"
            + "    <!-- Required meta tags -->\n"
            + "    <meta charset=\"utf-8\">\n"
            + "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1, shrink-to-fit=no\">\n"
            + "\n"
            + "    <!-- Bootstrap CSS -->\n"
            + "    <link rel=\"stylesheet\" href=\"https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap
            + "\n"
            + "    <title>Hello, world!</title>\n"
            + "  </head>\n"
            + "  <body>\n"
            + "<table class=\"table table-hover table-dark\">\n"
            + "  <thead>\n"
            + "    <tr>\n"
            + "      <th scope=\"col\">Tipo de Error</th>\n"
            + "      <th scope=\"col\">DescripciÃ³n</th>\n"
            + "      <th scope=\"col\">LÃnea</th>\n"
            + "      <th scope=\"col\">Columna</th>\n"
            + "    </tr>\n"
            + "  </thead>\n"
            + "  <tbody>\n" + AnalizadorS.errores_s + scanner.errores
            + "</tbody>\n"
            + "</table>\n"
            + "    <!-- Optional JavaScript -->\n"
            + "    <!-- jQuery first, then Popper.js, then Bootstrap JS -->\n"
            + "    <script src=\"https://code.jquery.com/jquery-3.2.1.slim.min.js\" integrity=\"sha384-KJ3o2DKtIk
```

Agregar el código traducido y generar el reporte de símbolos:

```
 crearArchivo( dir: "src/REPORT_SIMBOLOS/", nombre: "Reporte de simbolos.html ", texto: codigo1);
consola.setText( t: Traductor.Traduc.mostrar());
```

Graficar:

Código utilizado para generar las gráficas, al hacerlo reiniciar los valores:

```
Graficas.Graficar.barras( Titulo: data.Valorees.TituloBarras,  TituloX: data.Valorees.TituloX,  TituloY: data.Valorees.TituloY,  valores1: data.Valorees.valores,  ejex1: data.Valorees.ejex);

Graficas.Graficar.Pie( Titulo: data.Valorees.TituloPie,  TituloX: data.Valorees.TituloX,  TituloY: data.Valorees.TituloY,  valores1: data.Valorees.valoresP,  ejex1: data.Valorees.ejexP);
 data.Valorees.TituloBarras = "";
data.Valorees.TituloX = "";
data.Valorees.TituloY = "";
data.Valorees.valores.clear();
data.Valorees.ejex.clear();
 data.Valorees.TituloPie = "";
data.Valorees.valoresP.clear();
 data.Valorees.ejexP.clear();
```

Crear archivo:

Código para crear un archivo y si el archivo existe, escribir sobre el

```
public void crearArchivo(String dir, String nombre, String texto) {
    File file = new File( parent: dir,  child: nombre);

    try {
        if (!file.exists()) {
            file.createNewFile();
        }

        try (PrintWriter pw = new PrintWriter(file)) {
            pw.write( s: texto);
        }
    } catch (IOException ex) {
        // Manejo de excepciones aquí (puedes imprimir mensajes de
        ex.printStackTrace();
    }
}
```