Universidad de San Carlos de Guatemala
Facultad de ingeniería
Ingeniería en Ciencias y Sistemas
Organización de Lenguajes y Compiladores 1

MANUAL TECNICO

Alberto Josue Hernández Armas 201903553

Guatemala 19 de sepiembre de 2022

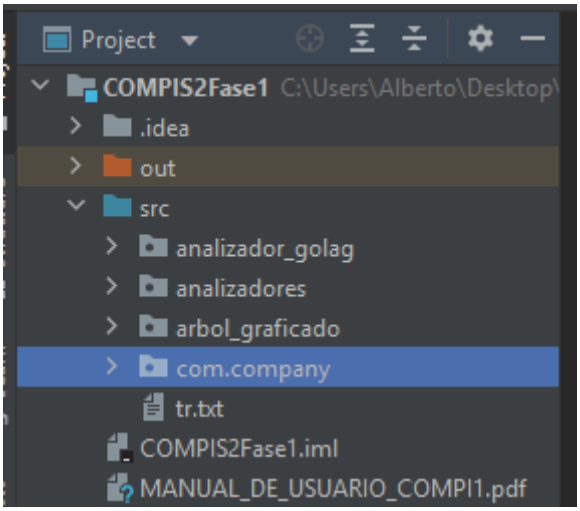
INTRODUCCION

En el presente manual técnico podremos observar cómo es que está diseñado el código y explicando que forman la funcionalidad de una manera general, teniendo como objetivo principal que sea más entendible. Determinando los métodos utilizados y explicando algunas palabras claves refiriéndonos a sus propiedades demostrando cuál es su función dentro de los bloques de código que se nos presentan a continuación.

MANUAL TECNICO

El proyecto se divide en 4 paquetes, el paquete main, donde se encuentras todas las clases abstractas de funcionalidad y la interfas grafica. Por otra parte tenemos los otros 3 paquetes, cada uno de estos cuenta con un compilador que realiza una tarea especifica, uno traduce del pseudocodigo a python, el segundo de pseudocodigo a Golang, y el tercero realiza la graficacion del árbol sintáctico de la gramatica ingresada.

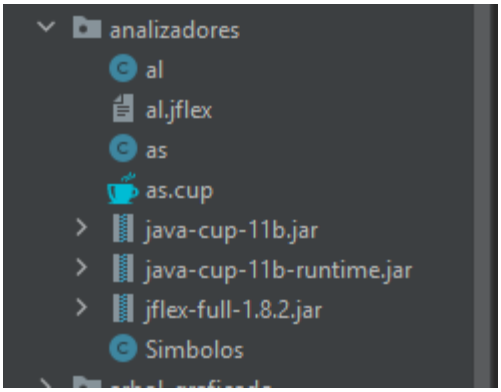
Se dividieron en paquees ya que cada uno debe ir acompañado por una librería especifica, que es la misma en los paquetes, pero que se configura de una manera diferente para cada uno. Se realiza para que las herramientas tengan un entorno ermetico y no se mire afectada por las demás.



