2023

Proyecto 1

Manual Técnico

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1
SECCION C
TOBÍAS RAFAEL ZAMORA SANTOS, 202010828

Índice

Introducción

El siguiente reporte describe los métodos, funciones, estructuras entre otros correspondientes al proyecto 1, con los cuales se obtuvo el resultado esperado, un programa básico desarrollado en el lenguaje Java, que consta de la carga de un archivo que contiene expresiones regulares con las cual se generan los pasos correspondientes para llegar a su correspondiente AFD.

1. Técnica de Programación

1.1 Programación Orientada a Objetos

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas. La programación orientad a objetos disminuye los errores y promociona la reutilización del código, Java es un lenguaje orientado a objetos, por este motivo se decidió trabajar de este modo.

2. Convenciones de Nomenclatura

2.1 Declaración de Variables

Para la declaración de variables, se usó un nombre el cual hiciera referencia a lo que se contendrá en la variable, por ejemplo, al momento que se iban ingresando nombres, la variable correspondiente que contendrá el nombre fue nombrada “nombre”.

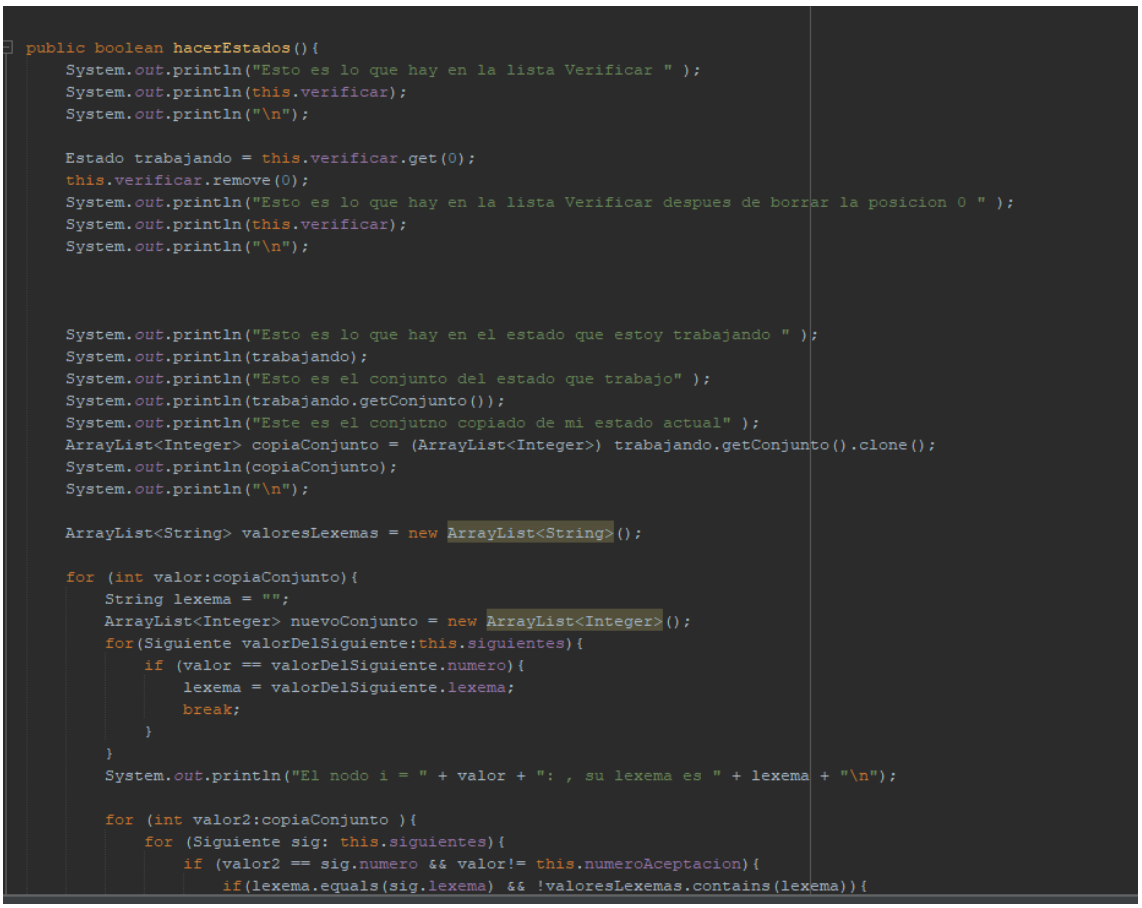
```
String nombre;
```

El nombre de cada variable corresponde a lo que contendrá.

