

| Nombre de la práctica | ANALIZADOR LEXICO (UNIDAD 4) |          |   | No.                           | 4        |
|-----------------------|------------------------------|----------|---|-------------------------------|----------|
| Asignatura:           | LENGUAJES Y AUTÓMATAS I      | Carrera: | INGENIERÍA EN SISTEMAS COMPUTACIONALES-3501 | Duración de la práctica (Hrs) | 10 horas |

NOMBRE DEL ALUMNO: Vanesa Hernández Martínez

GRUPO: 3501

## I. Competencia(s) específica(s):

Construye un analizador léxico a partir de un lenguaje de programación.

Encuadre con CACEI: Registra el (los) atributo(s) de egreso y los criterios de desempeño que se evaluarán en la materia.

| No. atributo | Atributos de egreso del PE que impactan en la asignatura   | No. Criterio | Criterios de desempeño   | No. Indicador | Indicadores   |
|--------------|--|--------------|--|---------------|---|
| 2            | El estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de Ingeniería en Sistemas Computacionales | CD1          | Identifica metodologías y procesos empleados en la resolución de problemas | I1            | Identificación y reconocimiento de distintas metodologías para la resolución de problemas |
|              |  | CD2          | Diseña soluciones a problemas, empleando metodologías apropiadas al área   | I1            | Uso de metodologías para el modelado de la solución de sistemas y aplicaciones            |
|              |  |              |  | I2            | Diseño algorítmico (Representación de diagramas de transiciones)                          |
| 3            | El estudiante plantea soluciones basadas en tecnologías empleando su juicio ingenieril para valorar necesidades, recursos y resultados esperados.              | CD1          | Emplea los conocimientos adquiridos para el desarrollar soluciones         | I1            | Elección de metodologías, técnicas y/o herramientas para el desarrollo de soluciones      |
|              |  |              |  | I2            | Uso de metodologías adecuadas para el desarrollo de proyectos                             |
|              |  |              |  | I3            | Generación de productos y/o proyectos   |
|              |  | CD2          | Analiza y comprueba resultados   | I1            | Realizar pruebas a los productos obtenidos  |
|              |  |              |  | I2            | Documentar información de las pruebas realizadas y los resultados                         |

## II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):

Laboratorio de cómputo y equipo de cómputo personal.

## III. Material empleado:

- Equipo de cómputo
- Software para desarrollo: Apache Netbeans IDE 22

## IV. Desarrollo de la práctica:

### ANALIZADOR LÉXICO

## Descripción del problema

Mi proyecto es un analizador léxico que procesa código fuente en español, dividiéndolo en componentes básicos llamados *tokens*. Su propósito principal es identificar y clasificar cada elemento del código según su función en el lenguaje que diseñé, como palabras clave, identificadores, operadores y símbolos. Esta clasificación es crucial porque representa el primer paso en el procesamiento de un programa, asegurando que el código cumpla con las reglas léxicas definidas.

El funcionamiento del proyecto inicia cuando el usuario ingresa un fragmento de código en un área de texto especialmente diseñada para ello. Al presionar el botón de análisis léxico, el sistema examina el código de manera secuencial, separándolo en lexemas y comparándolos con un conjunto de reglas predefinidas. Estas reglas, que yo mismo definí al diseñar el lenguaje, permiten determinar a qué categoría pertenece cada lexema y asignarle un token correspondiente. Por ejemplo, el sistema puede identificar si un término es un tipo de dato, una palabra reservada, un delimitador o cualquier otro elemento válido dentro del lenguaje.

Internamente, el analizador utiliza patrones definidos por expresiones regulares para reconocer los lexemas. A medida que procesa el código, verifica cada componente, valida su conformidad con el lenguaje y genera una representación estructurada de los tokens. Esta representación no solo permite al usuario entender cómo está compuesto su código, sino que también sirve como base para futuras etapas de análisis, como el análisis sintáctico.

En resumen, mi proyecto actúa como un puente entre el código fuente y su interpretación, desglosando cada elemento del programa de manera clara y ordenada. Funciona como una herramienta de validación inicial, garantizando que el código esté correctamente estructurado antes de ser procesado por etapas más avanzadas. Es un ejemplo práctico de cómo los principios de autómatas y gramáticas regulares se aplican en el diseño de lenguajes y herramientas de programación.

## ¿Qué es un analizador léxico?

El analizador léxico tiene como principal objetivo interpretar el código fuente ingresado por el programador para facilitar su comprensión y procesamiento. Este análisis se realiza siguiendo un conjunto de reglas predefinidas que permiten identificar patrones válidos dentro del lenguaje de programación.

### Principales Funciones

1. **Lectura del Código Fuente:** Toma el texto escrito por el usuario.
2. **División en Lexemas:** Identifica las unidades básicas del código, como palabras clave, símbolos y delimitadores.
3. **Clasificación:** Asocia cada lexema con su respectivo token.
4. **Generación de Resultados:** Presenta los resultados en un formato que muestra cada lexema y su categoría correspondiente.

### Para qué Sirve

El analizador léxico prepara el código para ser procesado por etapas más avanzadas, asegurándose de que siga las reglas básicas del lenguaje y reportando errores léxicos en caso de que existan. Además, permite trabajar con lenguajes personalizados o adaptados, como en este proyecto, donde el lenguaje base está en español.

### Importancia y Usos

- **Detección de Errores Iniciales:** Identifica errores básicos como el uso de caracteres inválidos.
- **Interoperabilidad:** Sirve de puente entre el código escrito por el programador y las etapas más complejas de un compilador.
- **Adaptabilidad:** Es la base para diseñar lenguajes personalizados o específicos para ciertos dominios.
- **Uso Educativo:** Facilita la enseñanza sobre cómo funcionan los compiladores.

## Tokens

En el análisis léxico de un lenguaje de programación, los **tokens** son las unidades mínimas significativas que componen el código fuente. Representan las piezas fundamentales con las que se construye la lógica y estructura de un programa, facilitando la interpretación del lenguaje por la máquina.

En este proyecto, se diseñó un lenguaje de programación adaptado al español, con un conjunto definido de **tokens** que establecen las reglas del lenguaje. Cada token está clasificado en categorías según su función:

1. **Operadores**

Incluyen operadores aritméticos (+, -, \*, /), relacionales (>, <, ==, !=), lógicos (&, |, !), y otros como incremento (++) y asignación (=>).

2. **Signos y constantes matemáticas**

Representan signos como positivo (pos), negativo (neg), y constantes universales como PI y E.

3. **Delimitadores**

Se utilizan para estructurar el código, como paréntesis, corchetes, y llaves ((, [, {}), así como delimitadores de cadenas (<<, >>, ", ').

4. **Comentarios**

Tokens como #, /\*, y \*/ permiten incluir comentarios en el código para mejorar su legibilidad.

5. **Tipos de datos**

Definen las bases del lenguaje, con soporte para enteros (ent), decimales (dec), valores booleanos (v, f), y cadenas (cadena).

6. **Estructuras de control**

Incluyen instrucciones como si, siNo, repite, y mientras para definir la lógica condicional y los ciclos.

7. **Declaraciones y entrada/salida**

Permiten definir funciones (fun), métodos (metodo), y clases (clase), además de gestionar la entrada (leer) y salida de datos (imprime).

## 8. Componentes de cadenas

Incluyen tokens como salto de línea (\n) y caracteres especiales (:, \$).

Cada uno de estos tokens tiene un **lexema asociado**, que es la representación literal en el código fuente. Por ejemplo, el token de suma corresponde al lexema "+", y el de impresión al lexema "imprime".

Los tokens son, en esencia, las piezas clave que dan forma al lenguaje, delimitando lo que se puede expresar dentro de él y cómo se interpreta cada elemento. Gracias a ellos, es posible transformar el texto escrito por el usuario en estructuras que la máquina puede procesar, facilitando la comunicación entre el programador y el sistema.

A continuación, muestro la tabla de tokens en la cual está basado el proyecto:

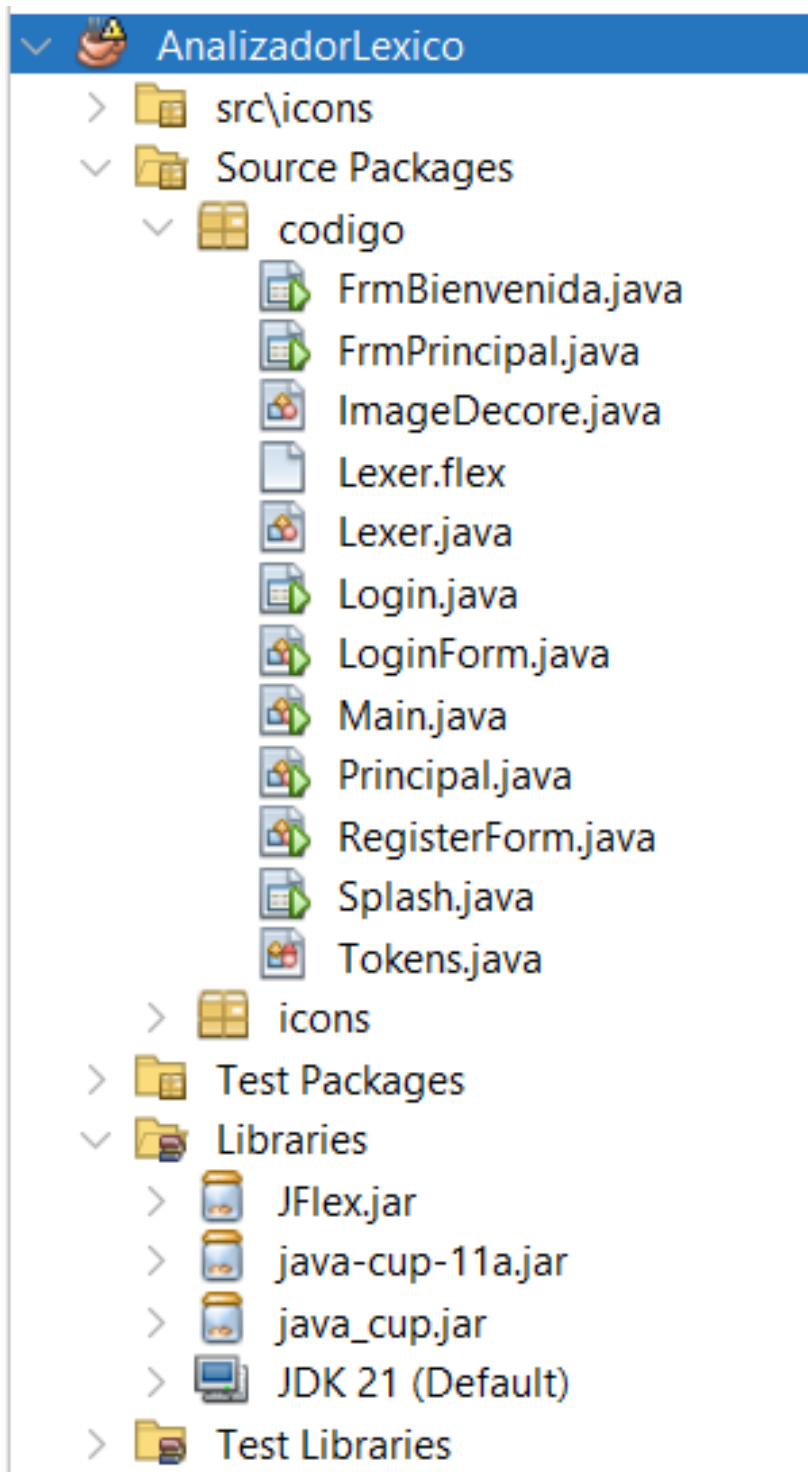
|                          | Número de token | Token              | Lexema |
|--------------------------|-----------------|--------------------|--------|
| Operadores aritméticos   | 1               | suma               | "+"    |
|                          | 2               | resta              | "-"    |
|                          | 3               | multiplicacion     | "*"    |
|                          | 4               | division           | "/"    |
| Operadores relacionales  | 5               | mayor              | ">"    |
|                          | 6               | menor              | "<"    |
|                          | 7               | mayorIgual         | ">="   |
|                          | 8               | menorIgual         | "<="   |
|                          | 9               | comparacion        | "=="   |
|                          | 10              | distinto           | "!="   |
| Operadores lógicos       | 11              | y                  | "&"    |
|                          | 12              | o                  | " "    |
|                          | 13              | no                 | "!"    |
| Operadores               | 14              | modulo             | "%"    |
|                          | 15              | potencia           | "**"   |
|                          | 16              | raiz               | "raiz" |
|                          | 17              | punto              | "."    |
|                          | 18              | asignacion         | ">="   |
|                          | 19              | igual              | "="    |
|                          | 20              | concatenacion      | "con"  |
| Signos                   | 21              | positivo           | "pos"  |
|                          | 22              | negativo           | "neg"  |
| Constantes Matemáticas   | 23              | pi                 | "PI"   |
|                          | 24              | euler              | "E"    |
| Delimitadores            | 25              | parentesisApertura | "{"    |
|                          | 26              | parentesisCierre   | "}"    |
|                          | 27              | corcheteApertura   | "["    |
|                          | 28              | corcheteCierre     | "]"    |
|                          | 29              | llaveApertura      | "{"    |
|                          | 30              | llaveCierre        | "}"    |
| Delimitadores de cadenas | 31              | inicioTexto        | <<     |
|                          | 32              | finalTexto         | >>     |
|                          | 33              | comillaDoble       | "      |
|                          | 34              | comillaSimple      | '      |

