Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Auxiliar: Maynor Pilo Tuy

Organización de Lenguajes y Compiladores 1

Proyecto 1 Exregan

Lesther Kevin Federico López Miculax

Carnet: 202110897

23/03/2023

DATOS DE DESARROLLADOR

Nombre del desarrollador: Lesther Kevin Federico López Miculax

Correo electrónico: lestherlopez64@gmail.com

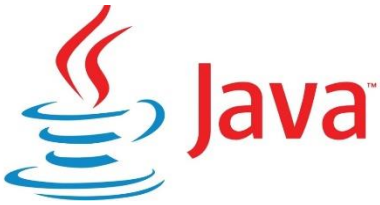
Nacionalidad: guatemalteca

Carné universitario: 202110897

DETALLES DE DESARROLLO

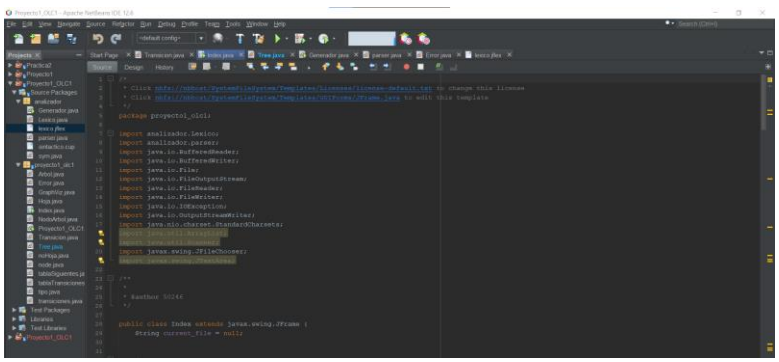
LENGUAJE DE PROGRAMACIÓN

Para la realización de este proyecto, Exregan, se utilizó principalmente un lenguaje de programación, el cual es java, lenguaje que pertenece a la sección de programación de alto nivel que por lo general se utiliza para el desarrollo de softwares, aplicaciones móviles, juegos, entre otros. Un dato histórico de este lenguaje es que fue desarrollado por Sun Microsystems en la década de los 90 y posteriormente adquirido por Oracle Corporation. Por otro lado, otras características de Java es que es un lenguaje de programación robusto, seguro y portátil, de este modo, este lenguaje se convierte en uno de los lenguajes de programación más populares del mundo y se utiliza ampliamente en la industria del desarrollo o la programación.



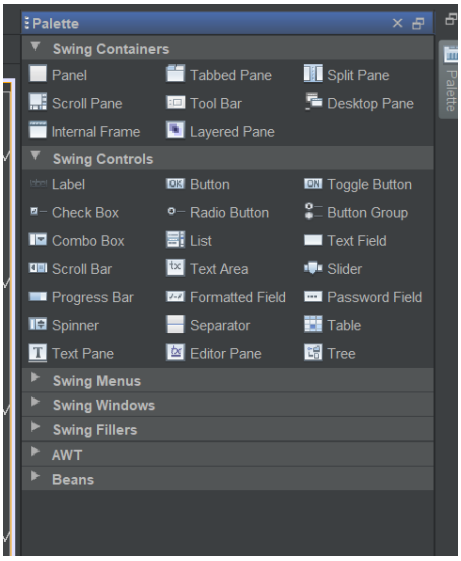
APACHE NETBEANS

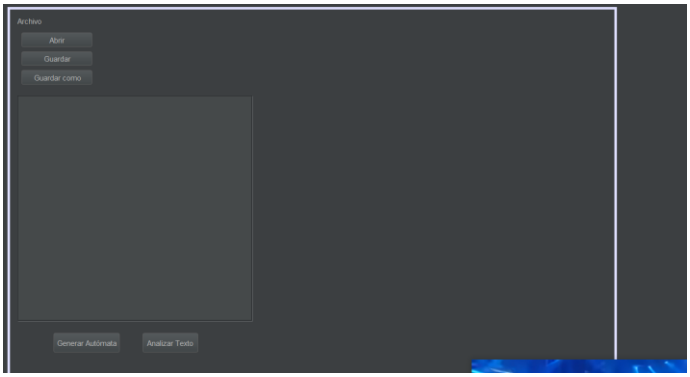
Para poder desarrollar el lenguaje de programación utilizado se utilizó un IDE, dicho de otro modo, un editor de texto o editor de desarrollo, el cual fue el denominado Apache Netbeans. El IDE utilizado es una herramienta que ofrece una inmensa diversidad de características y funcionalidades para tener un mejor desarrollo del lenguaje a utilizar, como la edición de código con resaltado de sintaxis, finalización automática de código, creación de ejecutable,depuración, pruebas y perfiles de código, herramientas de diseño de interfaz gráfica de usuario como dRag and drop la cual se profundizara a continuación, integración con sistemas de control de versiones, y soporte para diferentes tipos de frameworks para el desarrollo web.



