MANUAL TECNICO

Cristian Daniel Gomez Escobar
COMPILADORES 1

INTRODUCCION

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

OBJETIVOS

- Entender el funcionamiento de un compilador en sus dos primeras fases.
- Entender los conceptos de análisis.
- Realizar el uso correcto de herramientas de análisis.
- Comprender los tipos de análisis sintáctico.

EXPRESIONES REGULARES

```
PR_CONJ="CONJ"
LR_DOS_PUNTOS=":"
LR_COMA=","
LR_PUNTO="."
LR_PUNTO_COMA=";"
LR_LLAVE_IZQ="{"
LR_LLAVE_DER="}"
LR_DISYUNCION="|"
LR_CERRADURA_KLEENE="*"
LR_CERRADURA_POSITIVA="+"
LR_CERRADURA_INTERROGACION="?"
PORCENTAJE="%%"
SEPARADOR="~"

FINLINEA=\r|\n|\r\n
ESPACIOS = [ \r\n]+
CARACTER_ENTRADA = [^\r\n]
COMENTARIO_M="<!" [^/]~ "!>"
COMENTARIO_L="//" {CARACTER_ENTRADA}* {FINLINEA}?
LETRA_MINUSCULA=[a-z]
LETRA_MAYUSCULA=[A-Z]
NUMERO=[0-9]
FLECHA="-"{ESPACIOS}*">"

ENTR=[^\r\n\"]
SL=[\\']
SLL=[\\n]
SLLL=[\\]
SLLLL=[\"]
CARACTER_ESPECIAL=[ -/:-@\[-`{-}]
ID=[a-zA-Z][a-zA-Z0-9]+

SR=\"({ENTR}|{ESPACIOS}|{SL}|{SLLL}{SLLLL}|{SLL})+\"
SRR=({SL}|{SLLL}{SLLLL}|{SLL})+
```

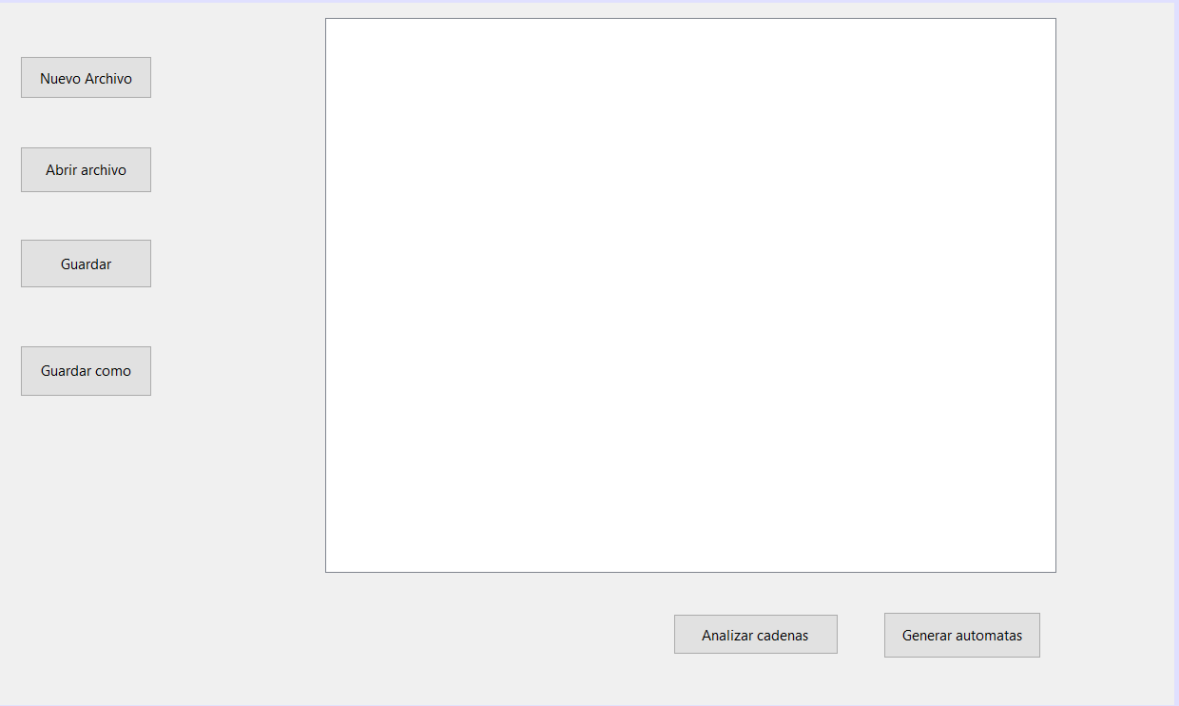
DESCRIPCION DEL SISTEMA

El presente sistema en el cual el usuario interactúa de forma directa con la interfaz grafica, permite la lectura del código en el área de texto, para mostrar sus respectivos análisis mostrados en graficas.