2.2 Métodos

Para la declaración de Métodos, el nombre de cada método corresponde a la acción que estos deben realizar, cabe mencionar que los métodos solo realizan las instrucciones que indiquemos, no retornan valor alguno.

Método para agregar estados

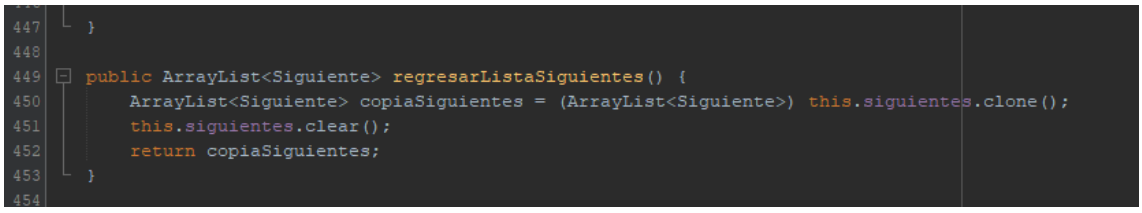


```
public boolean hacerEstados(){
    System.out.println("Esto es lo que hay en la lista Verificar " );
    System.out.println(this.verificar);
    System.out.println("\n");

    Estado trabajando = this.verificar.get(0);
    this.verificar.remove(0);
    System.out.println("Esto es lo que hay en la lista Verificar despues de borrar la posicion 0 " );
    System.out.println(this.verificar);
    System.out.println("\n");



    System.out.println("Esto es lo que hay en el estado que estoy trabajando " );
    System.out.println(trabajando);
    System.out.println("Esto es el conjunto del estado que trabajo" );
    System.out.println(trabajando.getConjunto());
    System.out.println("Este es el conjutno copiado de mi estado actual" );
    ArrayList<Integer> copiaConjunto = (ArrayList<Integer>) trabajando.getConjunto().clone();
    System.out.println(copiaConjunto);
    System.out.println("\n");

    ArrayList<String> valoresLexemas = new ArrayList<String>();

    for (int valor:copiaConjunto){
        String lexema = "";
        ArrayList<Integer> nuevoConjunto = new ArrayList<Integer>();
        for(Siguiente valorDelSiguiente:this.siguientes){
            if (valor == valorDelSiguiente.numero){
                lexema = valorDelSiguiente.lexema;
                break;
            }
        }
        System.out.println("El nodo i = " + valor + ": , su lexema es " + lexema + "\n");

        for (int valor2:copiaConjunto ){
            for (Siguiente sig: this.siguientes){
                if (valor2 == sig.numero && valor!= this.numeroAceptacion){
                    if(lexema.equals(sig.lexema) && !valoresLexemas.contains(lexema)){
```

2.3 Funciones

Para la declaración de funciones, al igual que los métodos, el nombre de cada función hace referencia a la acción que deben realizar, en este caso las funciones si retornan valores.

Función para retornar una lista del tipo indicado (nos retorna una lista del tipo Siguiente)

