COMPILADORES 1

PROYECTO #1

MANUAL TÉCNICO

EDUARDO JOSUÉ GONZÁLEZ CIFUENTES

201900647

INDICE

INTRODUCCION

Este manual técnico presenta un proyecto innovador que aborda el desafío de la comunicación efectiva entre lenguajes de programación, específicamente la traducción de StatPy a Python. Utilizando Java CUP y Java JFlex para análisis léxico y sintáctico, este proyecto en Java ofrece una interfaz gráfica intuitiva. El manual guía a desarrolladores y profesionales de la ciencia de datos en la instalación, configuración y uso de la herramienta, incluyendo detalles técnicos sobre las técnicas de traducción y consejos prácticos. Además también se analizaron archivos json para obtener información necesaria para el proceso generación de graficas.

Este proyecto representa un avance significativo en la interoperabilidad entre lenguajes de programación y tiene el potencial de acelerar el desarrollo de proyectos en ciencia de datos y programación. La combinación de Java CUP y Java JFlex para análisis léxico y sintáctico brinda una precisión excepcional en la traducción de StatPy a Python. La interfaz gráfica intuitiva simplifica el proceso, lo que lo convierte en una herramienta valiosa para quienes buscan simplificar la tarea de traducir código, aprovechando la eficiencia y versatilidad de ambos lenguajes.

OBJETIVOS

1. Facilitar la comprensión de las técnicas de análisis léxico y sintáctico en el contexto de la traducción de código entre lenguajes de programación.
2. Promover la adopción efectiva del pseudocódigo como una herramienta de programación y traducción entre StatPy y Python.
3. Fomentar la colaboración y el aprendizaje colaborativo en el campo de la programación y la ciencia de datos

ESPECIFICACIÓN TÉCNICA

REUERIMIENTOS HARDWARE

- Laptop o computadora de escritorio.
- Mínimo 4GB de Memoria RAM.
- 500GB de Disco Duro o Superior.
- Procesador Intel Core i3 o superior.

REQUERIMIENTOS SOFTWARE

- Sistema Operativo Windows 10 o superior.
- Java Runtime Enviroment (JRE) versión 8.2 o superior.
- Java Development Kit (JDK) version 8.2 o superior.
- Lenguaje de Programación Java.
- NetBeans IDE 8.2 o superior.
- Navegador web.
- Librería Java Cup.
- Librería Java Flex.
- Librería rsyntaxtextarea.
- Librería JfreeChart.
- Librería JTatto.
- Librería Quaqua.

LÓGICA DEL PROGRAMA

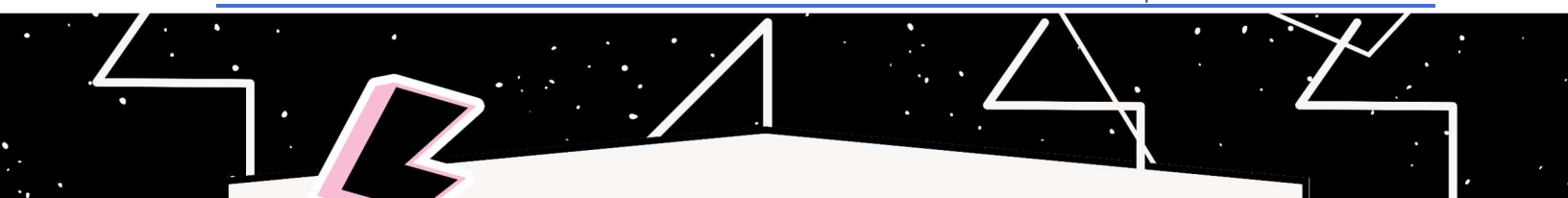
LENGUAJE STATPY

Si necesita información de como se estructura el lenguaje statpy sugiero consulta el manual de usuario donde se encuentra abordado ese tema.

ANÁLISIS LÉXICO STATPY

Acá está un listado de todos los tokens que se utilizaron para el análisis léxico del lenguaje statpy.

