Manual Técnico

Proyecto 1

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1

C

Edgar Rolando Alvarez Rodriguez
202001144

MARZO 2022

INTRODUCCION

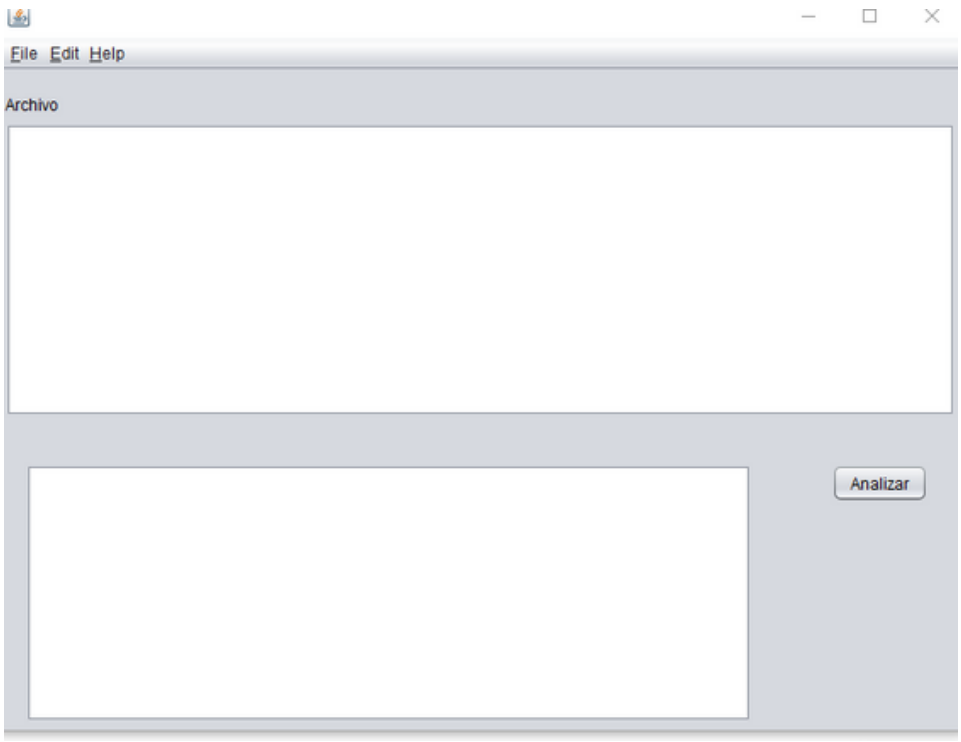
El presente manual abordará el tema del funcionamiento del código de la Proyecto 1, el cual consiste en realizar la lectura de un archivo "olc" así como generar a partir de este diversos gráficos sobre expresiones en notación polaca y verificar textos a partir de dichas expresiones.

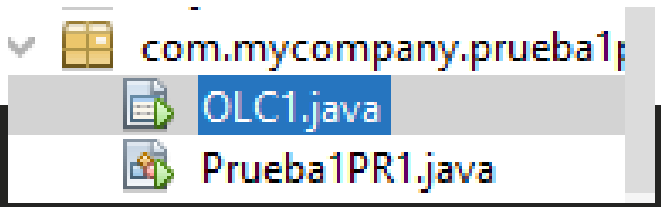
El manual esta dirigido a técnicos y programadores que manejan el soporte del software y puedan darle mantenimiento, añadir funcionalidades al programa o revisar el propósito de los archivos que componen el programa.

FUNCIONAMIENTO DEL SISTEMA

interfaz:

Se genera una interfaz tipo Java.ClassForm usando el lenguaje de programación Java el cual se encarga de abrir los archivos, asi como colocar el texto de este dentro de una caja de texto en la interfaz para luego analizar su contenido.





FUNCIONAMIENTO DEL SISTEMA

interfaz (Código):