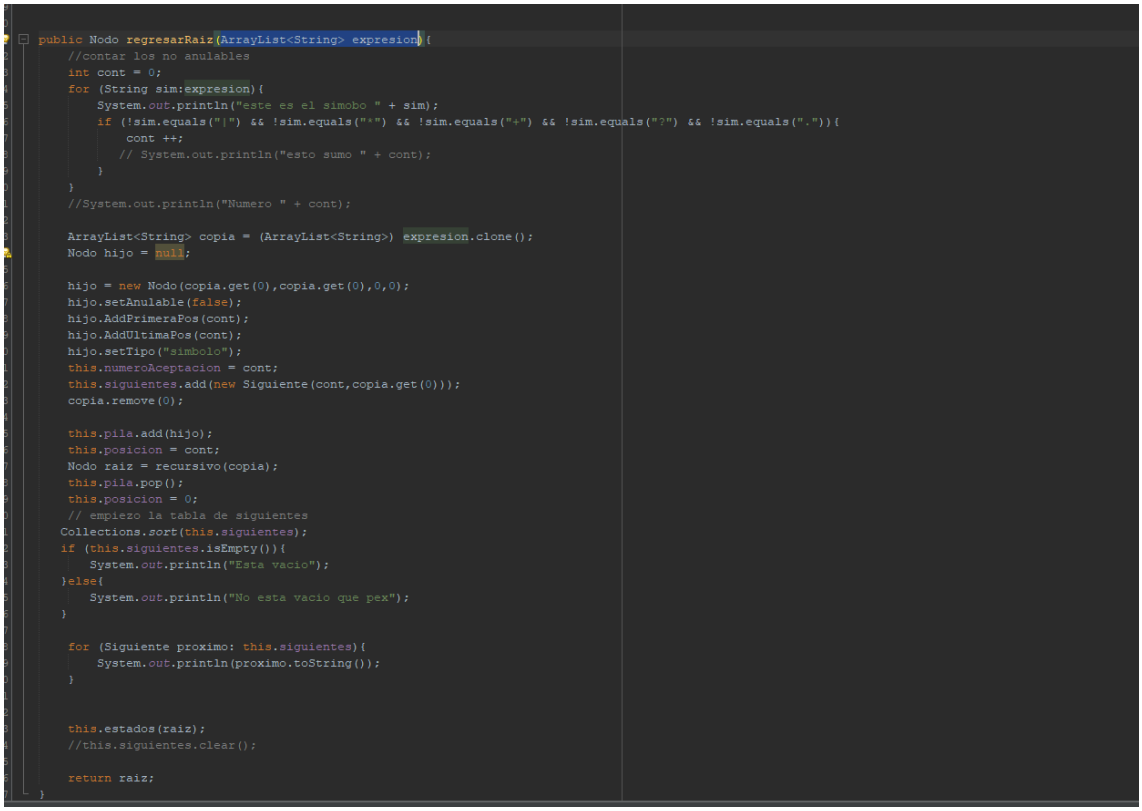


```
447     }
448
449     public ArrayList<Siguiente> regresarListaSiguientes() {
450         ArrayList<Siguiente> copiaSiguientes = (ArrayList<Siguiente>) this.siguientes.clone();
451         this.siguientes.clear();
452         return copiaSiguientes;
453     }
454
```

3. Métodos Principales

3.1 regresarRaiz(ArrayList<String> expresion)

Este método analiza cada expresión que venga en la lista y desglosa cada valor de ella, logrando así armar los nodos de el árbol de expresiones.



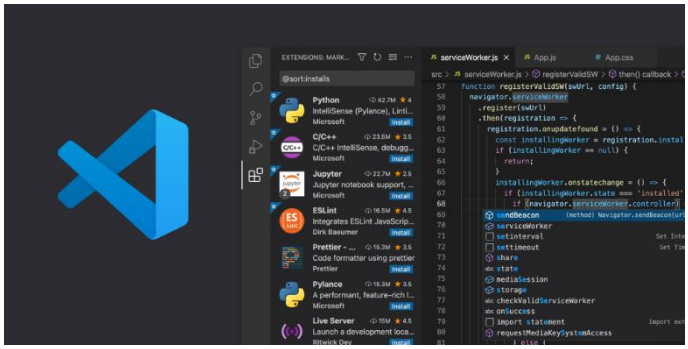
```
public Nodo regresarRaiz(ArrayList<String> expresion){
    //contar los no anulables
    int cont = 0;
    for (String sim:expresion){
        System.out.println("este es el simobo " + sim);
        if (!sim.equals("|") && !sim.equals("*") && !sim.equals("+") && !sim.equals("?") && !sim.equals(".")){
            cont ++;
           // System.out.println("esto sumo " + cont);
        }
    }
    //System.out.println("Numero " + cont);

    ArrayList<String> copia = (ArrayList<String>) expresion.clone();
    Nodo hijo = null;

    hijo = new Nodo(copia.get(0),copia.get(0),0,0);
    hijo.setAnulable(false);
    hijo.AddPrimeraPos(cont);
    hijo.AddUltimaPos(cont);
    hijo.setTipo("simbolo");
    this.numeroAceptacion = cont;
    this.siguientes.add(new Siguiente(cont,copia.get(0)));
    copia.remove(0);

    this.pila.add(hijo);
    this.posicion = cont;
    Nodo raiz = recursivo(copia);
    this.pila.pop();
    this.posicion = 0;
    // empiezo la tabla de siguientes
   Collections.sort(this.siguientes);
   if (this.siguientes.isEmpty()){
       System.out.println("Esta vacio");
   }else{
       System.out.println("No esta vacio que pex");
   }

    for (Siguiente proximo: this.siguientes){
        System.out.println(proximo.toString());
    }


    this.estados(raiz);
    //this.siguientes.clear();

    return raiz;
}
```

