- **Facultad de Ingeniería**
- **Escuela de Ciencias y Sistemas**
- **ORGANIZACION DE LENGUAJES Y COMPILADORES 1, 2do. Semestre 2023.**

Manual tecnico

PROYECTO 1
ORGANIZACION DE LENGUAJES Y COMPILADORES 1

GENESIS NAHOMI APARICIO ACAN

carne :20211329

Archivo jflex encargado del análisis sintáctico: Este archivo es el encargado de reconocer cada token que nuestro lenguaje analiza, en este, primero se definió aquellos tokens que necesiten alguna expresión regular para poder identificarlos, se hizo uno tanto para los archivos JSon como para stat

```
// ------> Expresiones Regulares


CADENA = \" ([^\"] | "\\\"")+ \"
entero = [0-9]+
DECIMAL = [0-9]+("."[  |0-9]+)?
ID = [a-zA-Z_][a-zA-Z0-9_]*
//----> COMENTARIOS
COMENT_S = "//" .*
COMENT_M = "/*"([^*]|("*"+[^*/]))*"*/"
CHARI=\'
```

Luego de esto se definieron que siguen un determinado patrón se definen como reconocibles mientras tiempo, creamos nuevos símbolos adjuntos para que podamos hacer esto más adelante en nuestro análisis , se hizo uno tanto para los archivos JSon como para stat

```
//---SIMBOLOS
"("       {  funciones.info.ListaTokensStat.add(new funciones.TokensStat(yytext(),"PARENTESIS_A",yyline,yycolumn));
             return new Symbol(sym.PARENTESIS_A, yycolumn, yyline, yytext());}

")"       {  funciones.info.ListaTokensStat.add(new funciones.TokensStat( yytext(),"PARENTESIS_C",yyline,yycolumn));
             return new Symbol(sym.PARENTESIS_C, yycolumn, yyline, yytext());}

"{"       {funciones.info.ListaTokensStat.add(new funciones.TokensStat(yytext(),"LLAVE_A",yyline,yycolumn));
             return new Symbol(sym.LLAVE_A, yycolumn, yyline, yytext());}
```

- **Universidad de San Carlos de Guatemala**
- **Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería**
- **Lenguajes Formales y de Programación, 2er. Semestre 2022.**

 Archivo cup encargado del análisis sintáctico: Determinación de los símbolos terminales y no terminales con su tipo.

```
//------> Declaración de terminales
terminal String ENTERO,PARENTESIS_A,PARENTESIS_C,LLAVE_A,LLAVE_C,PUNTO_COMA,DOSP,COMA,CORCHETE_A,
CORCHETE_C,DOLAR,IGUAL,MAS,MENOS,POR,DIV,MAYOR,MENOR,M_IGUAL,ENOR_IGUAL,
IGUALIGUAL,DIFERENTE,AND,OR,NOT,VOID,INT,DOUBLE,CHAR,BOOL,STRING,MAIN,IF,MASMAS,MENOSMENOS,
ELSE,SWITCH,BREAK,FOR,WHILE,CONSOL,DEFAULT,CASE,DO,VAR,DECIMAL,TRUE,FALSE,CHARI,GLOBALES,
ID,CADENA,NUEVOVALOR,BARRAS,PIE,TITULOY,TITULOX,EJEX,TITULO,VALORES;
```

y se empieza con la producción de la inicial llamada codigo encargada de leer la estructura del archivo, cuando la reconozcamos,

```
inicio ::=  VOID MAIN  PARENTESIS_A PARENTESIS_C LLAVE_A lista_Inst:a  LLAVE_C
 {:funciones.Traduccion.TraduccionPy.add("def main (): \n" +a.toString() + "if__name__ = \"__main__\": \n main()" +
        ;

lista_Inst::= instruccion:a lista_Inst:b      {:RESULT = a +""+ b ; :}
            |instruccion:a                        {:RESULT=a;:}
        ;
instruccion::= declaracionVar:a       {:RESULT=a+"\n";:}
            |ifif:a                   {:RESULT=a+"\n";:}
            |imprimir:a               {:RESULT=a+"\n";:}
            |unwhile:a                {:RESULT=a+"\n";:}
```