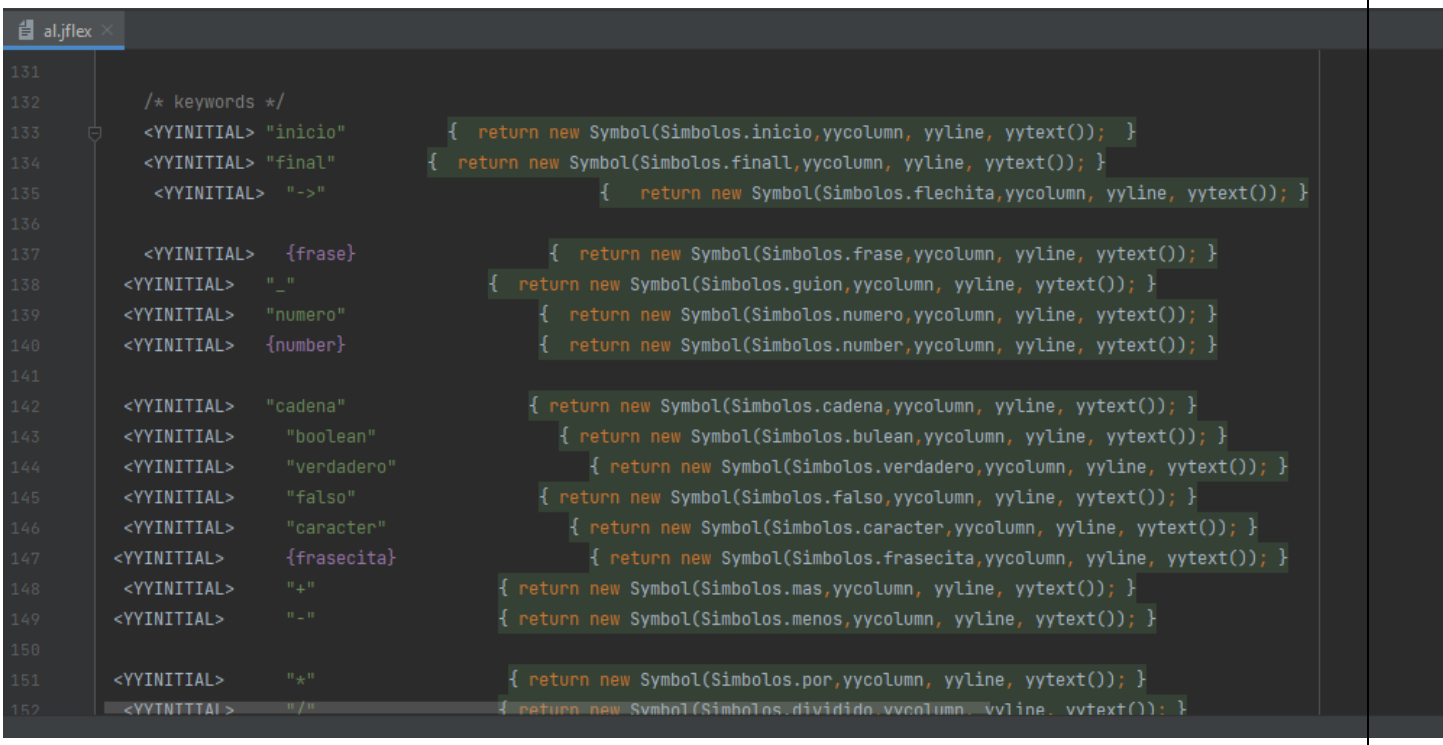
El primera analizador cuenta con un analizador léxico que trabaja con la herramienta Jflex, y con un analizadr sintáctico que se conecta con la primer herramienta, en esta se utiliza la herramienta Cup.

A estas herramientas se le ingresa un archivo de extensión .jflex y .cup que contienen las gramáticas especificadas para el reconocimiento del lenguaje, obteniendo com output tres clases java que se conectan entre si, generadas automáticamente por las herramientas.

Todos los analizadores de todos los paquetes funcionan igual.



El arhivo jflex contiene todas las especificaiones del análisis léxico para identificación de tokens, estas luego serann utilizadas como terminales en el analizador sintáctico.