TOKEN	LEXEMA	DESCRIPCIÓN
"TkSUMA"	+	Palabra R. Operador Aritmético
"TkMULTIPLICACION"	*	Palabra R. Operador Aritmético
"TkRESTA"	-	Palabra R. Operador Aritmético
"TkDIVISION"	/	Palabra R. Operador Aritmético
"TkIGUAL"	=	Palabra R. Operador Aritmético
"TkPUNTOYCOMA"	;	Palabra R. Operador Simbolos
"TkDOSPUNTOS"	:	Palabra R. Operador Simbolos
"TkPUNTO"	.	Palabra R. Operador Simbolos
"TkCOMA"	,	Palabra R. Operador Simbolos
"TkLLAVEA"	{	Palabra R. Operador Simbolos
"TkLLAVEC"	}	Palabra R. Operador Simbolos
"TkPARENTESISAbre"	(	Palabra R. Operador Simbolos
"TkPARENTESISCierra"	)	Palabra R. Operador Simbolos
"TkCORCHETEAbre"	[	Palabra R. Operador Simbolos
"TkCORCHETECierra"	]	Palabra R. Operador Simbolos
"TkCorchetesArreglo"	[]	Palabra R. Operador Simbolos
"TkMAYORQUE"	>	Palabra R. Operador Relacional
"TkMENORQUE"	<	Palabra R. Operador Relacional
"TkMAYORIGUAL"	>=	Palabra R. Operador Relacional
"TkMENORIGUAL"	<=	Palabra R. Operador Relacional
"TkIGUALIGUAL"	==	Palabra R. Operador Relacional



"TkSignoDolar"	$	
"TkIDISTINTO"	!=	Palabra R. Operador Relacional
"TkAND"	&&	Palabra R. Operador Logico
"TkOR"	||	Palabra R. Operador Logico
"TkNOT"	!	Palabra R. Operador Logico
"TkIncremento"	++	Palabra R. Operador
"TkINT_R"	int	Palabra Reservada Tipo Dato
"TkDOUBLE_R"	double	Palabra Reservada Tipo Dato
"TkCHAR_R"	char	Palabra Reservada Tipo Dato
"TkBOOL_R"	bool	Palabra Reservada Tipo Dato
"TkSTRING_R"	string	Palabra Reservada Tipo Dato
"TkVOID_R"	void	Palabra Reservada Void
"TkMAIN_R"	main	Palabra Reservada Main
"TkIF_R"	if	Palabra Reservada If
"TkELSE_R"	else	Palabra Reservada Else
"TkCASE_R"	case	Palabra Reservada Case
"TkBRAKE_R"	break	Palabra Reservada Break
"TkDEFAULT_R"	default	Palabra Reservada Default
"TkCONSOLE_R"	console	Palabra Reservada Console
"TkWRITE_R"	write	Palabra Reservada Write
"TkFOR_R"	for	Palabra Reservada For
"TkDO_R"	do	Palabra Reservada Do
"TkWHILE_R"	while	Palabra Reservada While
"TkDEFINIR_GLOBALES_R"	DefinirGlobales	Palabra Reservada Definirglobales
"TkGRAFICA_BARRAS_R"	GraficaBarras	Palabra Reservada Graficabarras
"TkGRAFICA_PIE_R"	GraficaPie	Palabra Reservada Graficapie
"TkNewValor"	NewValor	Palabra Reservada Newvalor
"TkEjeX"	Ejex	Palabra Reservada Ejex
"TkTituloX"	TituloX	Palabra Reservada Titulox
"TkTituloY"	TituloY	Palabra Reservada Tituloy
"TkTitulo"	Titulo	Palabra Reservada Titulo
"TkValores"	Valores	Palabra Reservada Valores
"TkTrue"	true	Palabra Reservada True
"TkFalse"	false	Palabra Reservada False
"TkSWITCH_R"	switch	Palabra Reservada Switch

Acá está un listado de todas las expresiones regulares que se utilizaron para el análisis léxico del lenguaje statpy.

TOKEN	EJEMPLO	EXPRESIÓN REGULAR
"TkENTERO"	52	[0-9]+
"TkDECIMAL"	20	([0-9]+\.[0-9]+|[0-9]+)
"TkCADENA"	"hola mundo"	[\"][^\"\n]*[\"]
"TkErChar"	'c'	\'[a-zA-ZñÑ]\'
"TkIdentificador"	var1	[a-z][a-z0-9_]*

ANÁLISIS SINTÁCTICO STATPY

El análisis se llevó a cabo por medio de la herramienta Jcup donde hay una gramática llena de producción la cual usted puede ver en el archivo Gramatia.pdf con notacón BNF además acá abajo se adejunta una foto de la gramática en la herramienta Jcup.