info

Esta es la clase encargada de contener las diferentes array para poder contener los datos para tanto la generacion de el reporte de tokens como de errores

```
import java.util.LinkedList;

/**
 *
 * @author genes
 */
public class info {
      public static LinkedList<TokensStat> ListaTokensStat = new LinkedList<TokensStat>();
      public static LinkedList<TokensJson> listaTokensJSon = new LinkedList<TokensJson>();
      public static LinkedList<ErroresStat> ListaErroresStat = new LinkedList<ErroresStat>();
      public static LinkedList<ErrorJson> ListaErroresJSon = new LinkedList<ErrorJson>();
}
```

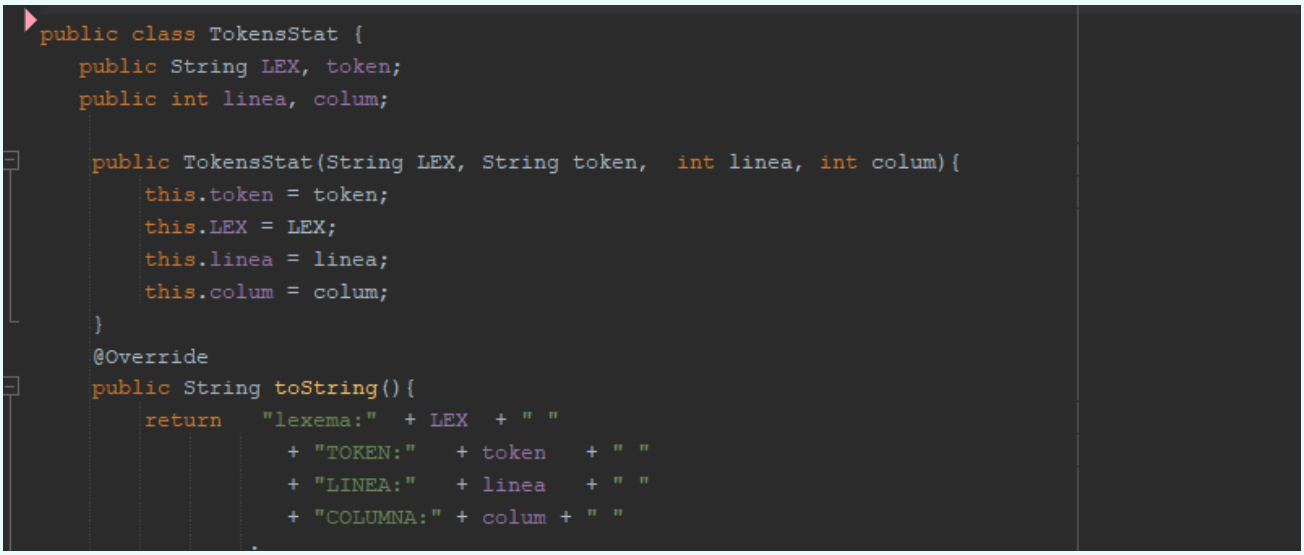
Maps

esta clase contiene en su interior los diferentes hasmap que contienen los datos para la generacion de las diferentes graficas que el programa genera

```
package funciones;
import java.util.HashMap;
import java.util.LinkedList;
import javax.swing.JOptionPane;


/**
 *
 * @author genes
 */
public class maps {
   public static  HashMap<String, Object> Globales_tabla = new HashMap<>();

  public static HashMap<String, Object> variables_Json = new HashMap<>();

  public static HashMap<String, Object> Archivos_Json = new HashMap<>();
    //--> funciones.funcion.ListaT.add(new Tokens(yytext() ,"punto_coma" ,yyline ,yycolumn)); PARA LA TABLA DE TOKENS
```

TokensStat -tokensJSon

esta clase se guardan los objetos que forman parte de los tokens , contienen la fila , la columna , el token y el lexema correspondiente, se hizo tanto para el archivo JSon como para el Stat



```
public class TokensStat {
    public String LEX, token;
    public int linea, colum;

    public TokensStat(String LEX, String token,  int linea, int colum){
        this.token = token;
        this.LEX = LEX;
        this.linea = linea;
        this.colum = colum;
    }
    @Override
    public String toString(){
        return   "lexema:"  + LEX  + " "
                 + "TOKEN:"   + token   + " "
                 + "LINEA:"   + linea   + " "
                 + "COLUMNA:" + colum + " "
```