DRAG AND DROP

Por otro lado, otro de las herramientas utilizados y que resultó de gran ayuda para la creación de este proyecto y logró en gran parte que sea funcional es la denominada drag and drop, la cual es una técnica de interfaz de usuario que permite a los desarrolladores mover elementos de una aplicación o página web de una ubicación a otra mediante el uso del mouse, siendo esto algo demasiado funcional que simplifico la labor de realizar una interfaz gráfica de manera manual, debido a que drag and drop es altamente intuitivo, logrando que las aplicaciones de escritorio se vean agradable para el usuario.

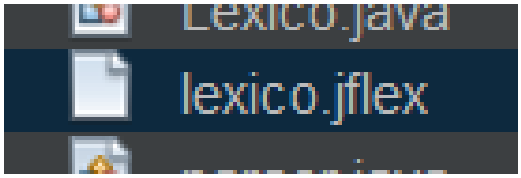




JFLEX

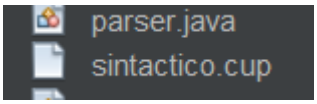
El corazón del proyecto son los analizadores, siendo el responsable del análisis léxico, de este modo, se puede decir que Jflex es una herramienta de generación de analizadores léxicos en Java, por consiguiente, permite definir reglas para reconocer y crear tokens de cadenas de texto según un la estructura de la gramática establecida que se encuentra especificada en el manual de usuario. Por otro lado, también es posible destacar de Jflex que hace uso de

expresiones regulares para definir patrones de texto que representan los diferentes tipos de tokens que se esperan en la entrada, por consiguiente, genera código javo que es encargado de analizar las entradas



CUP

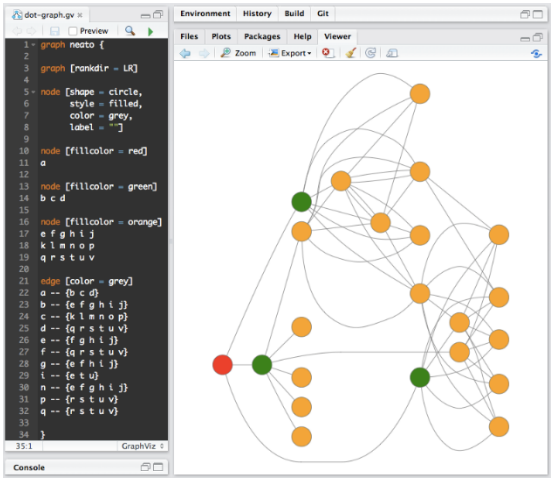
Por otro lado, luego de conocer la herramienta para formar el analizador léxico, es importante también conocer el analizador sintáctico, el cual es formado con la ayuda de la herramienta CUP, conocido también como Construction of Useful Parsers, la cual es útil para la generación de analizadores sintáctico o parsers en Java, de este modo, CUP permite definir la gramática de un lenguaje y generar automáticamente el código Java para un analizador sintáctico que puede ser integrado en una aplicación, logrando que las reglas sintácticas sean comprobadas en las entradas y lograr que el analisis sintactico se de correctamente, evitando errores.

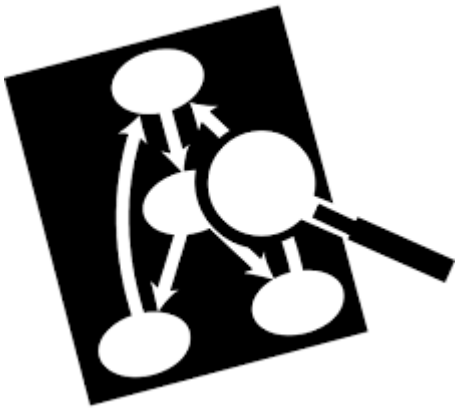



```
package analizador;
import java_cup.runtime.*;


import java.util.ArrayList;

import java_cup.runtime.XMLElement;
import proyecto1_olc1.NodoArbol;
import proyecto1_olc1.Tree;
import proyecto1_olc1.Error;

parser code
{:
    /**
     * Método al que se llama automáticamente ante algún error sintactico.
     **/

    public ArrayList<Tree> leaves = new ArrayList<>();
    public ArrayList<ArrayList> table = new ArrayList<>();

    public void syntax_error(Symbol s){
        System.out.println("Error Sintáctico en la Línea " + (s.left) +
        " Columna "+s.right+ ". No se esperaba este componente: " +s.value+".");
        Error error = new Error();
        error.setColumna(String.valueOf(s.right));
        error.setLinea(String.valueOf(s.left));
        error.setCaracter((String) s.value);
        error.getErrores().add(error);
        error.reporteErrores();
    }

    /**
     * Método al que se llama automáticamente ante algún error sintáctico
     * en el que ya no es posible una recuperación de errores.
     **/
    public void unrecovered_syntax_error(Symbol s) throws java.lang.Exception{
```