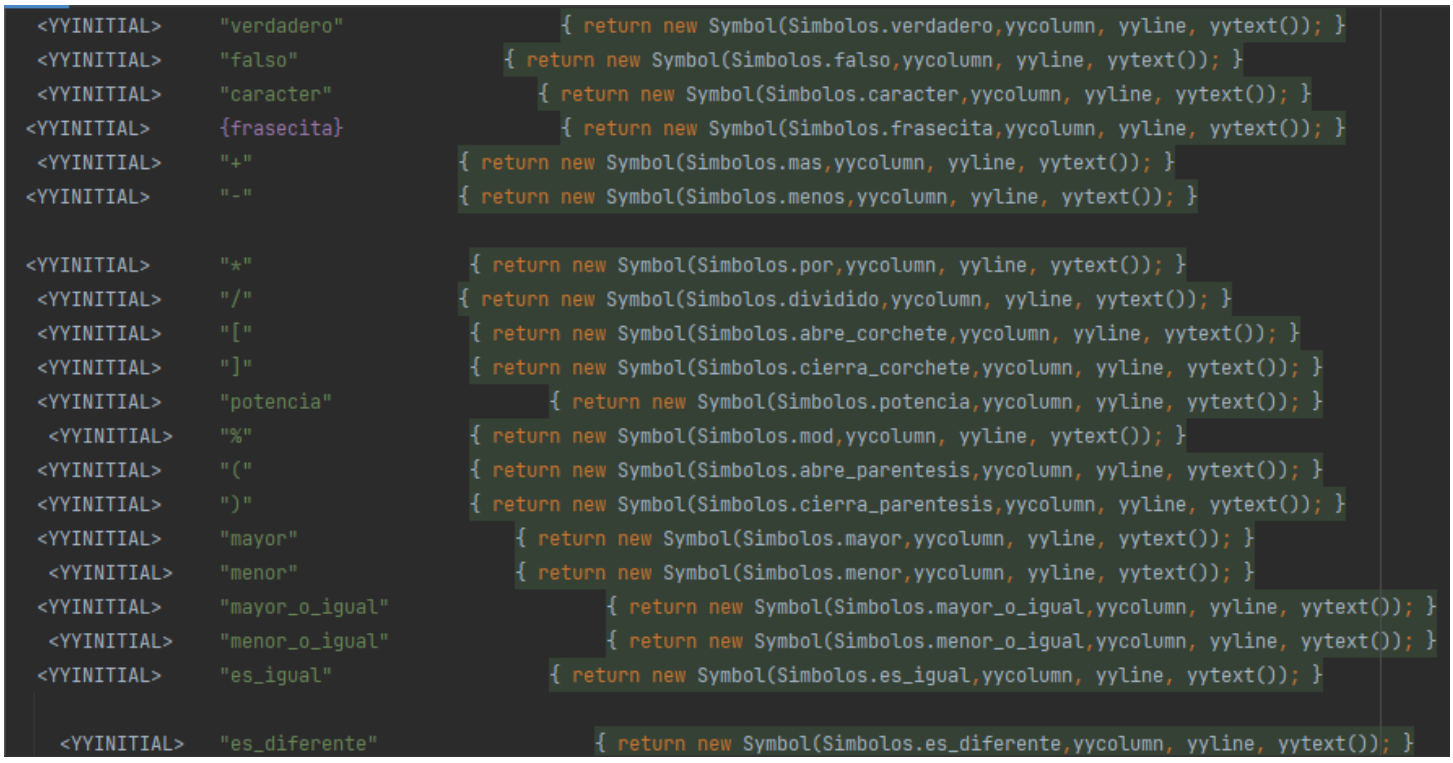
```
terminal menos  ;
terminal por ;
terminal dividido ;
terminal abre_corchete  ;
terminal cierra_corchete   ;
terminal potencia    ;
terminal mod    ;
terminal abre_parentesis ;
terminal cierra_parentesis;
terminal mayor;
terminal menor;
terminal mayor_o_igual ;
terminal menor_o_igual ;
terminal es_igual ;
terminal es_diferente ;
terminal or     ;
terminal and ;
terminal not ;
terminal ingresar  ;
terminal como;
terminal con_valor;
terminal punto_y_coma;
```

Luego se especifican los no terminales, estos son productos de la especificación sintáctica de los terminales, especificando asi y recursivamente la gramatica general de el pseudocodigo que se pide.

