Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela Ciencias y sistemas

MANUAL TÉCNICO

Organización de lenguajes y compiladores 1
Rubén Alejandro Ralda Mejia
202111835

Guatemala 23 de marzo del 2023

Paquete principal

Clase main del programa crea un objeto de la clase VentanaPrincipal que es un JFrame, la interfaz se puede ver en el manual de usuario. Las importaciones de la clase son:

```
import Analizadores.Lexico;
import Analizadores.parser;
import ER.ExpresionesRegulares;
import java.awt.Color;
import java.awt.Image;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;
```

Tiene varios métodos

```
private void nuevoArchivo()
private void cargarImagen(int i)
private void interpretar(String entrada)
```

creados como apoyo para la interfaz, el primero crea nuevos archivos con la extensión .olc, el segundo método carga la imagen en un label de una lista de archivos y el último analiza todo el texto si contiene errores.

```
 private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt)
 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt)
 private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt)
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
private void comboBoxImagenesActionPerformed(java.awt.event.ActionEvent
evt)
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

Los métodos anteriores son generados por el IDE donde se encuentra la programación de cada botón o funcionalidad de la interfaz.

Paquete Analizadores

La clase generador de este paquete en base a dos archivos .jflex y .cup ejecuta los comandos para generar los analizadores léxico y sintactico, no importa nada pero necesita de las librerías java-cup-11b.jar y jflex-full-1.9.0.jar para

funcionar y otros paquetes también la utilizan como veremos más adelante. El método es el siguiente:

```
private static void generarCompilador()
```

Importante he de destacar, que los archivos que lee para generar los analizadores son para únicamente la sintaxis del programa. Los archivos son: léxico.jflex y parser.cup.

Las clases Léxico.java, parser.java y sym.java es código generado por las bibliotecas.

Paquete ER

La clase expresión funciona como una especie de nodo para guardar información en una lista. Importaciones:

```
import java.util.Vector;
```

Métodos de la clase:

```
public void sumarUno()
public Expresion(String nombre, Vector<String> valor)
public String getNombre()
public Vector<String> getValor()
public Vector<String> getEntradas()
public int getCantCaracteres()
public void agregarEntrada(String entrada)
```

esta clase es una herramienta de la clase ExpresionesRegulares, los métodos son básicos como sumar la cantidad de caracteres, el constructor y los métodos get.

La clase ExpresionesRegulares es una clase estática para guardar de forma clasificada los datos y es utilizada por el analizador sintactico, las importaciones son:

```
import java.util.Vector;
```

los métodos son:

```
public static Vector<Expresion> getExpresiones()
public static Vector<Expresion> getConjuntos()
public static Vector<String> getExpresion()
public static Vector<String> getConjunto()
public static void agregarExpresion(String nombre)
public static void agregarLexema(String lexema)
public static void agregarLexemaConjunto(String lexema)
public static void agregarAscii(String primero, String segundo)
```

```
public static void agregarAsciiEspecial(String primero, String segundo)
public static void agregarConjunto(String nombre)
public static void sumarAlUltimo()
```

Paquete errores

Contiene una clase llamada ReporteErrores que es estática y su función es cuando ocurre un error en el análisis crear el reporte en un archivo HTML. Sus importaciones son:

```
import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
```

y los métodos son:

```
public static int getConteo()
public static void crear(String tipo, String descripcion, int linea, int
columna)
public static void vaciar()
```

Paquete metodoArbol

La clase Nodo que es la base de toda la funcionalidad para crear los autómatas, guarda la información y tiene hijos izquierdo y derecho para crear el árbol, sus importaciones son:

```
import ER.ExpresionesRegulares;
```

El constructor recibe los parámetros necesarios para enlazar nodos:

```
public Nodo(int identificador, String tipo, boolean anulable, String
primeros, String ultimos, Nodo izquierda, Nodo derecha)
```

y tiene los métodos get y set para los siguientes atributos de la clase:

```
private int identificador;
    private String tipo;
    private boolean anulable;
    private String primeros;
    private String ultimos;
    private Nodo izquierda;
    private Nodo derecha;
```

Por ultimo tiene un método que recorre sus hijos para crear un AFND y poder mostrarlo utilizando la herramienta Grahpviz, este retorna un string con la sintaxis de Grahpviz

```
public String graphviz()
```

La siguiente clase es Árbol y en ella esta la estructura del árbol a su vez la tabla de siguientes del árbol y también la tabla de transiciones, tiene un método que a partir de la tabla muestra su AFD por último tiene un método que recibe la cadena de entrada y recorre la tabla de transiciones para validar si esta correcta por la expresión regular a la que se le aplicó el método del árbol.

Los métodos son los siguientes:

```
public void crearArbol()
public String postOrden(Nodo nodo)
 public void mostrarArbol() {
public void mostrarTablaSiguientes()
public void CrearTablaTransicion()
public void mostrarTablaTransicion()
 public void mostrarAutomata()
public void crearArchivo(String nombre, String cuerpo, String carpeta)
public boolean validarCadenas(String entrada)
public void mostrarAFND()
```

además de los métodos ya mencionados, se cuenta con otro llamado postOrden, recorre el árbol en ese orden y devuelve un string con la sintaxis para conectar un árbol en Graphviz que es utiliza en el método mostrarArbol que su vez llama el método crearArchivo para crear la imagen que más adelante se muestra en la interfaz.

Cada uno de los métodos que muestra algo, llama al método crearArchivo para crear la imagen como se acaba de explicar.