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

import java_cup.runtime.*;
import clases.Error;
import func.Funcion;
import traductionPy.IndentPy;
import java.util.LinkedList;


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

parser code
{:
    public static String resultado = "";
    public static int contadorCases =0;

    public void syntax error(Symbol s)
    {
            Erroor errorsito = new Erroor("Sintactico", s.value.toString() ,"No es valido en este
lenguaje sintact", s.right, s.left);
            Funcion.ErrorList.add(errorsito);
            System.err.println("Error Sintactico: "+ s.value.toString() + " - Fila: " + s.right + " -
Columna: " + s.left + ". Recuperado" );
    }

    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception
    {
            Erroor errorsito = new Erroor("Sintactico", s.value.toString() ,"No es valido en esle
lenguaje sintact", s.right, s.left);
            Funcion.ErrorList.add(errorsito);
            System.err.println("Error Sintactico: "+ s.value.toString() + "   Fila: " + s.right + "
Columna: " + s.left + ". Sin recuperacion." );
    }

:}


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


//------> Declaración de terminales
terminal String TkENTERO, TkDECIMAL ,TkCADENA,TkErChar;
terminal String TkSUMA,R_MOSTRAR,TkMULTIPLICACION,TkRESTA,TkDIVISION,TkIGUAL;
terminal String TkPUNTOYCOMA,TkDOSPUNTOS,TkPUNTO,TkCOMA;
terminal String
TkLLAVEA,TkLLAVEC,TkPARENTESISAbre,TkPARENTESISCierra,TkCORCHETEAbre,TkCORCHETECierra,TkCorchetesArregl
o;
terminal String TkMAYORQUE, TkMENORQUE, TkMAYORIGUAL,
TkMENORIGUAL,TkIGUALIGUAL,TkSignoDolar,TkIDISTINTO;
terminal String TkAND,TkOR, TkNOT,TkIncremento;
terminal String TkINT_R,TkDOUBLE_R,TkCHAR_R,TkBOOL_R,TkSTRING_R;
terminal String TkVOID_R,TkMAIN_R;
terminal String TkIF_R, TkELSE_R, TkCASE_R,TkBRAKE_R,TkDEFAULT_R;
terminal String TkCONSOLE_R,TkWRITE_R;
terminal String TkFOR_R, TkDO_R,TkWHILE_R;
terminal String TkDEFINIR_GLOBALES_R,TkGRAFICA_BARRAS_R,TkGRAFICA_PIE_R, TkNewValor;
terminal String TkEjeX,TkTituloX,TkTituloY,TkTitulo,TkValores;
terminal String TkTrue,TkFalse;
//terminal String TkDimID
terminal String TkSWITCH_R;
terminal String TkIdentificador;

//------> Declaración de no terminales
non terminal inicio;
non terminal instruccionTraduccion;
non terminal instruccionGraficas;
non terminal LinkedList listainstrTraduccion;
non terminal listainstrGraficas;
non terminal expresionTraduccion, expresionGraficas;
non terminal instruccionIf, instruccionConsoleWr, instruccionWhile;
non terminal expresionRelacional;
non terminal instruccionDowhile;
non terminal instruccionFor;
non terminal declaracionvariable;
non terminal tipoDato;
non terminal asignacionvariable;
non terminal instruccionSwitch;
non terminal instruccionListaCasoss;
non terminal instruccionLlaveC;
non terminal instrucListCas;

//------> PRECEDENCIA
precedence left TkOR;
precedence left TkAND;
precedence right TkNOT;
precedence left TkIGUALIGUAL,TkIDISTINTO,TkMENORQUE,TkMENORIGUAL,TkMAYORQUE,TkMAYORIGUAL;
precedence left TkRESTA,TkSUMA;
precedence left TkMULTIPLICACION, TkDIVISION;
precedence right TkRESTA;


//------> Definir Simbolo Inicial
start with inicio;
```

```
// ------------>  Producciones  <------------


inicio::= TkVOID_R TkMAIN_R TkPARENTESISAbre  TkPARENTESISCierra  TkLLAVEA  listainstrTraduccion:lista
TkLLAVEC
                        {:

                        LinkedList<String> listaN = new LinkedList<>();
                        listaN.add("def main( ):");
                        func.Funcion.ContadorIndentacion++;
                        listaN.addAll(func.Funcion.IndentacionPY(lista));
                        func.Funcion.ContadorIndentacion--;
                        listaN.add("if __name__ = \"__main__\":\n\tmain()");
                        func.Funcion.ListaTraduccionPy = listaN;