Para navegar por el menu principal la programación esta hecha por botones.

```
private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JFileChooser fc = new JFileChooser();

    fc.showOpenDialog( parent: null);
    File archivo = fc.getSelectedFile();
    //System.out.println(archivo.getName());
    jLabel1.setText( text: archivo.getName());

    try{
        FileReader fr = new FileReader( file: archivo);
        BufferedReader br = new BufferedReader( in: fr);
        String texto = "";
        String linea = "";
        while ((linea=br.readLine())!=null){
            texto += linea+"\n";
        }
        jTextArea1.setText( t: texto);
        JOptionPane.showMessageDialog( parentComponent: null,  message: "archivo leido correctamente");
    }catch(Exception e){
```

```
    private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // asdf
        //System.out.println(jLabel1.getText());
        String namearchivo = jLabel1.getText();

        namearchivo = namearchivo.replaceAll( regex: ".olc",  replacement: "");
        System.out.println( x: namearchivo);


        FileWriter fichero = null;
        PrintWriter pw = null;
        try{
            String path = "C:\\Users\\wwwed\\OneDrive\\Escritorio\\"+namearchivo+".olc";
            fichero = new FileWriter( fileName: path);
            pw = new PrintWriter( out: fichero);
            pw.write( s: jTextArea1.getText());
```

CARGA DE DATOS

LECTURA DE ARCHIVOS:

Para ingresar los archivos OLC al sistema en esta ocasión se opto por la opción de abrir un buscador de archivos y cargar directamente el texto en la interfaz.

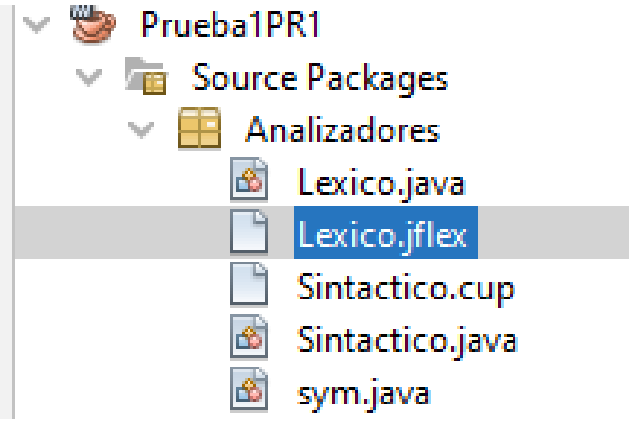
```
//NUEVO
Analizadores.Lexico scanner;
Analizadores.Sintactico parse;
ArrayList<Excepcion> errores = new ArrayList();
```

```
try {

    scanner = new Lexico(new BufferedReader(new StringReader( s: jTextArea1.getText())));
    parse = new Sintactico( s: scanner);
    parse.parse();
    errores.addAll( c: scanner.Errores);
    errores.addAll( c: parse.getErrores());

    generarReporteHTML(errores);

    String result = "";
    String temp = "";
    for (int i = 0; i < parse.listaExpresiones.size(); i++) {
        result += parse.listaExpresiones.get( index: i).id + '\n';
        result += parse.listaExpresiones.get( index: i).expolaca + '\n';
        temp = parse.listaExpresiones.get( index: i).expolaca;
        cua = new Graficas.Clase4( er: temp, contador: Integer.toString(i));
    }

    for (int i = 0; i < parse.resultados.size(); i++) {
        result += parse.resultados.get( index: i) + '\n';
    }
    this.jTextArea2.setText( t: result);
```

Analisis de los Datos

Lexico

Una vez ingresado los datos dentro de la caja de texto, se procede a analizar los datos con el boton de "analizar", primero se ejecuta el archivo léxico que analiza los datos en busca de tokens que no pertenezcan a la gramática.