|                           |    |                      |                  |
|---------------------------|----|----------------------|------------------|
| Comentarios               | 35 | comentario           | "#"              |
|                           | 36 | inicioComentario     | "{"              |
|                           | 37 | finComentario        | "}"              |
| Tipos de Datos            | 38 | entero               | "ent"            |
|                           | 39 | decimal              | "dec"            |
|                           | 40 | verdadero            | "v"              |
|                           | 41 | falso                | "f"              |
|                           | 42 | cadena               | "cadena"         |
| Estructuras de control    | 43 | si                   | "si"             |
|                           | 44 | siNo                 | "siNo"           |
|                           | 45 | siNoSi               | "siNoSi"         |
|                           | 46 | cicloRepite          | "repite"         |
|                           | 47 | mientras             | "mientras"       |
|                           | 48 | hacer                | "hacer"          |
|                           | 49 | seleccionador        | "seleccionar"    |
|                           | 50 | caso                 | "caso"           |
|                           | 51 | predeterminado       | "predeterminado" |
|                           | 52 | detener              | "detener"        |
| Declaraciones             | 54 | metodo               | "metodo"         |
|                           | 55 | arreglo              | "arreglo"        |
|                           | 56 | funcion              | "fun"            |
|                           | 57 | clase                | "clase"          |
|                           | 58 | procedimiento        | "proc"           |
|                           | 59 | principal            | "principal"      |
|                           | 60 | biblioteca           | "biblioteca"     |
| Entrada y salida de Datos | 61 | imprimir             | imprime          |
|                           | 62 | leer                 | leer             |
|                           | 63 | importar             | "importa"        |
|                           | 64 | retornar             | "retorna"        |
|                           | 65 | nulo                 | "nulo"           |
| Componentes de una cadena | 66 | salto de linea       | "\n"             |
|                           | 67 | caracteresEspeciales | ".", "\$"        |
|                           | 68 | letrasMayusculas     | [A-Z]            |
|                           | 69 | letrasMinusculas     | [a-z]            |
|                           | 70 | digitos              | [0-9]            |

## Estructura del proyecto

El proyecto este compuesto por los siguientes archivos:

- Lexer.flex
- Lexer.java
- Tokens.java
- Principal.java
- FrmPrincipa.java
- Splash.java
- LoginForm.java
- RegisterForm
- FrmBienvenida.java





## Lexer.flex

El archivo Lexer.flex es un componente fundamental en el desarrollo de un compilador o intérprete. Su propósito es definir el analizador léxico, que se encarga de descomponer el código fuente en unidades básicas conocidas como tokens. Estos tokens representan elementos clave del lenguaje, como operadores, palabras reservadas, identificadores y símbolos especiales.

El propósito principal de este archivo es procesar y clasificar el código fuente de forma estructurada, eliminando los elementos no relevantes, como los espacios en blanco y los comentarios, para facilitar el análisis sintáctico posterior. Al identificar y organizar los elementos del código en tokens, este archivo prepara el terreno para que las siguientes fases del compilador o intérprete puedan interpretarlos de manera efectiva.

```
1 package codigo;
2 import static codigo.Tokens.*;
3 %%
4 %class Lexer
5 %type Tokens
6 L=[a-zA-Z_]+
7 D=[0-9]+
8 espacio=[ ,\t,\r]+
9 %{
10     public String lexeme;
11 %}
12 %%
13 /* Espacios en blanco */
14 {espacio} { /*Ignore*/ }
15
16 /* Comentarios */
17 ( "/" (".")* ) { /*Ignore*/ }
18
19 /* Salto de línea */
20 ( "\n" ) { return Línea; }
21
22 /* OPERADORES ARITMÉTICOS */
23
24 /* Operador Suma */
25 ( "+" ) { lexeme=yytext(); return Suma; }
26
27 /* Operador Resta */
28 ( "-" ) { lexeme=yytext(); return Resta; }
29
30 /* Operador Multiplicación */
31 ( "*" ) { lexeme=yytext(); return Multiplicación; }
32
33 /* Operador División */
34 ( "/" ) { lexeme=yytext(); return División; }
```

```
37  /* OPERADORES RELACIONALES */
38
39  /* Operador Mayor */
40  ( ">" ) {lexeme=yytext(); return Mayor;}
41
42  /* Operador Menor */
43  ( "<" ) {lexeme=yytext(); return Menor;}
44
45  /* Operador Mayor o Igual */
46  ( ">=" ) {lexeme=yytext(); return MayorIgual;}
47
48  /* Operador Menor o Igual */
49  ( "<=" ) {lexeme=yytext(); return MenorIgual;}
50
51  /* Operador Comparación */
52  ( "==" ) {lexeme=yytext(); return Comparacion;}
53
54  /* Operador Distinto */
55  ( "!=" ) {lexeme=yytext(); return Distinto;}
56
57
58  /* OPERADORES LOGICOS */
59
60  /* Operador Y */
61  ( "&" ) {lexeme=yytext(); return Y;}
62
63  /* Operador O */
64  ( "|" ) {lexeme=yytext(); return O;}
65
66  /* Operador No */
67  ( "!" ) {lexeme=yytext(); return No;}
```

```
70  /* OPERADORES*/
71
72  /* Operador Módulo */
73  ( "%" ) {lexeme=yytext(); return Modulo;}
74
75  /* Operador Potencia */
76  ( "^" ) {lexeme=yytext(); return Potencia;}
77
78  /* Función Raíz */
79  ( "raiz" ) {lexeme=yytext(); return Raiz;}
80
81  /* Punto */
82  ( "\\." ) {lexeme=yytext(); return Punto;}
83
84  /* Operador Asignación */
85  ( "=>" ) {lexeme=yytext(); return Asignacion;}
86
87  /* Operador Igual */
88  ( "=" ) {lexeme=yytext(); return Igual;}
89
90  /* Operador Concatenación */
91  ( "con" ) {lexeme=yytext(); return Concatenacion;}
92
93
94  /* SIGNOS */
95
96  /* Operador Positivo */
97  ( "pos" ) {lexeme=yytext(); return Positivo;}
98
99  /* Operador Negativo */
100 ( "neg" ) {lexeme=yytext(); return Negativo;}
```



```
103  /* INCREMENTO */
104
105  /* Incremento */
106  ( "++" ) {lexeme=yytext(); return Incremento;}
107
108  /* Decremento */
109  ( "--" ) {lexeme=yytext(); return Decremento;}
110
111  /* CONSTANTES MATEMATICAS*/
112
113  /* Constante PI */
114  ( "PI" ) {lexeme=yytext(); return Pi;}
115
116  /* Constante Euler */
117  ( "E" ) {lexeme=yytext(); return Euler;}
118
119  /* DELIMITADORES */
120
121  /* Paréntesis Apertura */
122  ( "(" ) {lexeme=yytext(); return ParentesisApertura;}
123
124  /* Paréntesis Cierre */
125  ( ")" ) {lexeme=yytext(); return ParentesisCierre;}
126
127  /* Corchete Apertura */
128  ( "[" ) {lexeme=yytext(); return CorcheteApertura;}
129
130  /* Corchete Cierre */
131  ( "]" ) {lexeme=yytext(); return CorcheteCierre;}
132
133  /* Llave Apertura */
134  ( "{" ) {lexeme=yytext(); return LlaveApertura;}
135
136  /* Llave Cierre */
137  ( "}" ) {lexeme=yytext(); return LlaveCierre;}
```

```
140  /* DELIMITADOR DE CADENAS*/
141
142  /* Inicio de Texto */
143  ( "<<" ) {lexeme=yytext(); return InicioTexto;}
144
145  /* Final de Texto */
146  ( ">>" ) {lexeme=yytext(); return FinalTexto;}
147
148  /* Comilla Doble */
149  ( "\"" ) {lexeme=yytext(); return ComillaDoble;}
150
151  /* Comilla Simple */
152  ( "'" ) {lexeme=yytext(); return ComillaSimple;}
153
154  /* COMENTARIOS*/
155
156  /* Comentario de Línea */
157  ( "#" ) {lexeme=yytext(); return Comentario;}
158
159  /* Inicio de Comentario Multilínea */
160  ( "/*" ) {lexeme=yytext(); return InicioComentario;}
161
162  /* Fin de Comentario Multilínea */
163  ( "*/" ) {lexeme=yytext(); return FinComentario;}
164
165  /* TIPO DE DATO*/
166
167  /* Tipo de Dato Entero */
168  ( "ent" ) {lexeme=yytext(); return Entero;}
169
170  /* Tipo de Dato Decimal */
171  ( "dec" ) {lexeme=yytext(); return Decimal;}
172
173  /* Valor Booleano Verdadero */
174  ( "v" ) {lexeme=yytext(); return Verdadero;}
175
```



```
173  /* Valor Booleano Verdadero */
174  ( "v" ) {lexeme=yytext(); return Verdadero;}
175
176  /* Valor Booleano Falso */
177  ( "f" ) {lexeme=yytext(); return Falso;}
178
179  /* Tipo de Dato Cadena */
180  ( "cadena" ) {lexeme=yytext(); return Cadena;}
181
182  /* ESTRUCTURAS DE CONTROL*/
183
184  /* Condicional Si */
185  ( "si" ) {lexeme=yytext(); return Si;}
186
187  /* Condicional SiNo */
188  ( "siNo" ) {lexeme=yytext(); return SiNo;}
189
190  /* Condicional SiNoSi */
191  ( "siNoSi" ) {lexeme=yytext(); return SiNoSi;}
192
193  /* Ciclo Repite */
194  ( "repite" ) {lexeme=yytext(); return CicloRepite;}
195
196  /* Ciclo Mientras */
197  ( "mientras" ) {lexeme=yytext(); return Mientras;}
198
199  /* Palabra Clave Hacer */
200  ( "hacer" ) {lexeme=yytext(); return Hacer;}
201
202  /* Seleccionador */
203  ( "seleccionar" ) {lexeme=yytext(); return Seleccionador;}
204
205  /* Caso */
206  ( "caso" ) {lexeme=yytext(); return Caso;}
```

```
208  /* Predeterminado */
209  ( "predeterminado" ) {lexeme=yytext(); return Predeterminado;}
210
211  /* Detener */
212  ( "detener" ) {lexeme=yytext(); return Detener;}
213
214  /* DECLARACIONES */
215
216  /* Clase*/
217  ( "Clase" ) {lexeme=yytext(); return Clase;}
218
219  /* Arreglo */
220  ( "arreglo" ) {lexeme=yytext(); return Arreglo;}
221
222  /* Función */
223  ( "fun" ) {lexeme=yytext(); return Funcion;}
224
225  /* Procedimiento */
226  ( "proc" ) {lexeme=yytext(); return Procedimiento;}
227
228  /* Principal */
229  ( "principal" ) {lexeme=yytext(); return Principal;}
230
231  /* Biblioteca */
232  ( "biblioteca" ) {lexeme=yytext(); return Biblioteca;}
233
234  /* ENTRADA Y SALIDA DE DATOS */
235
236  /* Imprimir */
237  ( "imprime" ) {lexeme=yytext(); return Imprimir;}
238
239  /* Leer */
240  ( "leer" ) {lexeme=yytext(); return Leer;}
241
242  /* Importar */
243  ( "importa" ) {lexeme=yytext(); return Importar;}
```



## Lexer.java

El archivo Lexer.java es una clase generada automáticamente por la herramienta JFlex, que tiene la función de construir un analizador léxico o escáner. Su tarea principal es identificar los diferentes tokens o componentes léxicos de un lenguaje, que están definidos en el archivo de especificación Lexer.flex. Estos tokens incluyen palabras clave, identificadores, operadores, literales y otros elementos importantes del lenguaje.