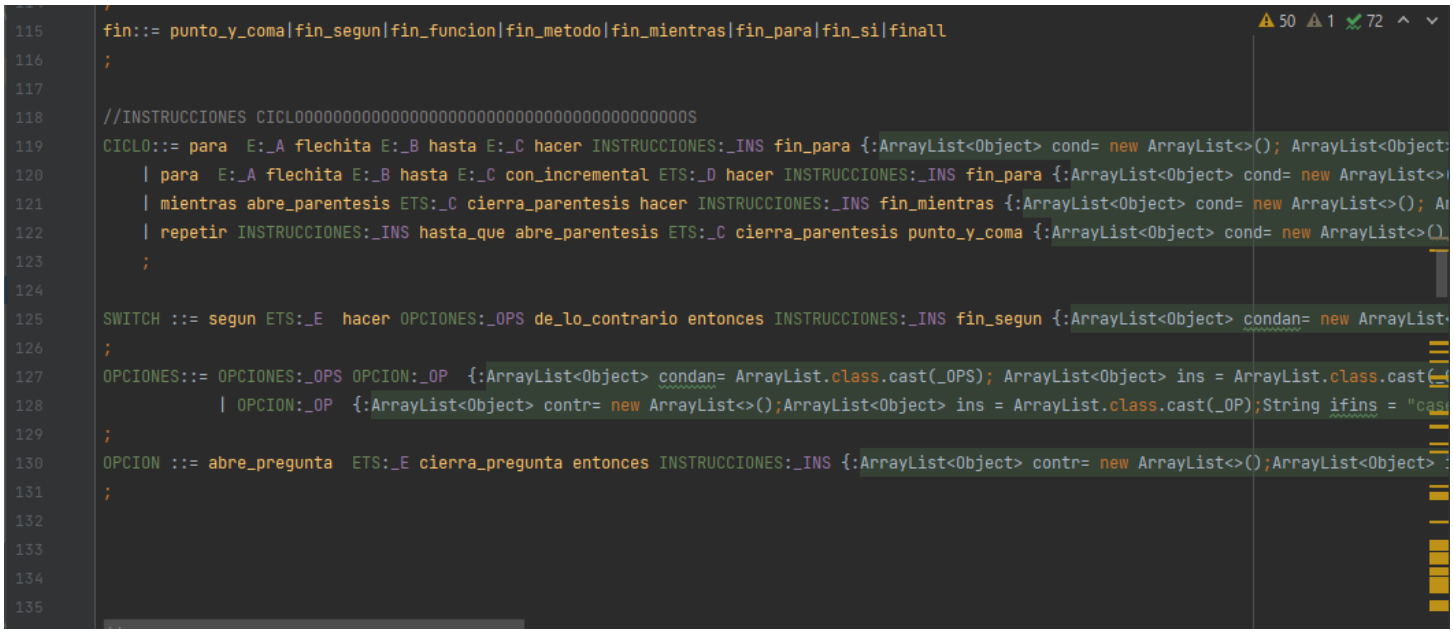
```
    | repetir INSTRUCCIONES:_INS hasta_que abre_parentesis ETS:_C cierra_parentesis punto_y_coma {:ArrayList<Object> co
    ;

SWITCH ::= segun ETS:_E  hacer OPCIONES:_OPS de_lo_contrario entonces INSTRUCCIONES:_INS fin_segun {:ArrayList<Object> condan= new Arr
;
OPCIONES::= OPCIONES:_OPS OPCION:_OP  {:ArrayList<Object> condan= ArrayList.class.cast(_OPS); ArrayList<Object> ins = ArrayList.class.
          | OPCION:_OP  {:ArrayList<Object> contr= new ArrayList<>();ArrayList<Object> ins = ArrayList.class.cast(_OP);String ifins
;
OPCION ::= abre_pregunta  ETS:_E cierra_pregunta entonces INSTRUCCIONES:_INS {:ArrayList<Object> contr= new ArrayList<>();ArrayList<Ob
;



//CONDICION IF
CONDICIONIF::= si abre_parentesis ETS:_C cierra_parentesis entonces INSTRUCCIONES:_INS fin_si {:ArrayList<Object> cond= new ArrayList<
             | si abre_parentesis ETS:_C cierra_parentesis entonces INSTRUCCIONES:_INS IFANIDADOS:_INFANIDADOS fin_si {:ArrayList<O
;
IFANIDADOS ::= IFANIDADOS:_INFANIDADOS  o_si abre_parentesis ETS:_C cierra_parentesis entonces INSTRUCCIONES:_INS {:ArrayList<Object>
             | IFANIDADOS:_INFANIDADOS  de_lo_contrario INSTRUCCIONES:_INS {:ArrayList<Object> condan= new ArrayList<>(); ArrayList
             |  o_si abre_parentesis ETS:_C cierra_parentesis entonces INSTRUCCIONES:_INS {:ArrayList<Object> conelse= new ArrayLis
             | de_lo_contrario INSTRUCCIONES:_INS {:ArrayList<Object> contr= new ArrayList<>(); ArrayList<Object> ins = ArrayList.c
```

Este en especifico genera el parser hacia python.
El objetivo del mismo es generar un AST, una estructura que permite el cambio de información, o ordenamiento de información prioritaria, reconoce desde los terminales, el nivel mas bajo de el árbol, hasta los no terminales mas complejos, y como salida se obtiene una gramatica con sentido, en este caso el código traducido a código destino.