GRAPHVIZ

Para hacer posible la generación de los reportes, es decir, la generación del método del árbol, de las tablas de transiciones, siguientes y los autómatas se hizo uso de la herramienta graphviz, la permite visualizar graficas y diagramas, de este modo, para lograr esto se utilizó un lenguaje de descripción de gráficos llamado DOT que permitió crear archivos con extensión DOT donde venia la información necesaria para generar la grafica y convertirla en un archivo JPG que sea posible de ver.





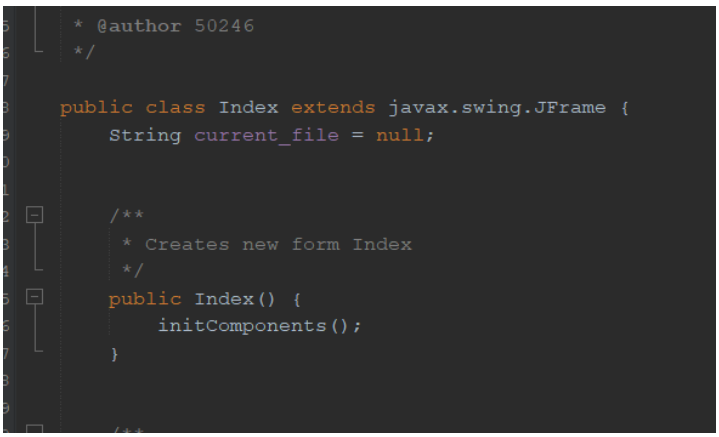
HTML

Finalmente, para lograr la realización del reporte de errores se utilizó otra tecnología altamente popular, la cual es el lenguaje de estructuración llamado HTML, el cual de manera básica es un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, etc. Por lo que, para el reporte de errores se creó un archivo de extensión HTML.

LOGICA DE PROGRAMA

INDEX

El index del proyecto, es decir, la interfaz gráfica se realiza de forma que se creo una clase denominada Index, dentro de ella se colocaron los componentes de las interfaces, siendo estos otones, paneles, cajas de texto, entre otros. Por otro lado, cada uno de estos botones del index se le agregó una funcionalidad las cuales se explicarán de manera resumida y concisa a continuación.



```
 * @author 50246
 */

public class Index extends javax.swing.JFrame {
    String current_file = null;


    /**
     * Creates new form Index
     */
    public Index() {
        initComponents();
    }
```

ABRIR

Para la función abrir como se hizo uso de la librería JfileChooser y BufferedReader para poder detectar y leer línea por línea el archivo que se desea abrir, por consiguiente, al cumplir con esto se procede a mostrar el texto del archivo que se abrió en la caja de texto, por consiguiente, para evitar que el programa culmine al detectar un error se usó el método Try y Catch para lograr recuperarse del error y continuar con normalidad. JfileChooser, específicamente se utilizó para que se pueda usar el showOpenDialog, el cual nos permitirá acceder al archivo que se desea abrir de mejor manera.

```
// </editor-fold>

rivate void openButtonActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.showOpenDialog(fileChooser);
    current_file = fileChooser.getSelectedFile().getAbsolutePath();
    String line;
    String contenido="";
    BufferedReader lector= null;
    try{

        lector=new BufferedReader(new FileReader(current_file));
        line = lector.readLine();
        while(line != null){
            contenido = contenido + line + "\n";
            line = lector.readLine();
            System.out.println(contenido);
        }

    }catch (Exception e){
       System.out.println(e);
    }
    editorTextArea.setText(contenido);
```

GUARDAR

Para guardar el contenido escrito simplemente se extrajo la información en el editor de textos y se procedió a sobrescribir en el archivo que se encontraba previamente abierto, de este modo, esto se consiguió con File OutputStream, utputStreamWriter y BufferedWriter.

```
    }

    private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String file = current_file;
        String text = editorTextArea.getText();

        try (FileOutputStream fos = new FileOutputStream(file);
             OutputStreamWriter osw = new OutputStreamWriter(fos, StandardCharsets.UTF
             BufferedWriter bf = new BufferedWriter(osw))
        {
            bf.write(text);
            System.out.println("El archivo se ha guardado exiotosamente!");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

GUARDAR COMO

La opción de guardar como se desarrollo de manera similar a la de guardar, con la diferencia que luego de extraer la información de la caja de texto se creo un nuevo archivo de manera que se realizo el nuevo archivo, de este modo, se utilizó las librerías JFileChooser, fileChooser, BufferedWriter, FileWriter.