El analizador léxico es esencial en cualquier compilador o intérprete, ya que es el primer paso en el proceso de análisis del código fuente. El archivo Lexer.java se encarga de leer la secuencia de caracteres del código e identificar y clasificar estos tokens. Su propósito es garantizar que el código sea procesado de manera adecuada, y también detectar cualquier error léxico de forma temprana. Esto es fundamental para asegurar que el código se entienda correctamente y se ejecute sin problemas durante las etapas posteriores del análisis.

```
1  /* The following code was generated by JFlex 1.4.3 on 04/01/25, 04:08 */
2
3  package codigo;
4  import static codigo.Tokens.*;
5
6  /**
7   * This class is a scanner generated by
8   * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9   * on 04/01/25, 04:08 from the specification file
10  * <tt>C:/Users/vanes/OneDrive/Documentos/hh/VANE_HP/hp/Nueva carpeta/UNIVERSIDAD/SEMESTRE 5/LENGUAJES Y AUTOMATAS I/Proyecto/AnalizadorLexico/AnalizadorLexico/src/codigo/Lexer.
11  */
12  class Lexer {
13
14      /** This character denotes the end of file */
15      public static final int YYEOF = -1;
16
17      /** initial size of the lookahead buffer */
18      private static final int ZZ_BUFFER_SIZE = 16384;
19
20      /** lexical states */
21      public static final int YYINITIAL = 0;
22
23      /**
24       * ZZ_LEXSTATE[1] is the state in the DFA for the lexical state 1
25       * ZZ_LEXSTATE[1+1] is the state in the DFA for the lexical state 1
26       * at the beginning of a line
27       * 1 is of the form 1 = 2*k, k a non negative integer
28       */
29      private static final int ZZ_LEXSTATE[] = {
30          0, 0
31      };
32
33      /**
34       * Translates characters to character classes
35       */
36      private static final String ZZ_CMAP_PACKED =
```



```
36 private static final String ZS_CMAP_PACKED =
37     "111011311162101132210113112511531155"+
38     "1116611301126115411451146112211201131121"+
39     "1113511151121166110112411711231170110"+
40     "141111421111114311111144111114511371136"+
41     "12111140111111471101150113111111133"+
42     "1116411571115611101171111141142111111"+
43     "1116311511611162111132111211611651160"+
44     "1111312111341115111271152143101171135101167"+
45     "\uff40\0";
46
47 /**
48  * Translates characters to character classes
49  */
50 private static final char [] ZS_CMAP = zzUnpackCMap(ZS_CMAP_PACKED);
51
52 /**
53  * Translates DFA states to action switch labels.
54  */
55 private static final int [] ZS_ACTION = zzUnpackAction();
56
57 private static final String ZS_ACTION_PACKED_0 =
58     "11101111121131141212115152"+
59     "11161171110111111211311141115"+
60     "11161117112011211121122121123"+
61     "111241125112611271130113111321133"+
62     "11134121211351321136113711401141"+
63     "11121142115211431321144144145"+
64     "111461147111501151115211531154132"+
65     "111551121101721156112115711160"+
66     "11101121161172116211631164142"+
67     "1116511311211661102116711121170"+
68     "121211711521172117311211741175"+
69     "1721176121211771421110011101132"+
70     "\1102";
```

```
72 private static int [] zzUnpackAction() {
73     int [] result = new int[169];
74     int offset = 0;
75     offset = zzUnpackAction(ZS_ACTION_PACKED_0, offset, result);
76     return result;
77 }
78
79 private static int zzUnpackAction(String packed, int offset, int [] result) {
80     int i = 0; /* index in packed string */
81     int j = offset; /* index in unpacked array */
82     int l = packed.length();
83     while (i < l) {
84         int count = packed.charAt(i++);
85         int value = packed.charAt(i++);
86         do result[j++] = value; while (--count > 0);
87     }
88     return j;
89 }
90
91 /**
92  * Translates a state to a row index in the transition table
93  */
94 private static final int [] ZS_ROWMAP = zzUnpackRowMap();
95
96 private static final String ZS_ROWMAP_PACKED_0 =
97     "\0\0\0\72\0\164\0\256\0\350\0\0122\0\015c\0\164"+
98     "\0\0196\0\01d0\0\020a\0\0244\0\027e\0\02b8\0\02f2\0\72"+
99     "\0\72\0\032c\0\0366\0\03a0\0\03da\0\72\0\72\0\72"+
100     "\0\72\0\0414\0\72\0\044e\0\0488\0\164\0\04c2\0\72"+
101     "\0\72\0\72\0\72\0\72\0\72\0\72\0\72\0\04fc"+
102     "\0\0536\0\164\0\0570\0\05aa\0\05e4\0\72\0\72\0\72"+
103     "\0\72\0\061e\0\164\0\0658\0\0692\0\06cc\0\0706\0\0740"+
104     "\0\077a\0\07b4\0\07ee\0\0828\0\0862\0\72\0\72\0\72"+
105     "\0\72\0\72\0\72\0\72\0\72\0\089c\0\08d6"+
106     "\0\0910\0\164\0\094a\0\0984\0\09be\0\09f8\0\0a32\0\0a6c"+
```



```
private static int zzUnpackRowMap() {
    int [] result = new int[169];
    int offset = 0;
    offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);
    return result;
}

private static int zzUnpackRowMap(String packed, int offset, int [] result) {
    int i = 0; /* index in packed string */
    int j = offset; /* index in unpacked array */
    int l = packed.length();
    while (i < l) {
        int high = packed.charAt(i++) << 16;
        result[j++] = high | packed.charAt(i++);
    }
    return j;
}

/**
 * The transition table of the DFA
 */
private static final int [] ZZ_TRANS = zzUnpackTrans();
```

```
private static final String ZZ_TRANS_PACKED_0 =
    "\1\2\3\4\5\6\7\8\9\10\11\12\13\14\15\16\17\18\19\20\21\22\23\24\25\26\27\28\29\30\31\32\33\34\35\36\37\38\39\40\41\42\43\44\45\46\47\48\49\50\51\52\53\54\55\56\57\58\59\60\61\62\63\64\65\66\67\68\69\70\71\72\73\74\75\76\77\78\79\80\81\82\83\84\85\86\87\88\89\90\91\92\93\94\95\96\97\98\99\100\101\102\103\104\105\106\107\108\109\110\111\112\113\114\115\116\117\118\119\120\121\122\123\124\125\126\127\128\129\130\131\132\133\134\135\136\137\138\139\140\141\142\143\144\145\146\147\148\149\150\151\152\153\154\155\156\157\158\159\160\161\162\163\164\165\166\167\168\169\170\171\172\173\174\175\176\177\178\179\180\181\182\183\184\185\186\187\188\189\190\191\192\193\194\195\196\197\198\199\200\201\202\203\204\205\206\207\208\209\210\211\212\213\214\215\216\217\218\219\220\221\222\223\224\225\226\227\228\229\230\231\232\233\234\235\236\237\238\239\240\241\242\243\244\245\246\247\248\249\250\251\252\253\254\255\256\257\258\259\260\261\262\263\264\265\266\267\268\269\270\271\272\273\274\275\276\277\278\279\280\281\282\283\284\285\286\287\288\289\290\291\292\293\294\295\296\297\298\299\300\301\302\303\304\305\306\307\308\309\310\311\312\313\314\315\316\317\318\319\320\321\322\323\324\325\326\327\328\329\330\331\332\333\334\335\336\337\338\339\340\341\342\343\344\345\346\347\348\349\350\351\352\353\354\355\356\357\358\359\360\361\362\363\364\365\366\367\368\369\370\371\372\373\374\375\376\377\378\379\380\381\382\383\384\385\386\387\388\389\390\391\392\393\394\395\396\397\398\399\400\401\402\403\404\405\406\407\408\409\410\411\412\413\414\415\416\417\418\419\420\421\422\423\424\425\426\427\428\429\430\431\432\433\434\435\436\437\438\439\440\441\442\443\444\445\446\447\448\449\450\451\452\453\454\455\456\457\458\459\460\461\462\463\464\465\466\467\468\469\470\471\472\473\474\475\476\477\478\479\480\481\482\483\484\485\486\487\488\489\490\491\492\493\494\495\496\497\498\499\500\501\502\503\504\505\506\507\508\509\510\511\512\513\514\515\516\517\518\519\520\521\522\523\524\525\526\527\528\529\530\531\532\533\534\535\536\537\538\539\540\541\542\543\544\545\546\547\548\549\550\551\552\553\554\555\556\557\558\559\560\561\562\563\564\565\566\567\568\569\570\571\572\573\574\575\576\577\578\579\580\581\582\583\584\585\586\587\588\589\590\591\592\593\594\595\596\597\598\599\600\601\602\603\604\605\606\607\608\609\610\611\612\613\614\615\616\617\618\619\620\621\622\623\624\625\626\627\628\629\630\631\632\633\634\635\636\637\638\639\640\641\642\643\644\645\646\647\648\649\650\651\652\653\654\655\656\657\658\659\660\661\662\663\664\665\666\667\668\669\670\671\672\673\674\675\676\677\678\679\680\681\682\683\684\685\686\687\688\689\690\691\692\693\694\695\696\697\698\699\700\701\702\703\704\705\706\707\708\709\710\711\712\713\714\715\716\717\718\719\720\721\722\723\724\725\726\727\728\729\730\731\732\733\734\735\736\737\738\739\740\741\742\743\744\745\746\747\748\749\750\751\752\753\754\755\756\757\758\759\760\761\762\763\764\765\766\767\768\769\770\771\772\773\774\775\776\777\778\779\780\781\782\783\784\785\786\787\788\789\790\791\792\793\794\795\796\797\798\799\800\801\802\803\804\805\806\807\808\809\810\811\812\813\814\815\816\817\818\819\820\821\822\823\824\825\826\827\828\829\830\831\832\833\834\835\836\837\838\839\840\841\842\843\844\845\846\847\848\849\850\851\852\853\854\855\856\857\858\859\860\861\862\863\864\865\866\867\868\869\870\871\872\873\874\875\876\877\878\879\880\881\882\883\884\885\886\887\888\889\890\891\892\893\894\895\896\897\898\899\900\901\902\903\904\905\906\907\908\909\910\911\912\913\914\915\916\917\918\919\920\921\922\923\924\925\926\927\928\929\930\931\932\933\934\935\936\937\938\939\940\941\942\943\944\945\946\947\948\949\950\951\952\953\954\955\956\957\958\959\960\961\962\963\964\965\966\967\968\969\970\971\972\973\974\975\976\977\978\979\980\981\982\983\984\985\986\987\988\989\990\991\992\993\994\995\996\997\998\999\1000\1001\1002\1003\1004\1005\1006\1007\1008\1009\1010\1011\1012\1013\1014\1015\1016\1017\1018\1019\1020\1021\1022\1023\1024\1025\1026\1027\1028\1029\1030\1031\1032\1033\1034\1035\1036\1037\10
```