Todos los analizadores funcionan de la misma manera, ya que reconocen la misma gramatica, solamente cambia la función del AST producido.



```
<YYINITIAL>     "verdadero"                 { return new Symbol(Simbolos.verdadero,yycolumn, yyline, yytext()); }
<YYINITIAL>     "falso"                 { return new Symbol(Simbolos.falso,yycolumn, yyline, yytext()); }
<YYINITIAL>     "caracter"                  { return new Symbol(Simbolos.caracter,yycolumn, yyline, yytext()); }
<YYINITIAL>     {frasecita}                 { return new Symbol(Simbolos.frasecita,yycolumn, yyline, yytext()); }
<YYINITIAL>     "+"                 { return new Symbol(Simbolos.mas,yycolumn, yyline, yytext()); }
<YYINITIAL>     "-"                 { return new Symbol(Simbolos.menos,yycolumn, yyline, yytext()); }

<YYINITIAL>     "*"                  { return new Symbol(Simbolos.por,yycolumn, yyline, yytext()); }
<YYINITIAL>     "/"                 { return new Symbol(Simbolos.dividido,yycolumn, yyline, yytext()); }
<YYINITIAL>     "["                  { return new Symbol(Simbolos.abre_corchete,yycolumn, yyline, yytext()); }
<YYINITIAL>     "]"                  { return new Symbol(Simbolos.cierra_corchete,yycolumn, yyline, yytext()); }
<YYINITIAL>     "potencia"                  { return new Symbol(Simbolos.potencia,yycolumn, yyline, yytext()); }
<YYINITIAL>     "%"                 { return new Symbol(Simbolos.mod,yycolumn, yyline, yytext()); }
<YYINITIAL>     "("                 { return new Symbol(Simbolos.abre_parentesis,yycolumn, yyline, yytext()); }
<YYINITIAL>     ")"                 { return new Symbol(Simbolos.cierra_parentesis,yycolumn, yyline, yytext()); }
<YYINITIAL>     "mayor"                 { return new Symbol(Simbolos.mayor,yycolumn, yyline, yytext()); }
<YYINITIAL>     "menor"                 { return new Symbol(Simbolos.menor,yycolumn, yyline, yytext()); }
<YYINITIAL>     "mayor_o_igual"                 { return new Symbol(Simbolos.mayor_o_igual,yycolumn, yyline, yytext()); }
<YYINITIAL>     "menor_o_igual"                 { return new Symbol(Simbolos.menor_o_igual,yycolumn, yyline, yytext()); }
<YYINITIAL>     "es_igual"                  { return new Symbol(Simbolos.es_igual,yycolumn, yyline, yytext()); }

<YYINITIAL>     "es_diferente"                  { return new Symbol(Simbolos.es_diferente,yycolumn, yyline, yytext()); }
```