```
private void saveasButtonActionPerformed(java.awt.event.
    JFileChooser fileChooser = new JFileChooser();
     fileChooser.showOpenDialog(fileChooser);
```

GENERADOR ANALIZADORES

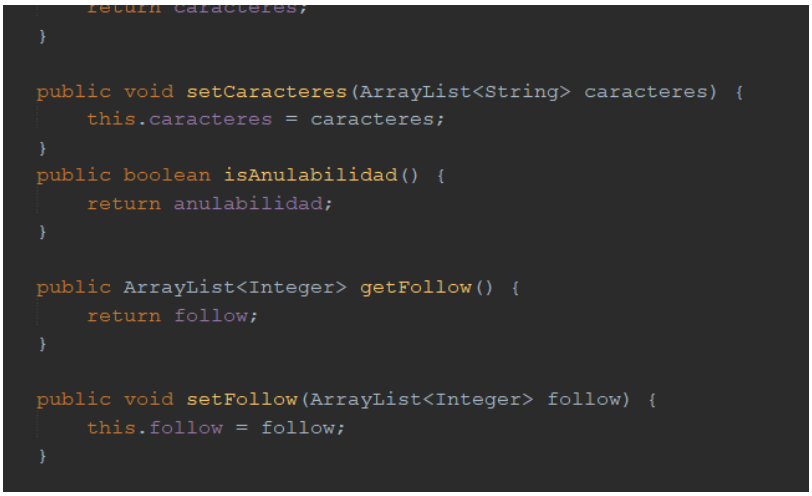
Para generar los analizadores se contó con una clase en específico, la cual se denomino como Generador, dentro de ella se ejecuta el archivo Jflex y Cup que se explico con anterioridad, por consiguiente, se generan los analizadores tanto sintáctico como léxico.

```
private static void generarCompilador(){
    try {

        String ruta = "src/analizador/";
        //ruta donde tenemos los archivos con extension .jflex y .cup
        String opcFlex[] = { ruta + "lexico.jflex", "-d", ruta };
        jflex.Main.generate(opcFlex);
        String opcCUP[] = { "-destdir", ruta, "-parser", "parser", ruta + "sintactico.cup" };
        java_cup.Main.main(opcCUP);
```

CLASE NODOARBOL

La clase nodo árbol es altamente importante, esta clase contiene los atributos que contiene cada nodo de árbol, es decir, los caracteres, la anulabilidad, los primeros, los

últimos, el numero de hoja, entre otros. Cada uno de los atributos descritos se le creo un método Get y Setter en esta misma clase para poder manipular de mejor manera.



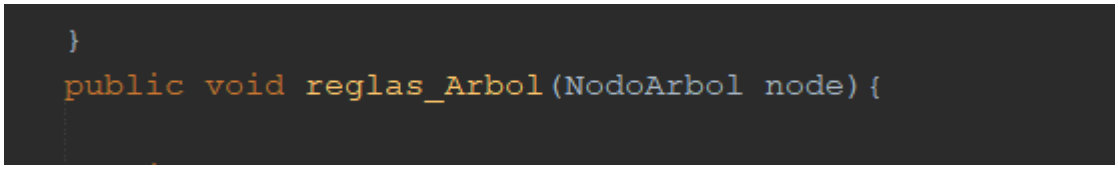
```
    return caracteres;
}

public void setCaracteres(ArrayList<String> caracteres) {
    this.caracteres = caracteres;
}
public boolean isAnulabilidad() {
    return anulabilidad;
}

public ArrayList<Integer> getFollow() {
    return follow;
}

public void setFollow(ArrayList<Integer> follow) {
    this.follow = follow;
}
```

CLASE ARBOL

Dentro de la clase árbol se encontró todo lo relacionado con el método de árbol, es decir, generación de árbol binario, generación de tabla de siguientes, generación de tabla de transiciones. Por consiguiente, se creó una variable de tipo NodoArbol la cual es la raíz del árbol, esta raíz se ingresó en varios métodos que se especificaran a continuación.

a. Árbol

Para la creación del árbol se uso un método denominado, reglasarbol, donde se genero la anulabilidad, primeros y últimos de cada nodo del árbol para poder después mandar estos nodos a otro método donde se creo de manera formal código de graphviz para representar el árbol en una imagen de extensión jpg