|     |   |  |
|-----|---|--|
| 178 | "\7\3\1\1\0\1\0\3\5\0\2\3\1\0\1\121\10\3"+                |  |
| 179 | "\1\5\0\3\3\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3"+               |  |
| 180 | "\1\1\0\2\3\1\1\6\3\6\3\1\5\0\3\3\1\0\1\7\3"+             |  |
| 181 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\3\3"+             |  |
| 182 | "\1\1\0\1\7\3\1\1\0\1\4\3\1\122\3\3\3\5\0\2\3"+           |  |
| 183 | "\1\1\0\1\3\1\123\3\3\1\5\0\3\3\1\0\1\7\3"+               |  |
| 184 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\2\3\1\124\1\3"+             |  |
| 185 | "\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3"+             |  |
| 186 | "\1\1\0\1\3\1\125\2\3\1\5\0\3\3\1\0\1\7\3"+               |  |
| 187 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\4\3\1\126\1\3"+             |  |
| 188 | "\1\5\0\3\3\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3"+               |  |
| 189 | "\1\1\0\1\1\3\1\5\0\3\3\1\1\0\1\3\1\127\1\3"+             |  |
| 190 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\5\1\130\1\3"+               |  |
| 191 | "\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3"+             |  |
| 192 | "\1\1\0\1\131\1\0\3\1\5\0\3\3\1\0\1\7\3\1\1\0"+           |  |
| 193 | "\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\3\3\1\0"+               |  |
| 194 | "\1\7\3\1\1\0\1\3\1\132\1\6\3\4\0\1\6\7\5\1\1\0"+         |  |
| 195 | "\1\5\1\7\5\1\1\0\2\3\1\1\0\2\3\1\133\1\6\3\1\5\1\0"+     |  |
| 196 | "\1\3\3\1\0\1\7\3\1\1\0\1\4\3\1\134\1\3\3\5\1\0"+         |  |
| 197 | "\1\2\3\1\1\0\1\135\1\0\3\1\5\0\3\3\1\0\1\7\3"+           |  |
| 198 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\3\3"+             |  |
| 199 | "\1\1\0\2\3\1\136\1\4\3\1\1\0\1\0\3\5\0\2\3"+             |  |
| 200 | "\1\1\0\1\1\3\1\5\0\3\3\1\1\0\1\5\3\1\137\1\3"+           |  |
| 201 | "\1\1\0\1\0\3\5\0\2\3\1\140\1\7\0\1\2\3\1\1\0\2\3"+       |  |
| 202 | "\1\141\1\6\3\1\5\0\3\3\1\0\1\7\3\1\1\0\1\3"+             |  |
| 203 | "\1\142\1\6\3\5\0\2\3\1\1\0\1\6\3\1\143\1\2\3"+           |  |
| 204 | "\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3"+             |  |
| 205 | "\1\1\0\1\144\1\3\3\1\145\1\4\3\1\5\0\3\3\1\1\0"+         |  |
| 206 | "\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\4\3\1\146"+           |  |
| 207 | "\1\4\3\1\5\0\3\3\1\0\1\7\3\1\1\0\1\0\3\5\1\0"+           |  |
| 208 | "\1\2\3\1\1\0\1\1\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0"+         |  |
| 209 | "\1\6\3\1\147\1\1\3\5\0\2\3\1\1\0\1\1\3\1\5\1\0"+         |  |
| 210 | "\1\1\5\0\2\3\1\1\0\1\7\3\1\1\0\1\0\3\3\1\151\1\3\1\0"+   |  |
| 211 | "\1\5\0\2\3\1\1\0\1\1\3\1\5\0\3\3\1\0\1\7\3"+             |  |
| 212 | "\1\1\0\1\3\1\152\1\4\3\5\0\2\3\1\1\0\1\4\3"+             |  |
| 213 | "\1\1\6\3\1\4\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3"+       |  |
| 214 | "\1\5\0\2\3\1\1\0\1\1\3\1\5\0\1\153\1\2\3\1\1\0"+         |  |
| 215 | "\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\1\0"+         |  |
| 216 | "\1\3\3\1\1\0\1\7\3\1\1\0\1\3\3\1\154\1\4\3\5\1\0"+       |  |
| 217 | "\1\2\3\1\1\0\1\4\3\1\155\1\4\3\1\5\0\3\3\1\1\0"+         |  |
| 218 | "\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\5\3\1\125"+           |  |
| 219 | "\1\3\3\1\5\0\3\3\1\0\1\7\3\1\1\0\1\0\3\5\1\0"+           |  |
| 220 | "\1\2\3\1\1\0\1\4\3\1\156\1\4\3\1\5\0\3\3\1\1\0"+         |  |
| 221 | "\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\1\0"+         |  |
| 222 | "\1\3\3\1\1\0\1\7\3\1\1\0\1\3\3\1\157\1\4\3\5\1\0"+       |  |
| 223 | "\1\2\3\1\1\0\1\160\1\0\3\1\5\0\3\3\1\0\1\7\3"+           |  |
| 224 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\2\3\1\0"+         |  |
| 225 | "\1\161\1\1\0\1\7\3\1\1\0\1\0\1\3\1\6\0\1\140\1\4\3\1\0"+ |  |
| 226 | "\1\162\1\2\1\0\2\3\1\1\0\1\4\3\1\163\1\4\3\1\5\1\0"+     |  |
| 227 | "\1\3\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0"+           |  |
| 228 | "\1\1\1\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\3\3\1\164"+       |  |
| 229 | "\1\4\3\5\0\2\3\1\1\0\1\1\3\1\165\1\7\3\1\5\1\0"+         |  |
| 230 | "\1\3\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0"+           |  |
| 231 | "\1\1\1\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\166\1\7\3"+       |  |
| 232 | "\1\5\0\2\3\1\1\0\1\3\1\167\1\3\1\5\0\3\3\1\0"+           |  |
| 233 | "\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\5\1\3"+         |  |
| 234 | "\1\1\7\0\3\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3"+         |  |
| 235 | "\1\5\0\2\3\1\1\0\1\171\1\0\3\1\5\0\3\3\1\1\0"+           |  |
| 236 | "\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\1\0"+         |  |
| 237 | "\1\172\1\2\3\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\0"+       |  |
| 238 | "\1\1\0\1\1\3\1\5\0\3\3\1\1\0\1\2\3\1\173\1\4\3"+         |  |
| 239 | "\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\3\3\1\0"+         |  |
| 240 | "\1\1\0\1\7\3\1\1\0\1\1\3\1\174\1\6\3\5\0\2\3\1\0"+       |  |
| 241 | "\1\1\0\1\1\3\1\5\0\3\1\175\1\2\3\1\1\0\1\7\3\1\1\1\0"+   |  |
| 242 | "\1\1\0\3\5\0\2\3\1\1\0\1\1\3\1\5\0\1\176\1\2\3\1\0"+     |  |
| 243 | "\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\2\3\1\0"+       |  |
| 244 | "\1\177\1\6\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\3\1\0"+     |  |
| 245 | "\1\5\0\2\3\1\1\0\1\1\3\1\200\1\7\3\1\5\0\3\3\1\0"+       |  |
| 246 | "\1\1\0\1\7\3\1\1\0\1\0\3\5\0\2\3\1\1\0\1\1\3\1\0"+       |  |



|     |  |  |
|-----|--|--|
| 247 | "\150\133\101\73\110\113\1\201\6\3"+       |  |
| 248 | "\50\23\101\43\1\202\43\150\3\3"+          |  |
| 249 | "\1\01\73\110\101\3\50\23\110\2\3"+        |  |
| 250 | "\1\203\63\150\3\3\101\73\110\10\3"+       |  |
| 251 | "\50\23\101\1\204\10\3\150\3\3\10"+        |  |
| 252 | "\73\110\101\3\50\12\3\110\11\3\150"+      |  |
| 253 | "\3\3\101\73\110\1\3\1\205\2\3\50"+        |  |
| 254 | "\2\3\110\2\3\1\206\63\150\3\3\10"+        |  |
| 255 | "\73\110\101\3\50\12\3\110\1\207\10\3"+    |  |
| 256 | "\150\13\101\73\110\101\3\50\2\3"+         |  |
| 257 | "\1\01\13\150\3\3\1\01\73\110\1\3"+        |  |
| 258 | "\1\210\63\50\2\3\1\01\13\1\211\7\3"+      |  |
| 259 | "\150\13\101\73\110\101\3\50\2\3"+         |  |
| 260 | "\1\01\4\3\1\212\43\150\3\3\110\7\3"+      |  |
| 261 | "\1\01\101\3\50\12\3\1\01\4\3\1\213\4\3"+  |  |
| 262 | "\150\13\101\73\110\101\3\50\2\3"+         |  |
| 263 | "\1\01\1\214\10\3\150\3\3\1\01\73\110\10"+ |  |
| 264 | "\10\3\50\2\3\1\01\2\3\1\215\63\150\10"+   |  |
| 265 | "\3\3\101\73\110\101\3\50\2\3\10"+         |  |
| 266 | "\1\13\150\1\216\2\3\1\01\73\110\10\3"+    |  |
| 267 | "\50\23\101\13\150\3\3\1\01\7\3"+          |  |
| 268 | "\1\01\3\1\217\43\50\2\3\110\1\3"+         |  |
| 269 | "\1\220\43\150\3\3\101\73\110\10\3"+       |  |
| 270 | "\50\23\101\13\150\1\3\1\221\1\3"+         |  |
| 271 | "\1\01\73\110\101\3\50\23\10\1\222"+       |  |
| 272 | "\10\3\150\3\3\110\1\73\110\10\3\50\10"+   |  |
| 273 | "\2\3\110\113\150\1\3\1\223\1\3\110\10"+   |  |
| 274 | "\73\110\101\3\50\2\3\110\11\3\150\10"+    |  |
| 275 | "\1\224\2\3\110\1\73\110\10\3\50\2\3"+     |  |
| 276 | "\1\01\13\150\3\3\110\1\73\110\1\3"+       |  |
| 277 | "\1\225\3\3\50\2\3\110\1\3\1\226\4\3"+     |  |
| 278 | "\150\13\101\73\110\101\3\50\2\3"+         |  |
| 279 | "\1\01\13\150\1\3\1\227\1\3\101\7\3"+      |  |
| 280 | "\1\10\101\3\50\2\3\110\2\3\1\230\6\3"+    |  |
| 281 | "\150\13\101\73\110\101\3\50\2\3"+         |  |
| 282 | "\1\01\13\150\3\3\110\1\73\110\10\3\3"+    |  |

|     |   |  |
|-----|---|--|
| 283 | "\1\231\43\50\2\3\110\11\3\150\1\3"+  |  |
| 284 | "\1\232\1\3\110\1\73\110\101\3\50\2\3"+   |  |
| 285 | "\1\01\13\150\1\233\2\3\110\1\73\110\10"+   |  |
| 286 | "\10\3\50\2\3\110\6\3\1\234\2\3\150\10"+  |  |
| 287 | "\3\3\101\73\110\101\3\50\2\3\10"+  |  |
| 288 | "\4\3\1\235\43\150\1\3\110\1\73\110\10"+  |  |
| 289 | "\10\3\50\2\3\110\1\3\1\236\7\3\150\10"+  |  |
| 290 | "\3\3\110\1\73\110\101\3\50\2\3\110\10"+  |  |
| 291 | "\5\3\1\237\3\150\10\3\3\101\7\3\110\10"+   |  |
| 292 | "\10\3\50\2\3\110\11\3\150\1\3\110\10"+   |  |
| 293 | "\73\110\1\3\1\240\2\3\150\2\3\110\10"+   |  |
| 294 | "\11\3\150\1\3\110\1\73\110\11\3\1\241"+  |  |
| 295 | "\6\3\50\2\3\110\11\3\150\1\3\1\242"+   |  |
| 296 | "\1\3\110\1\73\110\101\3\50\2\3\110\10"+  |  |
| 297 | "\1\243\10\3\150\3\3\110\1\73\110\10\3\3"+  |  |
| 298 | "\50\2\3\110\11\3\150\1\3\1\244\1\3\10"+  |  |
| 299 | "\1\01\73\110\101\3\50\2\3\110\11\3"+   |  |
| 300 | "\150\1\245\2\3\110\1\73\110\101\3\50\10"+  |  |
| 301 | "\2\3\110\1\3\1\246\7\3\150\1\3\110\10"+  |  |
| 302 | "\73\110\101\3\50\2\3\110\11\3\150\10"+   |  |
| 303 | "\1\3\1\247\1\3\110\1\73\110\101\3\50\10"+  |  |
| 304 | "\2\3\110\11\3\150\1\3\110\1\73\110\10"+  |  |
| 305 | "\1\250\1\73\1\50\2\3\110\11\3\150\1\3\3"+  |  |
| 306 | "\1\01\73\110\1\3\1\251\4\3\4\0";   |  |
| 307 |   |  |
| 308 | <pre>private static int [] zzUnpackTrans() {</pre>                                      |  |
| 309 | <pre>    int [] result = new int[6670];</pre>   |  |
| 310 | <pre>    int offset = 0;</pre>  |  |
| 311 | <pre>    offset = zzUnpackTrans(ZZ_TRANS_PACKED_0, offset, result);</pre>               |  |
| 312 | <pre>    return result;</pre>   |  |
| 313 | <pre>}</pre>  |  |
| 314 |   |  |
| 315 | <pre>private static int zzUnpackTrans(String packed, int offset, int [] result) {</pre> |  |
| 316 | <pre>    int i = 0; /* index in packed string */</pre>                                  |  |
| 317 | <pre>    int j = offset; /* index in unpacked array */</pre>                            |  |



```

351 private static int [] zzUnpackAttribute() {
352     int [] result = new int[169];
353     int offset = 0;
354     offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
355     return result;
356 }
357
358 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
359     int i = 0; /* index in packed string */
360     int j = offset; /* index in unpacked array */
361     int l = packed.length();
362     while (i < l) {
363         int count = packed.charAt(i++);
364         int value = packed.charAt(i++);
365         do result[j++] = value; while (--count > 0);
366     }
367     return j;
368 }
369
370 /** the input device */
371 private java.io.Reader zzReader;
372
373 /** the current state of the DFA */
374 private int zzState;
375
376 /** the current lexical state */
377 private int zzLexicalState = YYINITIAL;
378
379 /** this buffer contains the current text to be matched and is
380  * | the source of the yytext() string */
381 private char zzBuffer[] = new char[ZZ_BUFFERSIZE];
382
383 /** the textposition at the last accepting state */
384 private int zzMarkedPos;