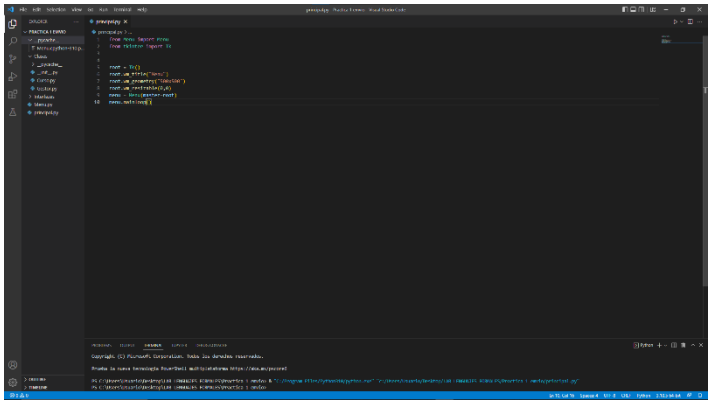
4. Requerimiento o Funcionalidad Especifica

Para la elaboración de este proyecto se usaron los siguientes programas y herramientas:

4.1 Visual Studio Code

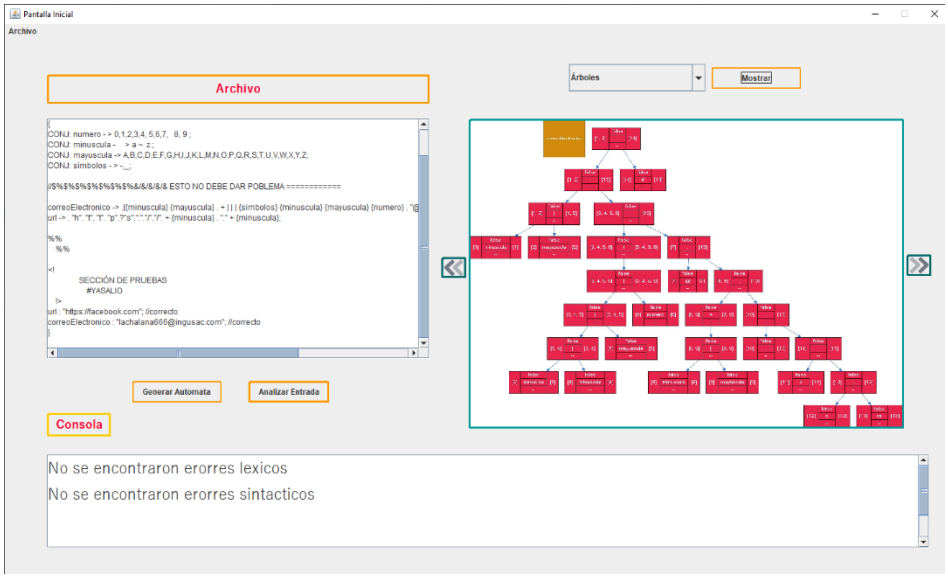
Se uso este editor de código debido a que la forma de visualizar el código es mejor que en java ya que podemos agregar extensiones que nos faciliten el cómo se ve el código.