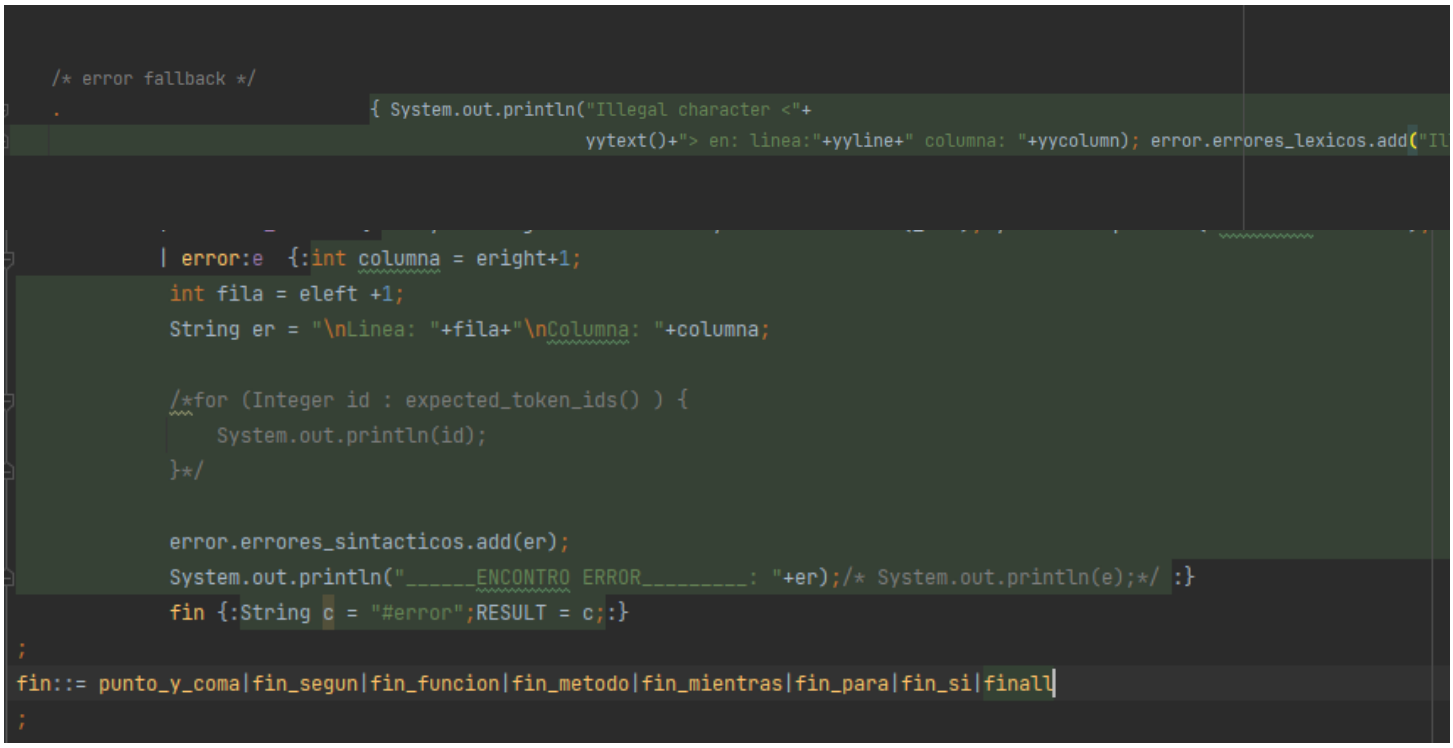


```
115  fin::= punto_y_coma|fin_segun|fin_funcion|fin_metodo|fin_mientras|fin_para|fin_si|finall
116  ;
117
118  //INSTRUCCIONES CICLOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOS
119  CICLO::= para  E:_A flechita E:_B hasta E:_C hacer INSTRUCCIONES:_INS fin_para {:ArrayList<Object> cond= new ArrayList<>(); ArrayList<Object
120      | para  E:_A flechita E:_B hasta E:_C con_incremental ETS:_D hacer INSTRUCCIONES:_INS fin_para {:ArrayList<Object> cond= new ArrayList<>(
121      | mientras abre_parentesis ETS:_C cierra_parentesis hacer INSTRUCCIONES:_INS fin_mientras {:ArrayList<Object> cond= new ArrayList<>(); A
122      | repetir INSTRUCCIONES:_INS hasta_que abre_parentesis ETS:_C cierra_parentesis punto_y_coma {:ArrayList<Object> cond= new ArrayList<>()
123      ;
124
125  SWITCH ::= segun ETS:_E  hacer OPCIONES:_OPS de_lo_contrario entonces INSTRUCCIONES:_INS fin_segun {:ArrayList<Object> condan= new ArrayList
126  ;
127  OPCIONES::= OPCIONES:_OPS OPCION:_OP  {:ArrayList<Object> condan= ArrayList.class.cast(_OPS); ArrayList<Object> ins = ArrayList.class.cast(
128            | OPCION:_OP  {:ArrayList<Object> contr= new ArrayList<>();ArrayList<Object> ins = ArrayList.class.cast(_OP);String ifins = "cas
129  ;
130  OPCION ::= abre_pregunta  ETS:_E cierra_pregunta entonces INSTRUCCIONES:_INS {:ArrayList<Object> contr= new ArrayList<>();ArrayList<Object>
131  ;
132
133
134
135
```

```
 <YYINITIAL>     "caracter"                { return new Symbol(Simbolos.caracter,yycolumn, yyline, yytext()); }
<YYINITIAL>      {frasecita}               { return new Symbol(Simbolos.frasecita,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "+"                 { return new Symbol(Simbolos.mas,yycolumn, yyline, yytext()); }
<YYINITIAL>      "-"                 { return new Symbol(Simbolos.menos,yycolumn, yyline, yytext()); }

<YYINITIAL>      "*"                  { return new Symbol(Simbolos.por,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "/"                 { return new Symbol(Simbolos.dividido,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "["                  { return new Symbol(Simbolos.abre_corchete,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "]"                  { return new Symbol(Simbolos.cierra_corchete,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "potencia"                { return new Symbol(Simbolos.potencia,yycolumn, yyline, yytext()); }
  <YYINITIAL>    "%"                 { return new Symbol(Simbolos.mod,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "("                 { return new Symbol(Simbolos.abre_parentesis,yycolumn, yyline, yytext()); }
 <YYINITIAL>     ")"                 { return new Symbol(Simbolos.cierra_parentesis,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "mayor"                 { return new Symbol(Simbolos.mayor,yycolumn, yyline, yytext()); }