                        :}
;

listainstrTraduccion ::= listainstrTraduccion:lista  instruccionTraduccion: val
                        {:
                            lista.addAll((LinkedList) val);
                            RESULT = (LinkedList) lista;
                        :}
                      | instruccionTraduccion:val
                        {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.addAll((LinkedList)val);
                            RESULT = (LinkedList) lista;
                        :}
;

instruccionTraduccion ::= instruccionConsoleWr: val {:RESULT = val;:}
                        | instruccionIF: val {:RESULT = val;:}
                        | instruccionWhile: val {:RESULT = val;:}
                        | instruccionDowhile: val {:RESULT = val;:}
                        | declaracionvariable: val {:RESULT = val;:}
                        | asignacionvariable: val {:RESULT = val;:}
                        | instruccionFor : val {:RESULT = val;:}
                        | instrucccionSwitch : val {:RESULT = val;:}
                        | error TkPUNTOYCOMA
;


expresionTraduccion ::= expresionTraduccion: a  TkSUMA  expresionTraduccion:b {:RESULT =
 a.toString()+"+"+b.toString() ;:}
                        |expresionTraduccion: a  TkRESTA  expresionTraduccion:b {: RESULT = a.toString()+"-
"+b.toString();:}
                        |expresionTraduccion: a  TkMULTIPLICACION expresionTraduccion:b {:RESULT =
a.toString()+"*"+b.toString() ;:}
                        |expresionTraduccion: a  TkDIVISION  expresionTraduccion:b {:RESULT =
a.toString()+"/"+b.toString() ;:}
                        |TkRESTA expresionTraduccion:b {:RESULT = "-"+b.toString() ;:}
                        |TkENTERO: val {:RESULT = val;:}
                        |TkDECIMAL: val {:RESULT = val;:}
                        |TkErChar: val {:RESULT = val;:}
                        |TkCADENA: val {:RESULT = val;:}
                        |expresionTraduccion:b  expresionRelacional:a  expresionTraduccion:c  {: RESULT =
b.toString()+a.toString()+c.toString();:}
                        |TkNOT expresionTraduccion:a {:RESULT  = " not "+a.toString();:}
                        |expresionTraduccion:a TkAND expresionTraduccion:b {:RESULT  = a.toString()+" and
"+b.toString();:}
                        |expresionTraduccion:a TkOR expresionTraduccion:b {:RESULT  = a.toString()+" or
"+b.toString();:}
                        |TkTrue {:RESULT = " True ";:}
                        |TkFalse {:RESULT = " False ";:}
                        |TkCADENA: a  TkSUMA  expresionTraduccion:b {:RESULT = a.toString()+" ,
"+b.toString();:}
                        |TkIdentificador : val {:RESULT = val;:}
                        |TkIdentificador : a  TkSUMA expresionTraduccion:b {:RESULT = a.toString()+" +
"+b.toString();:}


;
```

```
expresionRelacional ::= TkMAYORQUE: val {:RESULT = ">";:}
                    | TkMENORQUE: val {:RESULT = "<";:}
                    | TkMAYORIGUAL: val {:RESULT = ">=";:}
                    | TkMENORIGUAL: val {:RESULT = "<=";:}
                    | TkIGUALIGUAL: val {:RESULT = "==";:}
                    | TkIDISTINTO: val {:RESULT = "!=";:}
;


instruccionWhile::= TkWHILE_R TkPARENTESISAbre expresionTraduccion:exp   TkPARENTESISCierra   TkLLAVEA
  listainstrTraduccion:listainst  TkLLAVEC
                    {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.add("while " + exp + " :");
                            func.Funcion.ContadorIndentacion++;
                            lista.addAll(func.Funcion.IndentacionPY(listainst));
                            func.Funcion.ContadorIndentacion--;
                            RESULT = lista;
                    :}
;



instruccionDowhile::=  TkDO_R TkLLAVEA listainstrTraduccion:listainst  TkLLAVEC TkWHILE_R
 TkPARENTESISAbre expresionTraduccion:exp   TkPARENTESISCierra TkPUNTOYCOMA
                    {:
                        LinkedList<String> lista = new LinkedList<>();
                        lista.add("while True:");
                        func.Funcion.ContadorIndentacion++;
                        lista.addAll(func.Funcion.IndentacionPY(listainst));
                        lista.add("if  (" + exp + "):");
                        func.Funcion.ContadorIndentacion++;
                        lista.add("break");
                        func.Funcion.ContadorIndentacion--;
                        func.Funcion.ContadorIndentacion--;
                        RESULT = lista;
                    :}
;


instruccionFor ::= TkFOR_R TkPARENTESISAbre  tipoDato TkIdentificador:a TkIGUAL expresionTraduccion:b
TkPUNTOYCOMA TkIdentificador:c  expresionRelacional:d  expresionTraduccion:e TkPUNTOYCOMA
TkIdentificador:f TkIncremento TkPARENTESISCierra TkLLAVEA listainstrTraduccion:listainstr TkLLAVEC
                    {:

                        // b y e son objetos
                        String nuevad = String.valueOf((Object)d);
                        int nuevab = Integer.parseInt((String) (Object) b);
                        int nuevae = Integer.parseInt((String) (Object) e);
                        LinkedList<String> lista = new LinkedList<>();

                        if ( "<=".equals(nuevad)){
                            lista.add("for " + a+ " in range(" + b + ", " + String.valueOf(nuevae+1) +
"):");

                        }else if ("<".equals(nuevad)){
                            lista.add("for " + a+ " in range(" + b + ", " + e + "):");
                        }else{
                            lista.add("for " + a+ " in range(" + b + ", " + e + "):");
                        }

                        func.Funcion.ContadorIndentacion++;
                        lista.addAll(func.Funcion.IndentacionPY(listainstr));
                        func.Funcion.ContadorIndentacion--;
                        RESULT = lista;
                    :}
;
```

```
instruccionConsoleWr ::= TkCONSOLE_R TkPUNTO TkWRITE_R TkPARENTESISAbre expresionTraduccion:val
TkPARENTESISCierra TkPUNTOYCOMA
                        {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.add("print("+val+")");
                            RESULT = lista;
                        :}
;

//TkElseIf

instruccionIF::=  TkIF_R TkPARENTESISAbre expresionTraduccion:exp TkPARENTESISCierra TkLLAVEA
 listainstrTraduccion:listaInstTrad  instruccionLlaveC:listacierre
                    {:
                        LinkedList<String> lista = new LinkedList<>();
                        lista.add("if " + exp + ":");
                        func.Funcion.ContadorIndentacion++;
                        lista.addAll(func.Funcion.IndentacionPY(listaInstTrad));
                        func.Funcion.ContadorIndentacion--;
                        lista.addAll((LinkedList<String>)listacierre);
                        RESULT = lista;
                    :}

;


instruccionLlaveC::=  TkLLAVEC
                    {:
                        LinkedList<String> lista = new LinkedList<>();
                        lista.add("");
                        RESULT = lista;

                    :}
                    | TkLLAVEC  TkELSE_R TkLLAVEA   listainstrTraduccion:listaInstTrad   TkLLAVEC
                    {:
                        LinkedList<String> lista = new LinkedList<>();
                        lista.add("else:");
                        func.Funcion.ContadorIndentacion++;
                        lista.addAll(func.Funcion.IndentacionPY(listaInstTrad));
                        func.Funcion.ContadorIndentacion--;
                        RESULT = lista;

                    :}
                    | TkLLAVEC  TkELSE_R TkIF_R TkPARENTESISAbre expresionTraduccion:exp
TkPARENTESISCierra  TkLLAVEA  listainstrTraduccion:listaInstTrad3    instruccionLlaveC:listllaveC
                    {:
                        LinkedList<String> lista = new LinkedList<>();
                        lista.add("elif " +exp+" :");
                        func.Funcion.ContadorIndentacion++;
                        lista.addAll(func.Funcion.IndentacionPY(listaInstTrad3));
                        func.Funcion.ContadorIndentacion--;
                        lista.addAll((LinkedList<String>)listllaveC);
                        RESULT = lista;
                    :}
;
```

```
instruccionListaCasoss::=  instruccionListaCasoss:lista2  instrucListCas :val
                            {:
                                LinkedList<String> lista = new LinkedList<>();
                                lista.addAll((LinkedList) lista2);
                                lista.addAll((LinkedList) val);
                                RESULT = (LinkedList) lista;
                            :}
                           | instrucListCas : val
                            {:
                                LinkedList<String> lista = new LinkedList<>();
                                lista.addAll((LinkedList)val);
                                RESULT = (LinkedList) lista;
                            :}
;


instrucListCas::=  TkCASE_R  expresionTraduccion:exp   TkDOSPUNTOS listainstrTraduccion:listainstr