LENGUAJES UTILIZADOS

Se utilizo el lenguaje java para la programación y diseño del programa, también se utilizo Graphviz para la creación de las graficas

INTERFAZ GRAFICA



GENERAR AUTOMATAS

En esta parte se ha hecho un método del árbol donde será el principal código que generará los demás autómatas, los cuales son el AFD, tabla de transiciones y tabla de siguientes, todas derivan del árbol. Este recibe su token inicial y la señal de ir almacenando los tokens y sus hijos

```
arbol.add(new Token((Nodo) a, String b));
:}
|error LR_PUNTO_COMA;
```

Posteriormente conforme se va recorriendo la expresión regular este ira almacenando

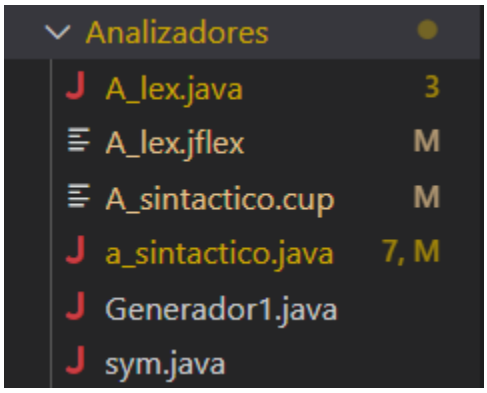
```
ER::= LR_PUNTO:a ER:b ER:c {:{System.out.println("ER: " + a + " " + b + " " + c);};
Nodo padre = new Nodo(a);
padre.setHijoIzq((Nodo)b);
padre.setHijoDer((Nodo)c);
RESULT=padre;
:}
    | LR_DISYUNCION:a ER:b ER:c {:{System.out.println("ER: " + a + "ER: "+ b +"ER: " +c);};
    Nodo padre = new Nodo(a);
    padre.setHijoIzq((Nodo)b);
    padre.setHijoDer((Nodo)c);
    RESULT=padre;
    :}
    | LR_CERRADURA_KLEENE:a ER:b {:{System.out.println("ER: " + a + " " + b);};
    Nodo padre = new Nodo(a);
    padre.setHijoIzq((Nodo)b);
    RESULT=padre;
    :}
    | LR_CERRADURA_POSITIVA:a ER:b {:{System.out.println("ER: " + a + " " + b);};
    Nodo padre = new Nodo(a);
    padre.setHijoIzq((Nodo)b);
    RESULT=padre;
    :}
    | LR_CERRADURA_INTERROGACION:a ER:b {:{System.out.println("ER: " + a + " " + b);};
    Nodo padre = new Nodo(a);
    padre.setHijoIzq((Nodo)b);
    RESULT=padre;
    :}
    | LR_LLAVE_IZQ ID:a LR_LLAVE_DER {:{System.out.println("ER: " + a);};
    Nodo padre = new Nodo(a);
    padre.setHoja(true);
    RESULT=padre;
    :}
    | SR:a {:{System.out.println("ER_sint: " + a);};
    Nodo padre = new Nodo(a);
    padre.setHoja(true);
    RESULT=padre;
```

Se iran declarando los nodos hojas y sus respectivos demás hijos para poder generar los demás análisis.

Desarrollo de package

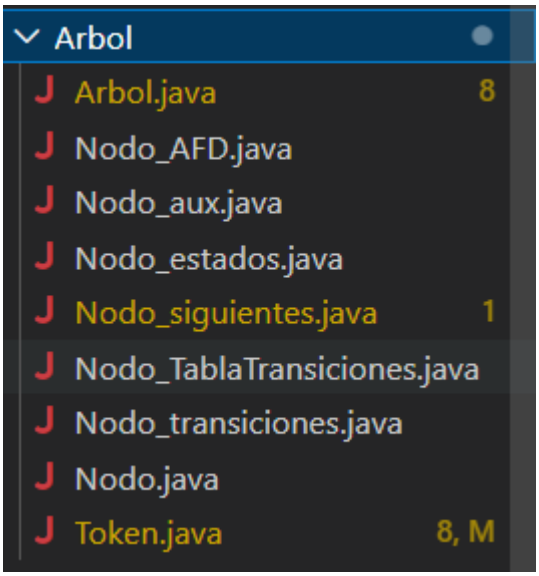
ANALIZADORES

El package analizadores fue realizado el analizador léxico y sintáctico con su respectivo generador