```



```
385
386 /** the current text position in the buffer */
387 private int zzCurrentPos;
388
389 /** startRead marks the beginning of the yytext() string in the buffer */
390 private int zzStartRead;
391
392 /** endRead marks the last character in the buffer, that has been read
393 | from input */
394 private int zzEndRead;
395
396 /** number of newlines encountered up to the start of the matched text */
397 private int yyline;
398
399 /** the number of characters up to the start of the matched text */
400 private int yychar;
401
402 /**
403 * the number of characters from the last newline up to the start of the
404 * matched text
405 */
406 private int yycolumn;
407
408 /**
409 * zzAtBOL == true <=> the scanner is currently at the beginning of a line
410 */
411 private boolean zzAtBOL = true;
412
413 /** zzAtEOF == true <=> the scanner is at the EOF */
414 private boolean zzAtEOF;
415
416 /** denotes if the user-EOF-code has already been executed */
417 private boolean zzEOFDone;
```

```
416 /** denotes if the user-EOF-code has already been executed */
417 private boolean zzEOFDone;
418
419 /* user code: */
420 public String lexeme;
421
422
423 /**
424 * Creates a new scanner
425 * There is also a java.io.InputStream version of this constructor.
426 *
427 * @param in the java.io.Reader to read input from.
428 */
429 Lexer(java.io.Reader in) {
430     this.zzReader = in;
431 }
432
433 /**
434 * Creates a new scanner.
435 * There is also java.io.Reader version of this constructor.
436 *
437 * @param in the java.io.InputStream to read input from.
438 */
439 Lexer(java.io.InputStream in) {
440     this(new java.io.InputStreamReader(in));
441 }
442
443 /**
444 * Unpacks the compressed character translation table.
445 *
446 * @param packed the packed character translation table
447 * @return the unpacked character translation table
448 */
449 private static char [] zzUnpackCMap(String packed) {
450     char [] map = new char[0x10000];
```



```
449 private static char [] zzUnpackMap(String packed) {
450     char [] map = new char[0x10000];
451     int i = 0; /* index in packed string */
452     int j = 0; /* index in unpacked array */
453     while (i < 162) {
454         int count = packed.charAt(i++);
455         char value = packed.charAt(i++);
456         do map[j++] = value; while (--count > 0);
457     }
458     return map;
459 }
460
461 /**
462  * Refills the input buffer.
463  *
464  * @return <code>false</code>, iff there was new input.
465  *
466  * @exception java.io.IOException if any I/O-Error occurs
467  */
468 private boolean zzRefill() throws java.io.IOException {
469     /* first: make room (if you can) */
470     if (zzStartRead > 0) {
471         System.arraycopy(zzBuffer, zzStartRead,
472             zzBuffer, 0,
473             zzEndRead-zzStartRead);
474
475         /* translate stored positions */
476         zzEndRead = zzStartRead;
477         zzCurrentPos = zzStartRead;
478         zzMarkedPos = zzStartRead;
479         zzStartRead = 0;
480     }
481 }
482
```

```
484 /* is the buffer big enough? */
485 if (zzCurrentPos >= zzBuffer.length) {
486     /* if not: blow it up */
487     char newBuffer[] = new char[zzCurrentPos*2];
488     System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
489     zzBuffer = newBuffer;
490 }
491
492 /* finally: fill the buffer with new input */
493 int numRead = zzReader.read(zzBuffer, zzEndRead,
494     zzBuffer.length-zzEndRead);
495
496 if (numRead > 0) {
497     zzEndRead += numRead;
498     return false;
499 }
500 // unlikely but not impossible: read 0 characters, but not at end of stream
501 if (numRead == 0) {
502     int c = zzReader.read();
503     if (c == -1) {
504         return true;
505     } else {
506         zzBuffer[zzEndRead++] = (char) c;
507         return false;
508     }
509 }
510 // numRead < 0
511 return true;
512 }
513
514 /**
515  * Closes the input stream.
516  */
517
518
```



```
519 public final void yyclose() throws java.io.IOException {
520     zzAtEOF = true; /* indicate end of file */
521     zzEndRead = zzStartRead; /* invalidate buffer */
522
523     if (zzReader != null)
524         zzReader.close();
525 }
526
527 /**
528  * Resets the scanner to read from a new input stream.
529  * Does not close the old reader.
530  *
531  * All internal variables are reset, the old input stream
532  * <b>cannot</b> be reused (internal buffer is discarded and lost).
533  * Lexical state is set to <tt>2Z_INITIAL</tt>.
534  *
535  * @param reader the new input stream
536  */
537 public final void yyreset(java.io.Reader reader) {
538     zzReader = reader;
539     zzAtBOL = true;
540     zzAtEOF = false;
541     zzEOFDone = false;
542     zzEndRead = zzStartRead = 0;
543     zzCurrentPos = zzMarkedPos = 0;
544     yyline = yychar = yycolumn = 0;
545     zzLexicalState = YYINITIAL;
546 }
547
548 /**
549  * Returns the current lexical state.
550  */
551 public final int yystate() {
552     return zzLexicalState;
553 }
```

```
558 /**
559  * Enters a new lexical state
560  *
561  * @param newState the new lexical state
562  */
563 public final void yybegin(int newState) {
564     zzLexicalState = newState;
565 }
566
567 /**
568  * Returns the text matched by the current regular expression.
569  */
570 public final String yytext() {
571     return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
572 }
573
574 /**
575  * Returns the character at position <tt>pos</tt> from the
576  * matched text.
577  *
578  * It is equivalent to yytext().charAt(pos), but faster
579  *
580  * @param pos the position of the character to fetch.
581  * A value from 0 to yylength()-1.
582  *
583  * @return the character at position pos
584  */
585 public final char yycharat(int pos) {
586     return zzBuffer[zzStartRead+pos];
587 }
588
589 }
```



```
600  /**
601   * Reports an error that occurred while scanning.
602   *
603   * In a wellformed scanner (no or only correct usage of
604   * yypushback(int) and a match-all fallback rule) this method
605   * will only be called with things that "Can't Possibly Happen".
606   * If this method is called, something is seriously wrong
607   * (e.g. a JFlex bug producing a faulty scanner etc.).
608   *
609   * Usual syntax/scanner level error handling should be done
610   * in error fallback rules.
611   *
612   * @param   errorCode   the code of the error message to display
613   */
614  private void zzScanError(int errorCode) {
615      String message;
616      try {
617          message = ZZ_ERROR_MSG[errorCode];
618      }
619      catch (ArrayIndexOutOfBoundsException e) {
620          message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
621      }
622
623      throw new Error(message);
624  }
625
626
627  /**
628   * Pushes the specified amount of characters back into the input stream.
629   *
630   * They will be read again by then next call of the scanning method
631   *
632   * @param   number       the number of characters to be read again.
633   *                       This number must not be greater than yylength()!
634   */
```

```
635  public void yypushback(int number) {
636      if ( number > yylength() )
637          zzScanError(ZZ_PUSHBACK_2BIG);
638
639      zzMarkedPos -= number;
640  }
641
642
643  /**
644   * Resumes scanning until the next regular expression is matched,
645   * the end of input is encountered or an I/O-Error occurs.
646   *
647   * @return   the next token
648   * @exception java.io.IOException if any I/O-Error occurs
649   */
650  public Tokens yylex() throws java.io.IOException {
651      int zzInput;
652      int zzAction;
653
654      // cached fields:
655      int zzCurrentPosL;
656      int zzMarkedPosL;
657      int zzEndReadL = zzEndRead;
658      char [] zzBufferL = zzBuffer;
659      char [] zzCMapL = ZZ_CMAP;
660
661      int [] zzTransL = ZZ_TRANS;
662      int [] zzRowMapL = ZZ_ROWMAP;
663      int [] zzAttrL = ZZ_ATTRIBUTE;
664
665      while (true) {
666          zzMarkedPosL = zzMarkedPos;
667
668          zzAction = -1;
669
670          zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
```



```
672 |         zzState = ZZ_LEXSTATE[zzLexicalState];
673 |
674 |
675 |         zzForAction: {
676 |             while (true) {
677 |
678 |                 if (zzCurrentPosL < zzEndReadL)
679 |                     zzInput = zzBufferL[zzCurrentPosL++];
680 |             else if (zzAtEOF) {
681 |                 zzInput = YYEOF;
682 |                 break zzForAction;
683 |             }
684 |             else {
685 |                 // store back cached positions
686 |                 zzCurrentPos = zzCurrentPosL;
687 |                 zzMarkedPos = zzMarkedPosL;
688 |                 boolean eof = zzRefill();
689 |                 // get translated positions and possibly new buffer
690 |                 zzCurrentPosL = zzCurrentPos;
691 |                 zzMarkedPosL = zzMarkedPos;
692 |                 zzBufferL = zzBuffer;
693 |                 zzEndReadL = zzEndRead;
694 |                 if (eof) {
695 |                     zzInput = YYEOF;
696 |                     break zzForAction;
697 |                 }
698 |                 else {
699 |                     zzInput = zzBufferL[zzCurrentPosL++];
700 |                 }
701 |             }
702 |             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
703 |             if (zzNext == -1) break zzForAction;
704 |             zzState = zzNext;
705 |
706 |             int zzAttributes = zzAttrL[zzState];
```

```
706 |         int zzAttributes = zzAttrL[zzState];
707 |         if ( (zzAttributes & 1) == 1 ) {
708 |             zzAction = zzState;
709 |             zzMarkedPosL = zzCurrentPosL;
710 |             if ( (zzAttributes & 8) == 8 ) break zzForAction;
711 |         }
712 |     }
713 |
714 | }
715 |
716 | // store back cached position
717 | zzMarkedPos = zzMarkedPosL;
718 |
719 | switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
720 |     case 2:
721 |         { lexeme=yytext(); return Identificador;
722 |         }
723 |     case 67: break;
724 |     case 13:
725 |         { return No;
726 |         }
727 |     case 68: break;
728 |     case 24:
729 |         { return LlaveApertura;
730 |         }
731 |     case 69: break;
732 |     case 38:
733 |         { return Asignacion;
734 |         }
735 |     case 70: break;
736 |     case 12:
737 |         { return Menor;
738 |         }
739 |     case 71: break;
740 |     case 28:
741 |         { return Comentario;
```





|     |                                       |  |
|-----|---------------------------------------|--|
| 743 | case 72: break;                       |  |
| 744 | case 36:                              |  |
| 745 | { return InicioComentario;            |  |
| 746 | }                                     |  |
| 747 | case 73: break;                       |  |
| 748 | case 47:                              |  |
| 749 | { return Positivo;                    |  |
| 750 | }                                     |  |
| 751 | case 74: break;                       |  |
| 752 | case 20:                              |  |
| 753 | { return ParentesisApertura;          |  |
| 754 | }                                     |  |
| 755 | case 75: break;                       |  |
| 756 | case 46:                              |  |
| 757 | { return Entero;                      |  |
| 758 | }                                     |  |
| 759 | case 76: break;                       |  |
| 760 | case 57:                              |  |
| 761 | { return CicloRepite;                 |  |
| 762 | }                                     |  |
| 763 | case 77: break;                       |  |
| 764 | case 63:                              |  |
| 765 | { return Principal;                   |  |
| 766 | }                                     |  |
| 767 | case 78: break;                       |  |
| 768 | case 55:                              |  |
| 769 | { return Hacer;                       |  |
| 770 | }                                     |  |
| 771 | case 79: break;                       |  |
| 772 | case 5:                               |  |
| 773 | { return Falso;                       |  |
| 774 | }                                     |  |
| 775 | case 80: break;                       |  |
| 776 | case 51:                              |  |
| 777 | { return Detener;                     |  |
| 778 | }                                     |  |
| 779 | case 81: break;                       |  |
| 780 | case 65:                              |  |
| 781 | { return Seleccionador;               |  |
| 782 | }                                     |  |
| 783 | case 82: break;                       |  |
| 784 | case 34:                              |  |
| 785 | { lexeme=yytext(); return Reservadas; |  |
| 786 | }                                     |  |
| 787 | case 83: break;                       |  |
| 788 | case 17:                              |  |
| 789 | { return Potencia;                    |  |
| 790 | }                                     |  |
| 791 | case 84: break;                       |  |
| 792 | case 48:                              |  |
| 793 | { return Negativo;                    |  |
| 794 | }                                     |  |
| 795 | case 85: break;                       |  |
| 796 | case 60:                              |  |
| 797 | { return Retornar;                    |  |
| 798 | }                                     |  |
| 799 | case 86: break;                       |  |
| 800 | case 16:                              |  |
| 801 | { return Modulo;                      |  |
| 802 | }                                     |  |
| 803 | case 87: break;                       |  |
| 804 | case 3:                               |  |
| 805 | { lexeme=yytext(); return Numero;     |  |
| 806 | }                                     |  |
| 807 | case 88: break;                       |  |
| 808 | case 64:                              |  |
| 809 | { return Biblioteca;                  |  |
| 810 | }                                     |  |
| 811 | case 89: break;                       |  |
| 812 | case 61:                              |  |
| 813 | { return Detener;                     |  |
| 814 | }                                     |  |