                        {:
                              contadorCases++;
                              LinkedList<String> lista = new LinkedList<>();
                              lista.add(exp + " :") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(listainstr));
                              func.Funcion.ContadorIndentacion--;
                              LinkedList<String> lista2 = new LinkedList<>();
                              lista2.add(",") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(lista2));
                              func.Funcion.ContadorIndentacion--;
                              RESULT = lista;

                        :}
                       | TkCASE_R  expresionTraduccion:exp   TkDOSPUNTOS
listainstrTraduccion:listainstr1 TkBRAKE_R TkPUNTOYCOMA
                        {:
                              contadorCases++;
                              LinkedList<String> lista = new LinkedList<>();
                              lista.add(exp + " :") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(listainstr1));
                              func.Funcion.ContadorIndentacion--;
                              LinkedList<String> lista2 = new LinkedList<>();
                              lista2.add(",") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(lista2));
                              func.Funcion.ContadorIndentacion--;
                              RESULT = lista;

                        :}
                       | TkDEFAULT_R TkDOSPUNTOS listainstrTraduccion:listainstr2
                        {:
                              int conta = Integer.parseInt((String) (Object)contadorCases);
                              LinkedList<String> lista = new LinkedList<>();
                              lista.add(String.valueOf(conta+1) + " :") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(listainstr2));
                              func.Funcion.ContadorIndentacion--;
                              LinkedList<String> lista2 = new LinkedList<>();
                              lista2.add(",") ;
                              func.Funcion.ContadorIndentacion++;
                              lista.addAll(func.Funcion.IndentacionPY(lista2));
                              func.Funcion.ContadorIndentacion--;
                              RESULT = lista;
                        :}
;
```

```
declaracionvariable ::= tipoDato TkIdentificador:b TkPUNTOYCOMA
                        {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.add(b.toString()+" = None");
                            RESULT = lista;
                        :}
                       | tipoDato TkIdentificador:b TkIGUAL expresionTraduccion:a TkPUNTOYCOMA
                        {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.add(b.toString()+" = "+a.toString());
                            RESULT = lista;
                        :}
;


tipoDato::=  TkINT_R: val {:RESULT = val;:}
            |TkDOUBLE_R: val {:RESULT = val;:}
            |TkCHAR_R: val {:RESULT = val;:}
            |TkBOOL_R: val {:RESULT = val;:}
            |TkSTRING_R: val {:RESULT = val;:}
;


asignacionvariable ::= TkIdentificador: a TkIGUAL  expresionTraduccion:b  TkPUNTOYCOMA
                        {:
                            LinkedList<String> lista = new LinkedList<>();
                            lista.add(a.toString() + "=" + b.toString());
                            RESULT = lista;
                        :}
;


instruccionSwitch::= TkSWITCH_R  TkPARENTESISAbre expresionTraduccion:exp  TkPARENTESISCierra TkLLAVEA
 instruccionListaCasoss:listass  TkLLAVEC
                            {:
                                LinkedList<String> lista = new LinkedList<>();
                                lista.add("def switch (" + exp + " ) :");
                                func.Funcion.ContadorIndentacion++;
                                lista.add("switcher = {");
                                func.Funcion.ContadorIndentacion++;
                                lista.addAll(func.Funcion.IndentacionPY((LinkedList<String>)listass));
                                lista.add("}");
                                func.Funcion.ContadorIndentacion--;
                                func.Funcion.ContadorIndentacion--;
                                RESULT = lista;
                            :}


;
```

ANÁLISIS LÉXICO JSON

Acá está un listado de todos los tokens que se utilizaron para el análisis léxico de los archivos Json.

TOKEN	LEXEMA	DESCRIPCIÓN
"TkJsLLAVEABRE"	{	Palabra R. Operador Simbolos
"TkJsLLAVECIERRA"	}	Palabra R. Operador Simbolos
"TkJsDOSPUNTOS"	:	Palabra R. Operador Simbolos
"TkJsCOMA"	,	Palabra R. Operador Simbolos
"TkJsDECIMAL"	25.0	[0-9]+
"TkJsCADENA"	"hola mundo"	[\"][^\"\n]*[\"]

ANÁLISIS SINTÁCTICO JSON