4.2 Swing

Se uso la librería Swing para la interfaz gráfica, el ide NetBeans trae la opción de Drag and Drop que nos facilita la implementación.



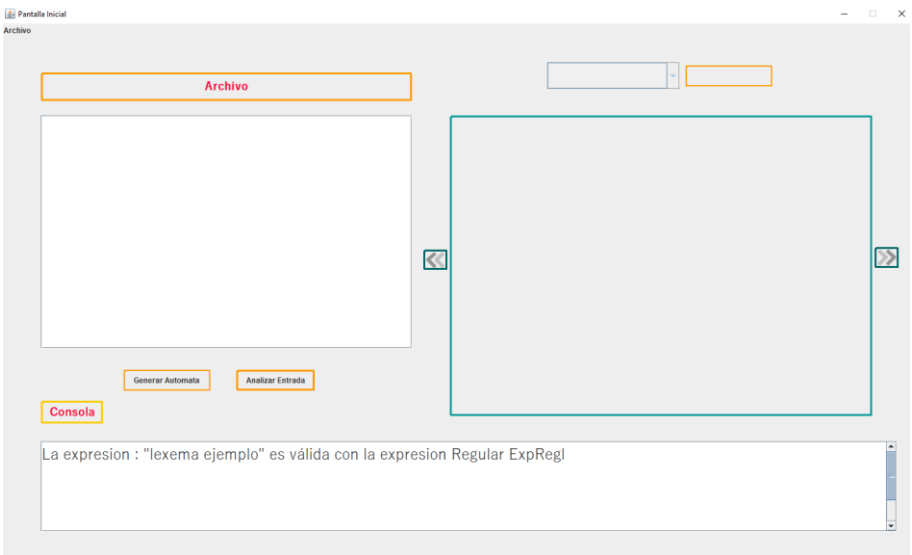
4.3 Java

Se uso java como lenguaje principal para el proyecto ya que era obligatorio y las librerías como Jflex y Jcup se tenían que implementar en este lenguaje.

```java
*/
package Analizador;

/**
 *
//  * @author Pilo Tuy
 */
public class Generador {
    public static void main(String[] args) {
        generarCompilador();
    }

    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 + "Parser.cup" };
            java_cup.Main.main(opcCUP);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5. Interfaces Principales

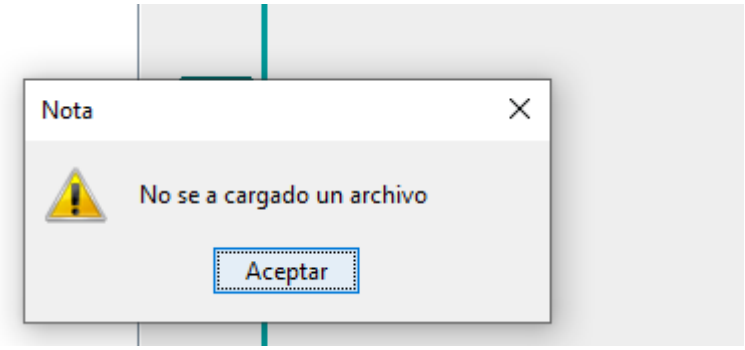
5.1 Pantalla Inicial



En esta primera interfaz se muestran las opciones que se tendrán en el programa.

5.1.1 Validación #1

Si se intenta guardar un archivo si antes haber abierto uno se mostrara un mensaje indicándoselo.



6. Planificación o Estimación

6.1 Tarea #1
[Creación de análisis léxico y sintáctico ---------- [15 Horas]

6.2 Tarea #2
[Creación de Interfases] ---------- [2 Horas]

6.3 Tarea # 3
[Validaciones y lógica del programa] ---------- [20 Horas]

7. Glosario

I. IndexError: es un error usual al momento de agregar un valor de índice fuera del rango de nuestras listas.
II. Identación: este término significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y distinguirlo mejor del texto adyacente
III. ValueError: es un error que sucede cuando trato de convertir un valor a otro tipo que no es posible, por ejemplo, una cadena de caracteres a números.