|     |                            |  |
|-----|----------------------------|--|
| 816 | case 35:                   |  |
| 817 | { return Si;               |  |
| 818 | }                          |  |
| 819 | case 91: break;            |  |
| 820 | case 25:                   |  |
| 821 | { return LlaveCierre;      |  |
| 822 | }                          |  |
| 823 | case 92: break;            |  |
| 824 | case 54:                   |  |
| 825 | { return Caso;             |  |
| 826 | }                          |  |
| 827 | case 93: break;            |  |
| 828 | case 50:                   |  |
| 829 | { return Mulo;             |  |
| 830 | }                          |  |
| 831 | case 94: break;            |  |
| 832 | case 44:                   |  |
| 833 | { return Distinto;         |  |
| 834 | }                          |  |
| 835 | case 95: break;            |  |
| 836 | case 26:                   |  |
| 837 | { return ComillaDoble;     |  |
| 838 | }                          |  |
| 839 | case 96: break;            |  |
| 840 | case 33:                   |  |
| 841 | { return AdmiracionInicio; |  |
| 842 | }                          |  |
| 843 | case 97: break;            |  |
| 844 | case 29:                   |  |
| 845 | { return Verdadero;        |  |
| 846 | }                          |  |
| 847 | case 98: break;            |  |
| 848 | case 27:                   |  |
| 849 | { return ComillaSimple;    |  |
| 850 | }                          |  |
| 851 | case 99: break;            |  |

|     |                            |  |
|-----|----------------------------|--|
| 848 | case 27:                   |  |
| 849 | { return ComillaSimple;    |  |
| 850 | }                          |  |
| 851 | case 99: break;            |  |
| 852 | case 8:                    |  |
| 853 | { return Suma;             |  |
| 854 | }                          |  |
| 855 | case 100: break;           |  |
| 856 | case 49:                   |  |
| 857 | { return Decimal;          |  |
| 858 | }                          |  |
| 859 | case 101: break;           |  |
| 860 | case 23:                   |  |
| 861 | { return CorcheteCierre;   |  |
| 862 | }                          |  |
| 863 | case 102: break;           |  |
| 864 | case 59:                   |  |
| 865 | { return Importar;         |  |
| 866 | }                          |  |
| 867 | case 103: break;           |  |
| 868 | case 66:                   |  |
| 869 | { return Predeterminado;   |  |
| 870 | }                          |  |
| 871 | case 104: break;           |  |
| 872 | case 32:                   |  |
| 873 | { return InterrogacionFin; |  |
| 874 | }                          |  |
| 875 | case 105: break;           |  |
| 876 | case 52:                   |  |
| 877 | { return SiNo;             |  |
| 878 | }                          |  |
| 879 | case 106: break;           |  |
| 880 | case 18:                   |  |
| 881 | { return Punto;            |  |
| 882 | }                          |  |
| 883 | case 107: break;           |  |



```
883 | case 107: break;
884 | case 56:
885 |     { return SiNoSi;
886 |     }
887 | case 108: break;
888 | case 31:
889 |     { return InterrogacionInicio;
890 |     }
891 | case 109: break;
892 | case 9:
893 |     { return Resta;
894 |     }
895 | case 110: break;
896 | case 22:
897 |     { return CorcheteApertura;
898 |     }
899 | case 111: break;
900 | case 11:
901 |     { return Mayor;
902 |     }
903 | case 112: break;
904 | case 14:
905 |     { return Y;
906 |     }
907 | case 113: break;
908 | case 21:
909 |     { return ParentesisCierre;
910 |     }
911 | case 114: break;
912 | case 30:
913 |     { return CaracteresEspeciales;
914 |     }
915 | case 115: break;
916 | case 62:
917 |     { return Mientras;
```

```
919 | case 116: break;
920 | case 45:
921 |     { return Pi;
922 |     }
923 | case 117: break;
924 | case 40:
925 |     { return MayorIgual;
926 |     }
927 | case 118: break;
928 | case 37:
929 |     { return Comparacion;
930 |     }
931 | case 119: break;
932 | case 43:
933 |     { return InicioTexto;
934 |     }
935 | case 120: break;
936 | case 39:
937 |     { return FinComentario;
938 |     }
939 | case 121: break;
940 | case 58:
941 |     { return Imprimir;
942 |     }
943 | case 122: break;
944 | case 1:
945 |     { return ERROR;
946 |     }
947 | case 123: break;
948 | case 41:
949 |     { return FinalTexto;
950 |     }
951 | case 124: break;
952 | case 10:
953 |     { return Multiplicacion;
954 |     }
```



```
955     case 125: break;
956     case 4:
957     { /*Ignore*/
958     }
959     case 126: break;
960     case 42:
961     { return MenorIgual;
962     }
963     case 127: break;
964     case 53:
965     { return Raiz;
966     }
967     case 128: break;
968     case 19:
969     { return Euler;
970     }
971     case 129: break;
972     case 7:
973     { return Igual;
974     }
975     case 130: break;
976     case 6:
977     { return Division;
978     }
979     case 131: break;
980     case 15:
981     { return O;
982     }
983     case 132: break;
984     default:
985     if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
986         zzAtEOF = true;
987         return null;
988     }
989     else {
990         zzScanError(zz_no_match);
991     }
```

```
989     else {
990         zzScanError(zz_no_match);
991     }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
```

## Tokens.java

El archivo Tokens.java define los distintos tipos de tokens que el analizador léxico utilizará para clasificar las unidades básicas del código fuente en tu proyecto. Cada token representa elementos esenciales del lenguaje como operadores, palabras reservadas, tipos de datos y símbolos especiales. Su propósito es proporcionar una estructura organizada que permita al analizador léxico identificar y procesar correctamente el código fuente, facilitando su conversión en unidades comprensibles para el análisis sintáctico posterior. Además, el token ERROR ayuda a detectar entradas no válidas, mejorando la precisión y eficiencia del proceso.

```
6 package codigo;
7
8
9 public enum Tokens {
10     Linea,
11     Suma,
12     Resta,
13     Multiplicacion,
14     Division,
15     Mayor,
16     Menor,
17     MayorIgual,
18     MenorIgual,
19     Comparacion,
20     Distinto,
21     Y,
22     O,
23     No,
24     Modulo,
25     Potencia,
26     Raiz,
27     Punto,
28     Asignacion,
29     Igual,
30     Concatenacion,
31     Positivo,
32     Negativo,
33     Incremento,
34     Decremento,
35     Pi,
36     Euler,
37     ParentesisApertura,
38     ParentesisCierre,
39     CorcheteApertura,
40     CorcheteCierre,
41     LlaveApertura,
42     LlaveCierre,
```

|    |                          |  |
|----|--------------------------|--|
| 43 | <i>InicioTexto,</i>      |  |
| 44 | <i>FinalTexto,</i>       |  |
| 45 | <i>ComillaDoble,</i>     |  |
| 46 | <i>ComillaSimple,</i>    |  |
| 47 | <i>Comentario,</i>       |  |
| 48 | <i>InicioComentario,</i> |  |
| 49 | <i>FinComentario,</i>    |  |
| 50 | <i>Entero,</i>           |  |
| 51 | <i>Decimal,</i>          |  |
| 52 | <i>Verdadero,</i>        |  |
| 53 | <i>Falso,</i>            |  |
| 54 | <i>Cadena,</i>           |  |
| 55 | <i>Si,</i>               |  |
| 56 | <i>SiNo,</i>             |  |
| 57 | <i>SiNoSi,</i>           |  |
| 58 | <i>CicloRepite,</i>      |  |
| 59 | <i>Mientras,</i>         |  |
| 60 | <i>Hacer,</i>            |  |
| 61 | <i>Seleccionador,</i>    |  |
| 62 | <i>Caso,</i>             |  |
| 63 | <i>Predeterminado,</i>   |  |
| 64 | <i>Detener,</i>          |  |
| 65 | <i>Clase,</i>            |  |
| 66 | <i>Arreglo,</i>          |  |
| 67 | <i>Funcion,</i>          |  |
| 68 | <i>Procedimiento,</i>    |  |
| 69 | <i>Principal,</i>        |  |
| 70 | <i>Biblioteca,</i>       |  |
| 71 | <i>Imprimir,</i>         |  |
| 72 | <i>Leer,</i>             |  |
| 73 | <i>Importar,</i>         |  |
| 74 | <i>Retornar,</i>         |  |
| 75 | <i>Metodo,</i>           |  |
| 76 | <i>Nulo,</i>             |  |

|    |                              |  |
|----|------------------------------|--|
| 77 | <i>CaracteresEspeciales,</i> |  |
| 78 | <i>P_coma,</i>               |  |
| 79 | <i>Main,</i>                 |  |
| 80 | <i>Constante,</i>            |  |
| 81 | <i>NombreClase,</i>          |  |
| 82 | <i>Identificador,</i>        |  |
| 83 | <i>Numero,</i>               |  |
| 84 | <i>Texto,</i>                |  |
| 85 | <i>ERROR</i>                 |  |
| 86 |                              |  |
| 87 | }                            |  |

## Principal.java

Este código tiene dos propósitos principales:

1. **Generar el analizador léxico:** Utiliza JFlex para procesar el archivo `Lexer.flex`, generando el código necesario para identificar los tokens del lenguaje definido. Esto se logra mediante el método `generarLexer`, que toma como entrada la ruta del archivo.
2. **Abrir la ventana gráfica:** Una vez generado el analizador léxico, se inicia la interfaz gráfica del programa mostrando una ventana llamada `Splash`, que podría ser una pantalla inicial o de bienvenida. Esta ventana se hace visible con `ventanaLogin.setVisible(true)`.

El código combina la preparación del analizador léxico con la interfaz gráfica, sirviendo como punto de entrada al proyecto.

```
package codigo;

import java.io.File;

public class Principal {
    public static void main(String[] args) {
        String ruta = "C:/Users/vanes/OneDrive/Documentos/SEMESTRE 5/LENGUAJES Y AUTOMATAS I/Proyecto/AnalizadorLexico/src/codigo/Lexer.flex";
        generarLexer(ruta);

        Splash ventanaLogin = new Splash();
        ventanaLogin.setVisible(true);
    }

    public static void generarLexer(String ruta) {
        File archivo = new File(ruta);
        JFlex.Main.generate(archivo);
    }
}
```

## FrmPrincipal.java

El código utiliza el paquete javax.swing para crear una interfaz de usuario. La ventana contiene componentes como un área de texto (JTextArea) para la entrada de texto, botones para realizar acciones (como "Analizar" y "Borrar"), y un área de texto para mostrar los resultados del análisis.

En particular, el método btnAnalizarActionPerformed se encarga de tomar el texto ingresado en el área de entrada, guardarlo en un archivo y luego procesarlo para analizarlo mediante un lexer (analizador léxico). Los tokens identificados en el texto se clasifican en categorías como identificadores, números o palabras reservadas, y luego se muestran en el área de resultados. Además, se actualizan los números de línea en tiempo real a medida que se modifica el contenido en el área de entrada de texto. La aplicación está diseñada con un diseño visual básico, utilizando GroupLayout para organizar los componentes y un esquema de color simple para la interfaz.

