UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

MANUAL TÉCNICO
ExpAnalyzer

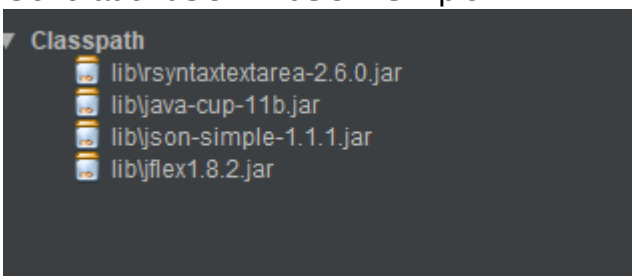
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1
PROYECTO 1

POR:

WALTER GUSTAVO COTI XALIN
201700522

LIBRERIAS

Para la realización de este proyecto se utilizaron las librerías:
Análisis Léxico = Java-Flex (Jflex)
Análisis Sintactico = Java-CUP
Editor de Codigo(TextArea) = Rsyntaxtextarea
Generador JSON = JSON-Simple



ESTRUCTURAS

Terror(Objeto) – Para un manejo más rápido se realizó se dejo un acceso publico a cada uno de sus atributos.

```
public class TErrores {
    public String lexema, tipo, descript;
    public int line, col;

    public TErrores(String lex_,int line_, int col_,String tipo_,String descript_){
        this.lexema = lex_;
        this.line = line_;
        this.col = col_;
        this.tipo = tipo_;
        this.descript = descript_;

    }

}
```

NHoja(Objeto) – Un objeto con las características de cada nodo de un arbol

```
public class NHoja {
    public int ID_nodo;
    public String valor;
    public String tipo;
    public NHoja HojaR;
    public NHoja HojaL;
    public boolean anulable;
    public String primeros;
    public String ultimos;

    public NHoja(String valor_,String tipo_,NHoja HojaL_,NHoja HojaR_){
        this.valor = valor_;
        this.tipo = tipo_;
        this.HojaR = HojaR_;
        this.HojaL = HojaL_;
        this.primeros = "";
        this.ultimos = "";
        try{
            esAnulable( tipo_,HojaL_, HojaR_);
        }catch(Exception e){
            System.out.println(x: "");//evitar los null en las hojas finales
        }
```

Métodos:

- esAnulable(): recibe como parámetro el tipo de hoja/nodo, nodo izquierdo y nodo derecho. Se encarga de asignarle un valor falso o verderol al atributo anulable.

```
public void esAnulable(String tipo_,NHoja HojaL_,NHoja HojaR_){
    switch(tipo_){
        case "h":
            this.anulable = false;
            break;
        case ".":
            this.anulable = HojaL_.anulable && HojaR_.anulable;
            break;
        case "|":
            this.anulable = HojaL_.anulable || HojaR_.anulable;
            break;
        case "?":
            this.anulable = true;
            break;
        case "*":
            this.anulable = true;
            break;
        case "+":
            this.anulable = false;
            break;
    default:
            this.anulable = false;
        break;
    }

}
```

NValidacion (Objeto) – Un objeto creado para cada validación a realizar, posee el nombre de la expresión regular, cadena a evaluar y un tercer atributo para confirmar su validez.

```
public class NValidacion {
    public String cadena,nameAFD,salida;
    public NValidacion(String nameAFD_,String cadena_){
        this.nameAFD = nameAFD_;
        this.cadena = cadena_;
        this.salida = "";
    }
```

AFD(Objeto) – recibe como parámetro el nombre de la expresión regular, lista de conjuntos utilizados y el nodo raíz del árbol.

```
public class AFD {
    public String nameAFD;
    NHoja raiz;
    int id = 1;
    int aux =  1;
    List<String> alfabeto = new ArrayList<>();
    List<String[]> transiciones = new ArrayList<>();
    List<String> terminales = new ArrayList<>();
    Map<String,String> conjuntos = new HashMap<>();
    Map<String, String> Lalfabeto = new HashMap<>();
    Map<Integer, String[]> LSiguientes = new HashMap<>();
    Map<String, String[]> LTransiciones = new HashMap<>();

    public AFD(String nombreAFD_, Map<String,String> conjuntos_,NHoja raiz_){
        this.nameAFD=nombreAFD_;
        this.conjuntos.putAll( m: conjuntos_);
        this.raiz = raiz_;
        try {
            generarDatos();
            CrearTransiciones();

        } catch (Exception e) {
            //System.out.println(e+"Es aquihdspm");
        }

    }
```

Métodos:
generarDatos(): Utilizado para agregar el id, primeros, últimos a cada nodo del árbol. Utilizando el método genData de manera recursiva para recorrer todo el árbol.
genData():recorre nodo por nodo, validando si existe un hijo derecho o izquierdo.
damePri_**ult**(): Agrega al objeto NHoja los primeros y últimos.

genSiguientes(): Produce la tabla de siguientes.
GraficarArbol():Grafica el árbol recorriendo cada nodo desde la raíz hasta los hijos.

getID(): Al momento de graficar es necesario eliminar el id de los nodos que no son hojas, este método retorna el id solamente si es hoja.

GraficarSiguientes():Grafica la tabla de siguientes. Produce un jpg que es mostrado en el visor de imágenes.

CrearTransiciones():Crea las transiciones necesarias para el automata de forma recursiva.

createTransition():Apoya a la función principal CrearTransiciones.

GraficarTransiciones(): Grafica la tabla de transiciones. Produce una imagen jpg que es mostrada en el visor de imágenes

GraficarAFD(): Grafica el automata correspondiente a la expresion regular, apoyado de la lista de transiciones obtenida con el método CrearTransiciones.

CrearDirectorio():Recibe como parámetro la ruta a generarse que en este caso es la ruta donde se encuentran los reportes y graficas.

ValidarCadena():Comprueba la cadena recibida como parámetro. Retorna un boolean.

ValidarConjunto():Utilizado en el método Validar cadena para saber si el el carácter leído pertenece al conjunto establecido.

FORM

HOME(): Inicializa el formulario, además de compilar los analizadores léxico y sintactico.

configTExtArea(): Configura el modo en el que se ve el editor de código, agregándole tema o marcador de sintaxis.

updateTextArea():Agrega el contenido de consola a el textarea designado como consola.

redirectSystemStreams(): (Método obtenido de internet) Redirige toda la salida a un elemento gui en especifico, en este caso es utilizado un TextArea.

OpenFile():Método Utilizado para Mostrar la ventana emergente con el navegador de archivos para posteriormente abrirlo.
SaveAsFile(): Comprueba si existe una ruta para ejecutar una escritura o sobreescritura.

ShowImagenes(): Obtiene las imágenes de las carpetas de reportes, recibe como parámetro el tipo de grafica(Árbol, tablas, o autómatas) y también el nombre de la imagen.

AnalizarContent():Ejecuta el analizador léxico y sintáctico dentro de la aplicación, agrega a las listas locales los valores obtenidos en el análisis del código.
Hace el llamado al método de comprobación de cadena.

GenerarAutomatas():
Genera todas las graficas correspondientes para cada expresión regular.

crearJSON():Recorre el listado de Validaciones y los agrega al archivo json de salidas.

ReporteErrores(): Recorre el listado de errores Globales, y genera un reporte html con el detalle de cada uno de los errores.