TablaErroresJson-TablaErroresStat

En esta clase setoman los diferentes datos de la lista que se tiene para los errores y se coloca codigo html para generar una tabla con estos , la estructura es la misma tanto para los errores como para los reportes de tokes

```
package funciones;
import java.io.FileWriter;
import java.io.IOException;

import java.util.LinkedList;
import javax.swing.JOptionPane;

public class TablaTokensJSonHTML {
 public static  int a=0;
    public static void generateHTMLFromTokensListJS (LinkedList<TokensJson> tokensListJson) {
        String htmlContent = "<html>\n" +
                             "<head><title>Lista de Tokens</title></head>\n" +
                             "<body>\n" +
                             "<div style=\"text-align:center;\">\n"+
                             "<table border=\"1\" style=\"margin: 0 auto;\">\n" +
                             "<tr align=\"center\"><th>LEXEMA</th>\n" +
                             "<th align=\"center\">TOKEN</th>\n" +
                             "<th align=\"center\">LINEA</th>\n" +
```

GraficaPie

Esta clase toma los datos de los hasmap para la grafica de pie y genera esta grafica usando la lbreria Freechard mostrando la grafica luego en un png

```
public class GraficaPie {
        public void Graficar(String titulo,ArrayList<String> ValuesX,ArrayList<Double> Values) throws IOExceptio
          System.out.println(x: "grafica de pie");
            DefaultPieDataset dataset = new DefaultPieDataset( );
                  for (int i = 0; i < ValuesX.size(); i++) {
                      dataset.setValue(key: ValuesX.get(index: i), value: Values.get(index: i) );
                  }
                  JFreeChart chart = ChartFactory.createPieChart(
                  title: titulo,   // chart title
                   dataset,          // data
                   legend: true,             // include legend
                   tooltips:true,
                   urls: false);

                int width = 640;   /* Width of the image */
                int height = 480;  /* Height of the image */
                File pieChart = new File( pathname:"GRAFICA_PIE .jpeg" );
                ChartUtilities.saveChartAsJPEG( file: pieChart , chart , width , height );
        }
```

GraficaBarras

Esta clase toma los datos de los hasmap para la grafica de barras y genera esta grafica usando la lbreria Freechard mostrando la grafica luego en un png

```
public class GraficarBarras {
       public void Graficar(String titulo,String titulox,String tituloy,ArrayList<String> ValuesX,ArrayList<Double
            System.out.println(x: titulo);
           System.out.println(x: titulo);
           System.out.println(x: titulox);
           System.out.println(x: tituloy);
           for (String valor : ValuesX) {
               System.out.println(x: valor);
           }
           for (double val : Values) {
               System.out.println(x: val);
           }
            final DefaultCategoryDataset dataset = new DefaultCategoryDataset( );

                   for (int i = 0; i < ValuesX.size(); i++) {
                       dataset.addValue( value: Values.get(index: i) , rowKey: ValuesX.get(index: i) , columnKey: Value

                   }
                   JFreeChart barChart = ChartFactory.createBarChart(
                       title: titulo,
```

interfaz

la clase interfas se hizo con ayuda de drag and drop de neatbeans , este cuenta con varios botones los cuales ayudan a que el sistema reciba ordenes y genere los diferentes reportes, modifique, guarde y abra archivos .json y sp

```
     */
    public static void main(String args[]) {
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz1().setVisible(b: true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton Abrir;
    private javax.swing.JComboBox<String> Analizadoresbotn;
    private javax.swing.JButton Ejecutar;
    private javax.swing.JTextArea Entrada;
    private javax.swing.JButton Reporte;
    private javax.swing.JTextArea Salida;
    private javax.swing.JButton guardar;
    private javax.swing.JButton guardarCo;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPanel;
    private javax.swing.JScrollPane jScrollPane2;
    // End of variables declaration
```