### Código:

```
1 package codigo;
2
3 import java.awt.Color;
4 import java.awt.Font;
5 import java.io.BufferedReader;
6 import java.io.File;
7 import java.io.FileNotFoundException;
8 import java.io.FileReader;
9 import java.io.IOException;
10 import java.io.PrintWriter;
11 import java.io.Reader;
12 import java.util.logging.Level;
13 import java.util.logging.Logger;
14 import javax.swing.BorderFactory;
15 import javax.swing.JButton;
16 import javax.swing.JTextArea;
17 import javax.swing.JScrollPane;
18 import javax.swing.JPanel;
19 import javax.swing.UIManager;
20 import javax.swing.plaf.metal.MetalLookAndFeel;
21 import javax.swing.text.Element;
22
23 public class FrmPrincipal extends javax.swing.JFrame {
24
25     public FrmPrincipal() {
26         initComponents();
27         this.setLocationRelativeTo(null);
28     }
29
30     private void initComponents() {
31         // Configurar apariencia general
32         try {
33             UIManager.setLookAndFeel(new MetalLookAndFeel());
34         } catch (Exception e) {
35             e.printStackTrace();
36         }
37     }
38 }
```





```

38 // Componentes
39 txtEntrada = new JTextArea();
40 lineNumberPane = new JTextArea();
41 btnAnalizar = new JButton();
42 btnBorrar = new JButton();
43 jScrollPane1 = new JScrollPane();
44 txtResultado = new JTextArea();
45
46 // Configurar ventana
47 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
48 getContentPane().setBackground(new Color(230, 240, 255));
49
50 // Configurar JTextArea para entrada de texto
51 txtEntrada.setFont(new Font("Consolas", Font.PLAIN, 16));
52 txtEntrada.setRows(10); // Número de filas visibles por defecto
53 txtEntrada.setTabSize(4); // Tamaño del tabulador
54 txtEntrada.setWrapStyleWord(true);
55 txtEntrada.setLineWrap(true);
56 txtEntrada.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
57 txtEntrada.getDocument().addDocumentListener(new javax.swing.event.DocumentListener() {
58     public void insertUpdate(javax.swing.event.DocumentEvent e) {
59         updateLineNumbers();
60     }
61     public void removeUpdate(javax.swing.event.DocumentEvent e) {
62         updateLineNumbers();
63     }
64     public void changedUpdate(javax.swing.event.DocumentEvent e) {
65         updateLineNumbers();
66     }
67 });
68
69 // Configurar JTextArea para números de línea
70 lineNumberPane.setFont(new Font("Consolas", Font.PLAIN, 16));
71 lineNumberPane.setEditable(false);
72 lineNumberPane.setBackground(new Color(220, 220, 220));

```

```

74 lineNumberPane.setBackground(new Color(220, 220, 220));
75 lineNumberPane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
76
77 // Añadir JTextAreas al JScrollPane
78 jScrollPane1.setViewportView(txtEntrada);
79 jScrollPane1.setRowHeaderView(lineNumberPane);
80
81 // Botón Analizar
82 btnAnalizar.setFont(new Font("Tahoma", Font.BOLD, 16));
83 btnAnalizar.setText("Analizar");
84
85 btnAnalizar.setBackground(new Color(50, 150, 250));
86 btnAnalizar.setForeground(Color.WHITE);
87 btnAnalizar.setFocusPainted(false);
88 btnAnalizar.setBorder(BorderFactory.createCompoundBorder(
89     BorderFactory.createLineBorder(Color.BLUE, 1),
90     BorderFactory.createEmptyBorder(10, 20, 10, 20)
91 ));
92 btnAnalizar.addActionListener(new java.awt.event.ActionListener() {
93     public void actionPerformed(java.awt.event.ActionEvent evt) {
94         btnAnalizarActionPerformed(evt);
95     }
96 });
97
98 // Botón Borrar
99 btnBorrar.setFont(new Font("Tahoma", Font.BOLD, 16));
100 btnBorrar.setText("Borrar");
101 btnBorrar.setBackground(new Color(200, 50, 50));
102 btnBorrar.setForeground(Color.WHITE);
103 btnBorrar.setFocusPainted(false);
104 btnBorrar.setBorder(BorderFactory.createCompoundBorder(
105     BorderFactory.createLineBorder(Color.RED, 1),
106     BorderFactory.createEmptyBorder(5, 15, 5, 15)
107 ));
108 btnBorrar.addActionListener(new java.awt.event.ActionListener() {
109     public void actionPerformed(java.awt.event.ActionEvent evt) {
110         txtEntrada.setText("");

```



```

110         txtEntrada.setText("");
111     }
112     });
113
114     // JTextArea para resultados
115     txtResultado.setColumns(20);
116     txtResultado.setRows(5);
117     txtResultado.setEditable(false);
118     txtResultado.setFont(new Font("Consolas", Font.PLAIN, 16));
119     jScrollPane1.setViewportView(txtResultado);
120
121     // Configurar Layout
122     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
123     getContentPane().setLayout(layout);
124     layout.setHorizontalGroup(
125         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
126             .addGroup(layout.createSequentialGroup()
127                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
128                     .addGroup(layout.createSequentialGroup()
129                         .add(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 580, Short.MAX_VALUE)
130                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
131                             .addGroup(layout.createSequentialGroup()
132                                 .addGap(150, 150, 150)
133                                 .addComponent(btnAnalizar, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.swing.GroupLayout.PREFERRED_SIZE)
134                                 .addGap(50, 50, 50)
135                                 .addComponent(btnBorrar, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.swing.GroupLayout.PREFERRED_SIZE)
136                                 .addGap(0, 0, Short.MAX_VALUE))
137                             .addContainerGap())
138                     .addGroup(layout.createSequentialGroup()
139                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
140                             .addGroup(layout.createSequentialGroup()
141                                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
142                                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
143                                         .addComponent(btnAnalizar, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
144                                         .addComponent(btnBorrar, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
145                                     .addPreferredGap(javax.swing.GroupLayout.Alignment.BASELINE)

```

```

145         .addComponent(btnBorrar, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
146         .addPreferredGap(javax.swing.GroupLayout.Alignment.BASELINE)
147         .addComponent(scrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 200, javax.swing.GroupLayout.PREFERRED_SIZE)
148         .addPreferredGap(javax.swing.GroupLayout.Alignment.BASELINE)
149         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)
150         .addContainerGap())
151     );
152
153     pack();
154     updateLineNumbers();
155 }
156
157 private void updateLineNumbers() {
158     StringBuilder lineNumbers = new StringBuilder();
159     int lines = txtEntrada.getLineCount();
160     for (int i = 1; i <= lines; i++) {
161         lineNumbers.append(i).append("\n");
162     }
163     lineNumberPane.setText(lineNumbers.toString());
164 }
165
166 private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {
167     File archivo = new File("archivo.txt");
168     PrintWriter escribir;
169     try {
170         escribir = new PrintWriter(archivo);
171         escribir.print(txtEntrada.getText());
172         escribir.close();
173     } catch (FileNotFoundException ex) {
174         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
175     }
176
177     try {
178         Reader lector = new BufferedReader(new FileReader("archivo.txt"));
179         Lexer lexer = new Lexer(lector);
180         String resultado = "";
181         while (true) {

```

```
181 | while (true) {
182 |     Tokens tokens = lexer.yylex();
183 |     if (tokens == null) {
184 |         resultado += "FIN";
185 |         txtResultado.setText(resultado);
186 |         return;
187 |     }
188 |     switch (tokens) {
189 |         case ERROR:
190 |             resultado += "Símbolo no definido\n";
191 |             break;
192 |         case Identificador:
193 |         case Numero:
194 |         case Reservadas:
195 |             resultado += lexer.lexeme + ": Es un " + tokens + "\n";
196 |             break;
197 |         default:
198 |             resultado += "Token: " + tokens + "\n";
199 |             break;
200 |     }
201 | }
202 | } catch (FileNotFoundException ex) {
203 |     Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
204 | } catch (IOException ex) {
205 |     Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
206 | }
207 | }
208 |
209 | public static void main(String args[]) {
210 |     java.awt.EventQueue.invokeLater(new Runnable() {
211 |         public void run() {
212 |             new FrmPrincipal().setVisible(true);
213 |         }
214 |     });
215 | }
216 | }
```

## Splash.java

Este código implementa una pantalla de inicio personalizada (splash screen) para la aplicación. Al ejecutarse, muestra una ventana sin bordes ni barra de título, con un fondo transparente y diseño visual atractivo.

La ventana incluye una barra de progreso dinámica que avanza del 0% al 100%, cambiando de color según el progreso y mostrando texto animado, como el título del proyecto ("Program-Arte"). Además, permite mover la ventana arrastrándola con el mouse y tiene un botón decorativo para cerrarla.

El diseño se adapta automáticamente al tamaño de la ventana, incluyendo la imagen de fondo y otros elementos visuales, ofreciendo una experiencia moderna y estética al iniciar la aplicación.

```
1 package codigo;
2 import java.awt.Color;
3 import java.awt.Image;
4 import java.awt.Point;
5 import java.awt.event.ActionEvent;
6 import javax.swing.ImageIcon;
7 import javax.swing.JLabel;
8 import javax.swing.Timer;
9
10 public class Splash extends javax.swing.JFrame {
11
12     private final Color mTransparent;
13     private Point mPoint;
14
15     public Splash() {
16         mTransparent = new Color(0,0,0,0);
17         setUndecorated(true);
18         initComponents();
19         setLocationRelativeTo(null);
20         setBackground(mTransparent);
21
22         // Pintar el fondo del JFrame y hacerlo transparente
23         ImageDecore mFondo = new ImageDecore(pnlBackground, "/icons/ff2.png");
24         pnlBackground.add(mFondo).repaint();
25         pnlBackground.setOpaque(false);
26         pnlBackground.setBorder(null);
27         pnlBackground.setBackground(mTransparent);
28
29         // Crear JLabel para la imagen
30         JLabel lblImagen = new javax.swing.JLabel();
31
32         // Pintar el icono de salir del JFrame y hacer su fondo transparente
33         ImageDecore mSalir = new ImageDecore(pnlCerrar, "/icons/salir.png");
34         pnlCerrar.add(mSalir).repaint();
35         pnlCerrar.setOpaque(false);
36         pnlCerrar.setBorder(null);
37         pnlCerrar.setBackground(mTransparent);
```



```
41 // Cargar la imagen y escalarla al tamaño del JFrame
42 ImageIcon iconOriginal = new javax.swing.ImageIcon(getClass().getResource("/icons/image.png"));
43 Image imagenEscalada = iconOriginal.getImage().getScaledInstance(getWidth(), getHeight(), Image.SCALE_SMOOTH);
44 lblImagen.setIcon(new javax.swing.ImageIcon(imagenEscalada));
45
46 // Configurar el JLabel
47 lblImagen.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
48 lblImagen.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
49
50 // Configurar layout del panel
51 pnlBackground.setLayout(null);
52
53 lblImagen.setBounds(0, 0, getWidth(), getHeight());
54 pnlBackground.add(lblImagen); // Añadir al panel
55 pnlBackground.setComponentZOrder(lblImagen, 0); // Mover el JLabel al frente
56
57 // Añadir un listener para ajustar la imagen cuando cambie el tamaño del JFrame
58 this.addComponentListener(new java.awt.event.ComponentAdapter() {
59     @Override
60     public void componentResized(java.awt.event.ComponentEvent e) {
61
62         // Cargar la imagen y escala al tamaño deseado
63         ImageIcon iconOriginal = new javax.swing.ImageIcon(getClass().getResource("/icons/image.png"));
64         int imagenAncho = 400;
65         int imagenAlto = 300;
66         Image imagenEscalada = iconOriginal.getImage().getScaledInstance(imagenAncho, imagenAlto, Image.SCALE_SMOOTH);
67         lblImagen.setIcon(new javax.swing.ImageIcon(imagenEscalada));
68
69         // Ajustar el tamaño del JLabel
70         lblImagen.setBounds((getWidth() - imagenAncho) / 2, 50, imagenAncho, imagenAlto);
71     }
72 });
73
74 // Iniciar el progress bar
75 ProgressBarInicado();
76 }
```

```
77 private void ProgressBarInicado() {
78     String textoCompleto = "Program-Arte ";
79     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
80     jLabel1.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
81     jLabel1.setFont(new java.awt.Font("Comic Sans MS", java.awt.Font.PLAIN, 32));
82     Color colorInicio = Color.decode("#F0E68C");
83     Color colorMedio = Color.decode("#FFD700");
84     Color colorFinal = Color.decode("#DAA520");
85     // Inicia el progress bar a partir de un Timer
86     Timer mTimer = new Timer(30, (ActionEvent e) -> {
87         int progress = pbCarga.getValue() + 1;
88         if (progress <= 100) {
89             pbCarga.setValue(progress); // Valor del progress bar
90             pbCarga.setBackground(Color.white);
91             pbCarga.setStringPainted(true);
92             pbCarga.setString("Loading... " + progress + "%");
93
94             // Cambiar el color del progress bar según el progreso usando los colores definidos
95             if (progress <= 30) {
96                 pbCarga.setForeground(colorInicio);
97             } else if (progress <= 60) {
98                 pbCarga.setForeground(colorMedio);
99             } else {
100                 pbCarga.setForeground(colorFinal);
101             }
102             // Determinar cuántas letras mostrar según el progreso
103             int numLetras = progress / 7; // Cada % se muestra una letra
104
105             String textoActual = textoCompleto.substring(0, numLetras);
106             // Actualizar el texto de la etiqueta con las letras correspondientes
107             jLabel1.setText(textoActual);
108         }
109     });
110
111     mTimer.start();
112 }
```



```
115 @SuppressWarnings("unchecked")
116 Generated Code
117
118 private void pnlBackgroundMousePressed(java.awt.event.MouseEvent evt) {
119     mPoint = evt.getPoint(); // Obtener el punto de partida del movimiento
120     getComponentAt(mPoint);
121 }
122
123 private void pnlBackgroundMouseDragged(java.awt.event.MouseEvent evt) {
124     // Obtener las posiciones actuales del JFrame en X, Y
125     int CurrentX = this.getLocation().x;
126     int CurrentY = this.getLocation().y;
127     // Obtener el movimiento del del JFrame en X, Y
128     int MoveX = (CurrentX + evt.getX()) - (CurrentX + mPoint.x);
129     int MoveY = (CurrentY + evt.getY()) - (CurrentY + mPoint.y);
130
131     // Calcular las nuevas posiciones
132     int x = CurrentX + MoveX;
133     int y = CurrentY + MoveY;
134
135     // Asignar las posiciones al JFrame para generar el movimiento
136     this.setLocation(x, y);
137 }
138
139 private void pnlCerrarMouseClicked(java.awt.event.MouseEvent evt) {
140     System.exit(0);
141 }
142
143 public static void main(String args[]) {
144     /* Set the Nimbus look and feel */
145     Look and feel setting code (optional)
146     //</editor-fold>
147     //</editor-fold>
148     //</editor-fold>
149     //</editor-fold>
```

```
244 public static void main(String args[]) {
245     /* Set the Nimbus look and feel */
246     Look and feel setting code (optional)
247     //</editor-fold>
248     //</editor-fold>
249     //</editor-fold>
250     //</editor-fold>
251     //</editor-fold>
252     //</editor-fold>
253     //</editor-fold>
254
255     /* Create and display the form */
256     java.awt.EventQueue.invokeLater(new Runnable() {
257         public void run() {
258             new Splash().setVisible(true);
259         }
260     });
261 }
262
263 // Variables declaration - do not modify
264 private javax.swing.JButton jButton1;
265 private javax.swing.JLabel jLabel1;
266 private javax.swing.JPanel jPanel1;
267 private javax.swing.JProgressBar pbCarga;
268 private javax.swing.JPanel pnlBackground;
269 private javax.swing.JPanel pnlCerrar;
270 // End of variables declaration
271 }
```