```
}
public void reglas_Arbol(NodoArbol node){
```

```
    }
    reglas_Arbol(node.getIzquierdo());
    reglas_Arbol(node.getDerecho());
```

b. Tabla de siguientes

La tabla de siguientes hizo uso de una lista de objetos tipo NodoArbol, dentro de ella se almaceno todos los nodos que son hojas y posteriormente se calculo sus respectivos siguientes con la regla establecida para determinar nodos siguientes. Luego, se recorrio la lista de siguientes para poder generar el archivo dot para posteriormente generar el JPG

```
}
public void reglas_siguientes(NodoArbol node){
    if(node==null){
```

```
if( node.isHojas()){
  //   <tr><td port='port_one'>First port</td><td port='port_two'>Second port</td><td port='port_three'>Third port</td></tr>
  if(node.getCharacter()=="#"){
      cadena += "<tr><td port=\'port_one\'>"+node.getCharacter()+"</td><td port=\'port_two\'>"+node.getNumber()+"</td><td port=\'p

  }else{
      cadena += "<tr><td port=\'port_one\'>"+node.getCharacter()+"</td><td port=\'port_two\'>"+node.getNumber()+"</td><td port=\'
  }
```

c. Tabla de transiciones

Para la tabla de transiciones se creo otra lista de tipo NodoArbol que tiene dentro de ella todos los nodos que tienen transiciones, por lo que cada nodo ingresado en esta lista tiene dentro de el los atributos de transición, estados, entre otros. Por consiguiente, la lista se llena haciendo uso de las reglas de las transiciones y después se recorre para generar el reporte y ubicarlo en su respectiva carpeta.

```
String cadena = "";

cadena += "parent[\n";
cadena += "shape=plaintext\n";
cadena += "label=<\n";
cadena += "<table border='1' cellborder='1'>\n";
 cadena += "<tr><td  rowspan=\"2\">Estado</td><td colspan=\""+(caracteres_hojas.size()-1)+"\">Terminales</td></tr>\n";
 cadena += "<tr>";
 for(int j = 0; j < caracteres_hojas.size()-1; j++) {
    cadena+= "<td>"+caracteres_hojas.get(j)+"</td>\n";
 }

 cadena += " </tr>\n";

 for(int i =0; i < transiciones.size(); i++ ){
     cadena +="<tr><td port='port_one'>"+transiciones.get(i).getId()+transiciones.get(i).getEstados()+"</td>";

         for(int j =0; j < caracteres_hojas.size()-1; j++){
         cadena +=    "<td>"+transiciones.get(i).getTerminal().get(j)+"</td>";
         }
     cadena += "</tr>";
 }

 cadena += "</table>>];\n";
```

ERRORES

Para los errores se creó una clase denominada Error la cual contiene dentro de ella los diversos atributos que tiene un error como lo son su fila, columna, carácter, tipo de error, entre otros. Por consiguiente, se creó los métodos getter y setter para cada uno de estos atributos.

REPORTE ERRORES

Finalmente, para la generación del reporte de errores se creó un método denominado reporteErrores en la clase de Errores, dentro de este método se utilizo el try y catch, además se abrió un archivo nuevo de extensión HTML, por consiguiente, dentro de ella se escribió todo lo relacionado para formar una tabla de errores

```
    public void reporteErrores(){

        try {
            Random random = new Random();
            String contenido = "";
            contenido +="<html>\n";
            contenido +="<head>Errores</head>\n";
            contenido +="<table border=\"1\">\n";
            contenido +="<tr>\n";
            contenido +="<th> # </ th>\n";
            contenido +="<th> Tipo de error </th>\n";
            contenido +="<th> Descripcion </th>\n";
            contenido +="<th> Linea </th>\n";
            contenido +="<th> Columna </th>\n";
            contenido +="</tr>\n";
            for (int j = 0; j < errores.size(); j++){ {

                contenido +="<body>\n";
                contenido +="<tr>\n";
                contenido += "<td>"+(j+1)+"</td>\n";
                contenido += "<td>Léxico</td>\n";
                contenido += "<td>No se esperaba el compone
                contenido += "<td>"+errores.get(j).getLinea
                contenido += "<td>"+errores.get(j).getColum
                contenido +="</tr>\n";
                contenido +="</body>\n";
            }
            contenido +="</table>\n";
            contenido +="</html>\n";
```