ARBOL

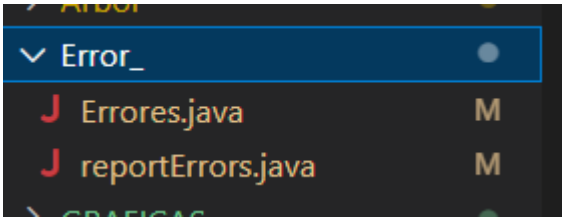
Aquí se creara el árbol, tabla transiciones, tabla siguientes y AFD, en esta se tendrán los aspectos de recursividad para poder llamar a cada uno de los nodos que posee



```
        if (actual.getToken().equals(anObject:"*")){
            actual.setAnulable(anulable:true);

            actual.getPrimero().addAll(actual.getHijoIzq().getPrimero());
            actual.getUltimo().addAll(actual.getHijoIzq().getUltimo());
            listas.add(new Nodo_siguientes(actual.getHijoIzq().getUltimo(),actual.getHijoIzq().getPrimero()));

            // System.out.println("**************************"+actual.getHijoIzq().getToken()+" . "+actual.getHijoIzq().getUltimo()+" .

        }else if (actual.getToken().equals(anObject:"|")){
            actual.setAnulable(actual.getHijoIzq().isAnulable() || actual.getHijoDer().isAnulable());
            actual.getPrimero().addAll(actual.getHijoIzq().getPrimero());
            actual.getPrimero().addAll(actual.getHijoDer().getPrimero());
            actual.getUltimo().addAll(actual.getHijoIzq().getUltimo());
            actual.getUltimo().addAll(actual.getHijoDer().getUltimo());

        }else if (actual.getToken().equals(anObject:".")){
            actual.setAnulable(actual.getHijoIzq().isAnulable() && actual.getHijoDer().isAnulable());
            if (actual.getHijoIzq().isAnulable()){
                actual.getPrimero().addAll(actual.getHijoIzq().getPrimero());
                actual.getPrimero().addAll(actual.getHijoDer().getPrimero());
//------                  complet+="<tr><td bgcolor=\"yellow\"> "+actual.getHijoIzq().getToken()+" </td><td bgcolor=\"#77ff33\">"+actual.getHi
                listas.add(new Nodo_siguientes(actual.getHijoIzq().getUltimo(),actual.getHijoDer().getPrimero()));
            }else{
                actual.getPrimero().addAll(actual.getHijoIzq().getPrimero());
//------                  complet+="<tr><td bgcolor=\"yellow\"> "+actual.getHijoIzq().getToken()+" </td><td bgcolor=\"#77ff33\">"+actual.getHi
                listas.add(new Nodo_siguientes(actual.getHijoIzq().getUltimo(),actual.getHijoDer().getPrimero()));
            }
            if (actual.getHijoDer().isAnulable()){
                actual.getUltimo().addAll(actual.getHijoIzq().getUltimo());
                actual.getUltimo().addAll(actual.getHijoDer().getUltimo());
            //      complet+="<tr><td bgcolor=\"yellow\"> "+actual.getHijoIzq().getToken()+" </td><td bgcolor=\"#77ff33\">"+actual.getHijoIzq(
            }else{
                actual.getUltimo().addAll(actual.getHijoDer().getUltimo());
            //      complet+="<tr><td bgcolor=\"yellow\"> "+actual.getHijoIzq().getToken()+" </td><td bgcolor=\"#77ff33\">"+actual.getHijoIzq(
            }
```

ERROR_

Aquí estará el reporte de errores léxicos y sintácticos, con estos se podrá saber cuando un archivo tenga los errores, en este caso no será capaz de generar ninguno de sus autómatas.




```
public void mostrarReporte() {
    try {
        FileWriter file = new FileWriter(fileName:"C:\\Users\\USER\\Desktop\\P1_OLC1_1S2023\\src\\GRAFICAS\\ERRORES_202107190\\Reporte_erro
        file.write(inicio + Lexico + sintactico + fin);
        file.close();
        //Desktop.getDesktop().open(new File(file_get_path));
        System.out.println(x:"Reporte generado con exito.");
    } catch (Exception e) {
    }
```

GRAFICAS

En esta carpeta estarán las gráficas respectivas que tendrá que generar el programa por cada análisis exitoso que se haga.

PROYECTO1

Aquí estarán las graficas que tendrá el programa, también se cuenta con las funciones que necesitará el análisis de cadenas del programa.