  <YYINITIAL>    "menor"                 { return new Symbol(Simbolos.menor,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "mayor_o_igual"                 { return new Symbol(Simbolos.mayor_o_igual,yycolumn, yyline, yytext()); }
  <YYINITIAL>    "menor_o_igual"                 { return new Symbol(Simbolos.menor_o_igual,yycolumn, yyline, yytext()); }
 <YYINITIAL>     "es_igual"                { return new Symbol(Simbolos.es_igual,yycolumn, yyline, yytext()); }

   <YYINITIAL>   "es_diferente"                  { return new Symbol(Simbolos.es_diferente,yycolumn, yyline, yytext()); }
   <YYINITIAL>   "or"                { return new Symbol(Simbolos.or,yycolumn, yyline, yytext()); }
   <YYINITIAL>   "and"                { return new Symbol(Simbolos.and,yycolumn, yyline, yytext()); }
```

```
                                                                                                                    ⚠ 3  ✓ 207
DECLARACION ::= ingresar IDS:_IDS  como  TIPODATO_DECLARACION:_TD con_valor ETS:_E punto_y_coma{:listaenlazada p = new listaenlazada(); p.
            ;

ETS ::= COMPARACIONES:_E {:listaenlazada p = new listaenlazada(); p.agrega(new nodo("COMPARACIONES")); p.agrega(new nodo(listaenlazada.cla
        | E:_E {:listaenlazada p = new listaenlazada(); p.agrega(new nodo("E")); p.agrega(new nodo(listaenlazada.class.cast(_E))); RESULT =
        | INSTRUCCION:_E{:listaenlazada p = new listaenlazada(); p.agrega(new nodo("BLOQUE_INSTRUCCCION")); p.agrega(new nodo(listaenlazad
;
IDS ::= IDS:_IDS coma E:_E {:listaenlazada p = listaenlazada.class.cast(_IDS); p.agrega(new nodo("COMA"));p.agrega(new nodo("E")); p.agreg
    | E:_E {:listaenlazada p =new listaenlazada();p.agrega(new nodo("E")); p.agrega(new nodo(listaenlazada.class.cast(_E))); RESULT = p;RE
    ;
VARIABLE ::= variable:_variable
;
COMPARACIONES::= not abre_parentesis COMPARACIONES:_E cierra_parentesis {:listaenlazada p = new listaenlazada(); p.agrega(new nodo("NOT"))
           |  COMPARACIONES:_E and COMP:_F {:listaenlazada p = listaenlazada.class.cast(_E); p.agrega(new nodo("AND")); p.concatena(lista
           |  COMPARACIONES:_E or COMP:_F {:listaenlazada p = listaenlazada.class.cast(_E); p.agrega(new nodo("OR")); p.concatena(listaen
           |   COMP:_E {:listaenlazada p = listaenlazada.class.cast(_E); RESULT = p;:}
;
COMP::=  E:_E menor E:_F {:listaenlazada p = listaenlazada.class.cast(_E); p.agrega(new nodo("MENOR")); p.concatena(listaenlazada.class.ca
    |  E:_E mayor_o_igual E:_F {:listaenlazada p = listaenlazada.class.cast(_E); p.agrega(new nodo("MAYOR_O_IGUAL")); p.concatena(listaenl
    |  E:_E menor_o_igual E:_F {:listaenlazada p = listaenlazada.class.cast(_E); p.agrega(new nodo("MENOR_O_IGUAL")); p.concatena(listaenl
```