```
package json;

import java_cup.runtime.*;
import clases.Errror;
import func.Funcion;
import clases.Simbolito;
import GUI.*;
import java.util.LinkedList;

parser code
{:
    public static String resultado = "";
    public static String Tipo ="";

    public void syntax_error(Symbol s)
    {
            Errror errorsito = new Errror("Sintactico", s.value.toString() ,"No es valido en  lenguaje
json sintact", s.right, s.left);
            Funcion.ErrorList.add(errorsito);
            System.err.println("Error Sintactico: "+ s.value + " - Fila: " + s.right + " - Columna: " +
s.left + ". Recuperado" );

    }

    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception
    {
            Errror errorsito = new Errror("Sintactico", s.value.toString() ,"No es valido en  lenguaje
json sintact", s.right, s.left);
            Funcion.ErrorList.add(errorsito);
            System.err.println("Error Sintactico: "+ s.value + " - Fila: " + s.right + " - Columna: " +
s.left + ". Sin recuperacion." );
    }
:}

action code
{: :}

terminal String TkJsLLAVEABRE, TkJsLLAVECIERRA;
terminal String TkJsDOSPUNTOS, TkJsCOMA;
terminal String TkJsDECIMAL,TkJsCADENA,TkJsCOMENTMULTI,TkJsCOMENTUNALINEA;

non terminal inicio;
non terminal instruccion;
non terminal lista_instr;
non terminal estructurajson;
non terminal lineaJson;
non terminal valorJsonLinea;
non terminal contenidoJson;

start with inicio;
```

```
// ------------>  Producciones  <------------

inicio ::= lista_instr
;

lista_instr ::= lista_instr instruccion
    | instruccion
;

instruccion ::= estructurajson
              | TkJsCOMENTMULTI
              | TkJsCOMENTUNALINEA
              | error  TkJsLLAVECIERRA
;


estructurajson ::= TkJsLLAVEABRE  contenidoJson    TkJsLLAVECIERRA
;


contenidoJson ::=  lineaJson
                |  lineaJson  TkJsCOMA contenidoJson
;

lineaJson ::= TkJsCADENA:a  TkJsDOSPUNTOS valorJsonLinea:b {:
                                                        String KeyNameJson =
GUI.NombreActualArchivoJson;

 Funcion.HashMapFileJson.computeIfAbsent(KeyNameJson, k -> new LinkedList<>()).add(new Simbolito(a,
b.toString(), Tipo));
                                                        RESULT =
Funcion.HashMapFileJson.get(KeyNameJson);
                                                    :}
;

valorJsonLinea ::=  TkJsCADENA: val {:  Tipo = "Cadena"; RESULT = val; :}
                |   TkJsDECIMAL: val {: Tipo = "Double"; RESULT = val; :}
;
```

El análisis se llevó a cabo por medio de la herramienta Jcup donde hay una gramática llena de producción la cual usted puede ver en el archivo GramatiaJson.pdf con notacón BNF además acá se adejunta una foto de la gramática en la herramienta Jcup.

TRADUCCIÓN

La traducción se realizó desde el archivo Parser.Cup ya que con la experiencia se vio que era una manera más fácil manejar desde ahí la traducción.

```
public static LinkedList<String> IndentacionPY (LinkedList<String> ListaTPY){
      String indentPy = "";
      for(int i=0; i<ContadorIndentacion;i++){
          indentPy = "\t"+indentPy;
      }

      for(int j=0; j < ListaTPY.size(); j++){
          ListaTPY.set(j, indentPy+ListaTPY.get(j));
      }

      return ListaTPY;

  }
```

MÉTODOS

Análisis Statpy

```
public static void analizadorStatpy(String ruta, String jflexFile, String cupFile) {

        try {

            String opcionesJflex[] = {ruta + jflexFile, "-d", ruta};
            jflex.Main.generate(opcionesJflex);

            String opcionesCup[] = {"-destdir", ruta, "-parser", "Parser", ruta + cupFile};
            java_cup.Main.main(opcionesCup);

        } catch (Exception e) {
            System.out.println("No se ha podido generar los analizadores");
            System.out.println(e);
        }

    }
```

Análisis Json

```
public static void analizadorJson(String ruta, String jflexFile, String cupFile) {
        try {
            String opcionesJflex[] = {ruta + jflexFile, "-d", ruta};
            jflex.Main.generate(opcionesJflex);

            String opcionesCup[] = {"-destdir", ruta, "-parser", "Parser", ruta + cupFile};
            java_cup.Main.main(opcionesCup);

        } catch (Exception e) {
            System.out.println("No se ha podido generar los analizadores");
            System.out.println(e);
        }