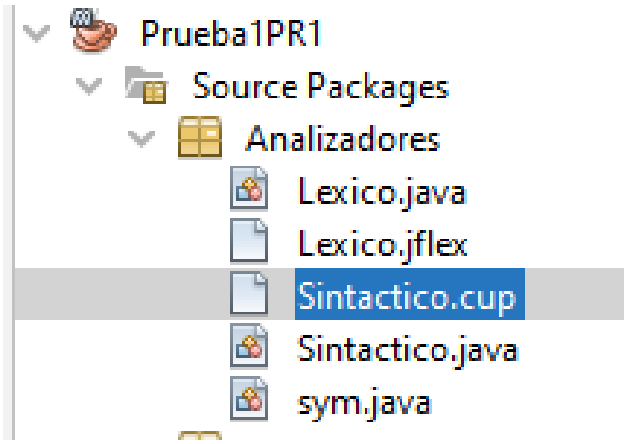



```
%%
/* 3. Reglas Semanticas*/
"CONJ" {  System.out.println("Reconocio PR: "+yytext()); return new Symbol(sym.PR_CONJ,yyline,yychar,y

";" { System.out.println("Reconocio "+yytext()+" punto y coma"); return new Symbol(sym.PTCOMA,yyline,y
//"(" { System.out.println("Reconocio "+yytext()+" parentesis abre"); return new Symbol(sym.PARIZQ,yyl
//")" { System.out.println("Reconocio "+yytext()+" parentesis cierra"); return new Symbol(sym.PARDER,y
//"[" { System.out.println("Reconocio "+yytext()+" corchete abre"); return new Symbol(sym.CORIZQ,yylin
//"]" { System.out.println("Reconocio "+yytext()+" corchete cierra"); return new Symbol(sym.CORDER,yyl
"}" { System.out.println("Reconocio "+yytext()+" llave cierra"); return new Symbol(sym.LLAVDER,yyline,
"{" { System.out.println("Reconocio "+yytext()+" llave abre"); return new Symbol(sym.LLAVIZQ,yyline,yy

//"->" { System.out.println("Reconocio "+yytext()+" flecha"); return new Symbol(sym.FLECH,yyline,yycha
":" { System.out.println("Reconocio "+yytext()+" dos puntos"); return new Symbol(sym.DOSPTS,yyline,yyc
"%%" { System.out.println("Reconocio "+yytext()+" %%"); }

{FLECHA} { System.out.println("Reconocio "+yytext()+" flecha"); return new Symbol(sym.FLECH,yyline,yyc

\n {yychar=1;}

{BLANCOS} {}
//{SEPARADOR} {}
/* comments */
{comentariosimple} {System.out.println("Comentario: "+yytext()); }
{comentariovariaslineas} {System.out.println("Comentario Varias Lineas: "+yytext()); }
```

Analisis de los Datos

Sintáctico

Luego de que el análisis léxico terminase y enviase los datos de los errores al sistema, el programa procede a ejecutar el archivo sintáctico, el cual se encarga de recolectar las variables y expresiones regulares encontradas en el archivo para luego guardarlos en listas para ser analizados.




```
/* 2. Codigo para el parser, variables, metodos */
parser code
{:
    //Clases, objetos, variables, lista, etc... en sintaxis java

    //Creo una lista de tipo String llamada 'resultados', donde guardare cada uno de los resultados an
    public List<String> resultados = new ArrayList<String>();

    //Lista de objetos con el ID y la expresion Regular
    public List<ExpresionRegular> listaExpresiones = new ArrayList<ExpresionRegular>();


    //Lista de erros
    public ArrayList<Excepcion> Errores = new ArrayList();

    /**
     * Método al que se llama automáticamente ante algún error sintactico.
     **/
    public void syntax_error(Symbol s){
        Errores.add(new Excepcion("Sintáctico", "Error de sintaxis detectado. Se detectó: " + s.value,
    }
```

Errores

Los errores se manejan por medio un objeto que recibe los parámetros del tipo de error, ya sea léxico o sintáctico y la fila y columna donde esta el error, para luego guardar este mismo en una lista y plasmarla en un archivo "html" que se abre automáticamente luego de finalizar todo el análisis.

```java
package Errores;

public class Excepcion {
    public String tipo;
    public String descripcion;
    public String linea;
    public String columna;

    public Excepcion(String tipo, String descripcion, String linea, String columna) {
        this.tipo = tipo;
        this.descripcion = descripcion;
        this.linea = linea;
        this.columna = columna;
    }

    @Override
    public String toString(){
        return this.tipo + ": " + this.descripcion + " en la linea " + this.linea + " y columna " +
    }
}
```

```java
for (Excepcion err : errores) {
    pw.println( x: "<tr>");
     pw.println("<td>" + err.tipo + "</td>");
    pw.println("<td>" + err.descripcion + "</td>");
    pw.println("<td>" + err.linea + "</td>");
    pw.println("<td>" + err.columna + "</td>");
    pw.println( x: "</tr>");
}
```

GENERAR LOS GRAFICOS

Para generar los gráficos de cada una de las expresiones regulares encontradas se utilizo Graphviz que es una librería que nos permite generar gráficos por medio de su lenguaje.

Se generan 3 graficas por medio de Graphviz.

1. La grafica del Árbol Sintáctico se genera en el archivo llamado "Arbol" en la carpeta "Graficas"
2. La grafica del AFD se genera en por medio del archivo llamado "transitontable" el cual se encarga de generar el código que luego es ingresado en un archivo txt para mandarlo a Graphviz.
3. La grafica del AFND se genera en por medio del archivo llamado "transAFND" el cual se encarga de generar el código que luego es ingresado en un archivo txt para mandarlo a Graphviz.