Los errores son validado de tal forma, que si el compilador se encuentra un error léxico o sintáctico, continue hasta encontrar una entrada valida ya sea lexica o sintácticamente. Al detectar uno de estos los agrega a una lista general de errores, un atributo estatico de una clase errores, se realizó estatica para el fácil acceso a esta desde cualquier objeto o clase.



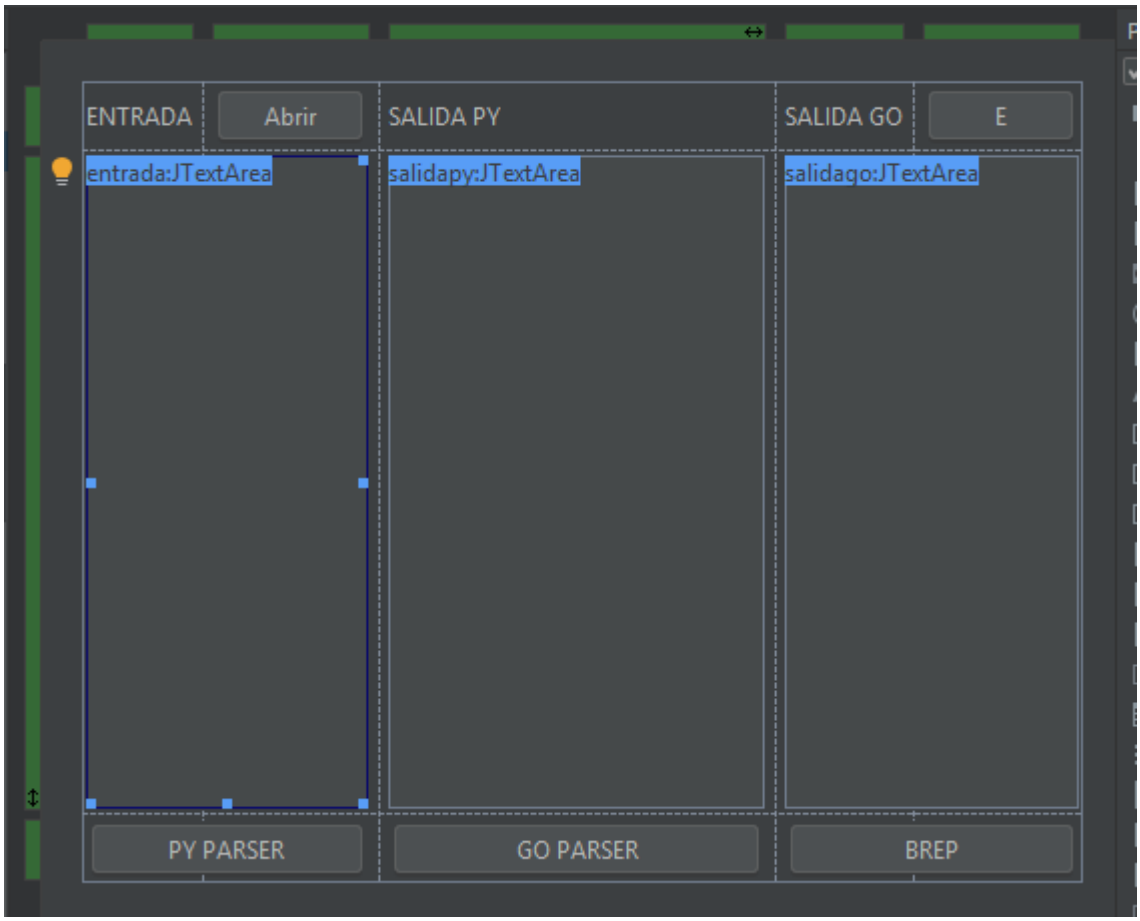
El motor del programa se encuentra en esta función, a la cual le entra como parámetro un arreglo de instrucciones, producto del Ast producido por la herramienta, Esta función se encarga de la separación de entornos y en el caso de python, de que la indentacion sea la adecuada.

```java
public void imprime(ArrayList p, int indent)
{

    for(Object k: p )
    {
        if (k.getClass()== ArrayList.class)
            imprime(ArrayList.class.cast(k), indent: indent+1);
        else
        {
            String inde = "";
            for(int i = 0;i<indent;i++)
                inde+="      ";
            //System.out.println(inde+k.toString());
            resultado += "\n"+inde+k.toString();
        }
    }
}
```

Esta función se encarga de la realización del código del árbol sintáctico grafico, tanto la función anterior como esta son recursivas, lo que nos permite trabajar con este tipo de estructuras de una manera más fácil y segura.

```
public void conecta(nodo x, nodo c)
{
    if ((x.value == c.value && x.Next != null) || (grado.get(x.value)!= null && grado.get(c.value)!=null &&grado.get(x.value)==grado.get

    if (apuntadores.get(x.value) != null) {
        if (apuntadores.get(x.value).contains(c.value)) {
            if (!flechitas.contains("\n"+x.hashCode() +"->"+c.hashCode()+";")) {
                flechitas.add("\n"+x.hashCode() +"->"+c.hashCode()+";");
                conexiones+= "\n"+x.hashCode() +"->"+c.hashCode()+";";
                //System.out.println("\n" + x.hashCode() + "->" + c.hashCode() + ";");
            }

            if (apuntadores.get(c.value) == null)
            {
                general.remove(c);
                if (x.Next!=null) {
                    c = x.Next;
                    if (x.value == "BLOQUE_INSTRUCCION") {
                        nodo temp = x.Prev;
                        general.remove(x);
                        x = temp;
                    }
                    conecta(x, c);
                }
            }
            else if (c.Next != null)
                conecta(c,c.Next);

        }
        if (c.Next != null)
```


La estructura graficada se trabajo como una lista enlazada, para un manejo más fácil del árbol de salida, y una organizacion mejor de la gramatica producida.



Por ultimo se tiene el diseño de la forma y El listener de cada botón, donde se instancian todos los objetos de las clases antes mencionadas, logrando asi la funcionalidad total del proyecto.

L