    }
```

Se utilizaron Packets y Clases de Java para llevar a cabo la realización de la traducción. A continuación, se dejará las clases mas importantes para la funcionalidad del proyecto.

Clase Main

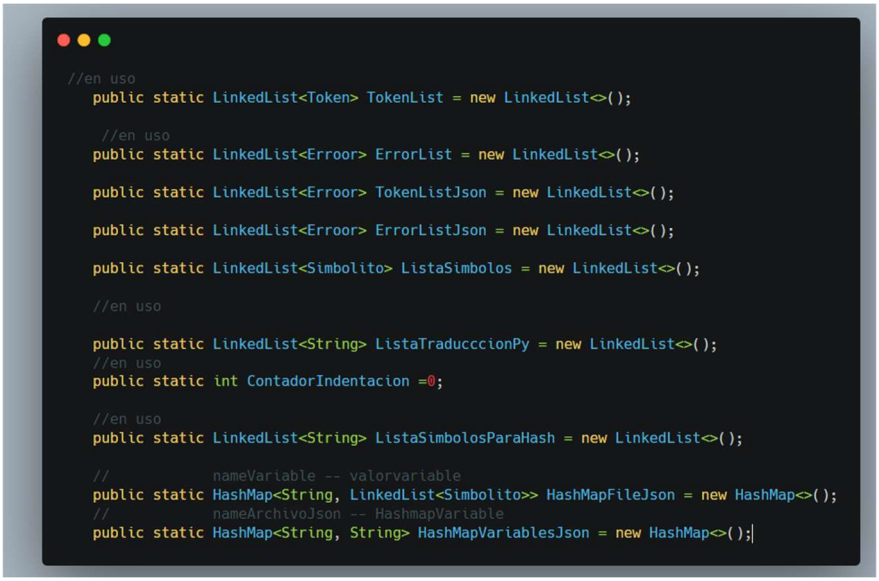
Acá se manda a llamar la interfaz Grafica

```
public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(new McWinLookAndFeel());
        } catch (UnsupportedLookAndFeelException e) {
            System.err.println("No se pudo aplicar el Look and Feel deseado: " + e.getMessage());
            e.printStackTrace();
        }


        GUI nuevaGUI = new GUI();
        nuevaGUI.setVisible(true);

    }
```

Clase Grafica Barras

```java
public class GraficaBarras {

    public static void GBarras(
                String Titulo,
                String TituloX,
                String TituloY,
                double valores[],
                String ejex []

    ) {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        for(int i= 0; i<5;i++){
                    dataset.addValue(valores[i], "Valor", ejex [i]);

        }

        JFreeChart charxd = ChartFactory.createBarChart3D(Titulo, TituloX, TituloY, dataset);
        // Personaliza los colores de las barras
        CategoryPlot plot = (CategoryPlot) charxd.getPlot();
        BarRenderer renderer = (BarRenderer) plot.getRenderer();

        // Establece el color de las barras individuales
        renderer.setSeriesPaint(0, Color.GREEN);   // Color de la primera serie

        ChartFrame frames = new ChartFrame("Ejemplo", charxd);
        frames.pack();
        frames.setVisible(true);

    }
}
```

Variables Globales



```java
//en uso
    public static LinkedList<Token> TokenList = new LinkedList<>();

     //en uso
    public static LinkedList<Erroor> ErrorList = new LinkedList<>();

    public static LinkedList<Erroor> TokenListJson = new LinkedList<>();

    public static LinkedList<Erroor> ErrorListJson = new LinkedList<>();

    public static LinkedList<Simbolito> ListaSimbolos = new LinkedList<>();

    //en uso

    public static LinkedList<String> ListaTraducccionPy = new LinkedList<>();
    //en uso
    public static int ContadorIndentacion =0;

    //en uso
    public static LinkedList<String> ListaSimbolosParaHash = new LinkedList<>();

    //            nameVariable -- valorvariable
    public static HashMap<String, LinkedList<Simbolito>> HashMapFileJson = new HashMap<>();
    //            nameArchivoJson -- HashmapVariable
    public static HashMap<String, String> HashMapVariablesJson = new HashMap<>();
```

CRÉDITOS

Proyecto desarrollado por Eduardo González del Curso Organización de Lenguajes y Compiladores 1, en ciudad de Guatemala para la Universidad San Carlos de Guatemala el 20 de septiembre del año 2022.

Se adjunta repositorio donde se encuentra todo el código fuente.

https://github.com/Lal0gg/OLC1_Proyecto1_201900647