## LoginForm.java

El código implementa un formulario de inicio de sesión con Java Swing. Tiene dos paneles: uno para registro y otro para login. Valida credenciales almacenadas en un HashMap. Si el inicio es exitoso, abre una ventana de bienvenida (FrmBienvenida); si falla, muestra un mensaje de error. Permite agregar nuevos usuarios con el método addUser. Ofrece un diseño moderno y funcional con colores y elementos centrados en la pantalla.

```
2 package codigo;
3 import javax.swing.*;
4 import java.awt.*;
5 import java.util.HashMap;
6
7 public class LoginForm extends JFrame {
8     private JTextField usernameField;
9     private JPasswordField passwordField;
10    private static HashMap<String, String> users = new HashMap<>(); // Almacena usuarios y contraseñas
11
12    public LoginForm() {
13        // Usuario predeterminado
14        users.put("admin", "12345");
15
16        // Configuración de la ventana principal
17        setTitle("Login");
18        setSize(600, 400); // Ventana más grande
19        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20        setLayout(null);
21
22        // Panel izquierdo (color sólido)
23        JPanel leftPanel = new JPanel();
24        leftPanel.setBackground(Color.decode("#33B5E5")); // Cambiar color aquí
25        leftPanel.setBounds(0, 0, 300, 400);
26        leftPanel.setLayout(null); // Uso de layout absoluto para ajustar componentes
27        add(leftPanel);
28
29        // Etiqueta de mensaje en el panel izquierdo
30        JLabel messageLabel = new JLabel("¿Aún no tienes una cuenta?");
31        messageLabel.setForeground(Color.WHITE);
32        messageLabel.setFont(new Font("Arial", Font.BOLD, 16));
33        messageLabel.setBounds(30, 100, 240, 30);
34        leftPanel.add(messageLabel);
35    }
```

```
37    // Botón de registro en el panel izquierdo
38    JButton registerButton = new JButton("Nuevo Registro");
39    registerButton.setBounds(60, 150, 180, 40);
40    registerButton.setFocusPainted(false);
41    registerButton.setBackground(Color.WHITE);
42    registerButton.setForeground(Color.decode("#33B5E5"));
43    registerButton.setFont(new Font("Arial", Font.BOLD, 14));
44    registerButton.setBorder(BorderFactory.createLineBorder(Color.decode("#33B5E5"), 2));
45    registerButton.addActionListener(e -> new RegisterForm());
46    leftPanel.add(registerButton);
47
48    // Estilo de hover para el botón
49    registerButton.addMouseListener(new java.awt.event.MouseAdapter() {
50        public void mouseEntered(java.awt.event.MouseEvent evt) {
51            registerButton.setBackground(Color.decode("#33B5E5"));
52            registerButton.setForeground(Color.WHITE);
53        }
54
55        public void mouseExited(java.awt.event.MouseEvent evt) {
56            registerButton.setBackground(Color.WHITE);
57            registerButton.setForeground(Color.decode("#33B5E5"));
58        }
59    });
60
61    // Panel derecho (login)
62    JPanel rightPanel = new JPanel();
63    rightPanel.setBackground(Color.WHITE); // Fondo blanco para el login
64    rightPanel.setBounds(300, 0, 300, 400);
65    rightPanel.setLayout(null);
66    add(rightPanel);
67
68    JLabel loginLabel = new JLabel("Login");
69    loginLabel.setFont(new Font("Arial", Font.BOLD, 30));
70    loginLabel.setForeground(Color.BLACK);
71    loginLabel.setBounds(110, 30, 100, 30);
72    rightPanel.add(loginLabel);
```

```

74 JLabel userLabel = new JLabel("Ingresa Usuario:");
75 userLabel.setBounds(50, 100, 120, 25);
76 userLabel.setFont(new Font("Arial", Font.PLAIN, 14));
77 rightPanel.add(userLabel);
78
79 usernameField = new JTextField();
80 usernameField.setBounds(50, 130, 200, 30);
81 rightPanel.add(usernameField);
82
83 JLabel passLabel = new JLabel("Ingresa Contraseña:");
84 passLabel.setBounds(50, 180, 150, 25);
85 passLabel.setFont(new Font("Arial", Font.PLAIN, 14));
86 rightPanel.add(passLabel);
87
88 passwordField = new JPasswordField();
89 passwordField.setBounds(50, 210, 200, 30);
90 rightPanel.add(passwordField);
91
92 JButton loginButton = new JButton("Entrar");
93 loginButton.setBounds(50, 270, 200, 40);
94 loginButton.setFocusPainted(false);
95 loginButton.setBackground(Color.decode("#33B5E5"));
96 loginButton.setForeground(Color.WHITE);
97 loginButton.setFont(new Font("Arial", Font.BOLD, 14));
98 loginButton.setBorder(BorderFactory.createLineBorder(Color.decode("#33B5E5"), 2));
99 loginButton.addActionListener(e -> {
100     String username = usernameField.getText();
101     String password = new String(passwordField.getPassword());
102
103     // Validar las credenciales
104     // Validar las credenciales
105     if (users.containsKey(username) && users.get(username).equals(password)) {
106         JOptionPane.showMessageDialog(null, "Inicio de sesión exitoso!");
107         // Crear e iniciar la ventana de bienvenida
108         FrmBienvenida ventanaBienvenida = new FrmBienvenida();
109         ventanaBienvenida.setVisible(true);

```

```

111     dispose(); // Cierra la ventana actual de inicio de sesión
112 } else {
113     JOptionPane.showMessageDialog(null, "Usuario o contraseña incorrectos");
114 }
115
116 });
117 rightPanel.add(loginButton);
118
119 // Centrar la ventana
120 setLocationRelativeTo(null);
121
122 setVisible(true);
123 }
124
125 public static void main(String[] args) {
126     new LoginForm();
127 }
128
129 // Método para agregar un usuario a la lista
130 public static void addUser(String username, String password) {
131     users.put(username, password);
132 }
133 }

```



## RegisterForm.java

El código implementa un formulario de registro con Java Swing. Permite a los usuarios crear una nueva cuenta al ingresar un nombre de usuario y una contraseña, que luego se almacenan en el LoginForm mediante el método addUser. Si el registro es exitoso, se muestra un mensaje de confirmación y se cierra la ventana de registro, abriendo el formulario de inicio de sesión. También permite regresar al formulario de login desde el panel derecho. El diseño tiene un panel dividido con colores y campos centrados para una experiencia visual atractiva.

```
1 package codigo;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class RegisterForm extends JFrame {
7     public RegisterForm() {
8         // Configuración de la ventana de registro
9         setTitle("Nuevo Registro");
10        setSize(600, 400);
11        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
12        setLayout(null);
13
14        // Panel izquierdo (campos de registro)
15        JPanel leftPanel = new JPanel();
16        leftPanel.setBackground(Color.WHITE);
17        leftPanel.setBounds(0, 0, 300, 400);
18        leftPanel.setLayout(null);
19        add(leftPanel);
20
21        JLabel registerLabel = new JLabel("Nuevo Registro");
22        registerLabel.setFont(new Font("Arial", Font.BOLD, 20));
23        registerLabel.setForeground(Color.BLACK);
24        registerLabel.setBounds(100, 30, 200, 30);
25        leftPanel.add(registerLabel);
26
27        JLabel userLabel = new JLabel("Nuevo usuario:");
28        userLabel.setBounds(50, 100, 120, 25);
29        userLabel.setFont(new Font("Arial", Font.PLAIN, 14));
30        leftPanel.add(userLabel);
31
32        JTextField userField = new JTextField();
33        userField.setBounds(50, 130, 200, 30);
34        leftPanel.add(userField);
```



|    |   |  |
|----|---|--|
| 36 | JLabel passLabel = new JLabel("Nueva contraseña:");                                 |  |
| 37 | passLabel.setBounds(50, 180, 150, 25);  |  |
| 38 | passLabel.setFont(new Font("Arial", Font.PLAIN, 14));                               |  |
| 39 | leftPanel.add(passLabel);   |  |
| 40 |   |  |
| 41 | JPasswordField passField = new JPasswordField();                                    |  |
| 42 | passField.setBounds(50, 210, 200, 30);  |  |
| 43 | leftPanel.add(passField);   |  |
| 44 |   |  |
| 45 | JButton submitButton = new JButton("Registrar");                                    |  |
| 46 | submitButton.setBounds(50, 270, 200, 40);   |  |
| 47 | submitButton.setFocusPainted(false);  |  |
| 48 | submitButton.setBackground(Color.decode("#33B5E5"));                                |  |
| 49 | submitButton.setForeground(Color.WHITE);  |  |
| 50 | submitButton.setFont(new Font("Arial", Font.BOLD, 14));                             |  |
| 51 | submitButton.setBorder(BorderFactory.createLineBorder(Color.decode("#33B5E5"), 2)); |  |
| 52 | submitButton.addActionListener(e -> {   |  |
| 53 | String newUsername = userField.getText();   |  |
| 54 | String newPassword = new String(passField.getPassword());                           |  |
| 55 |   |  |
| 56 | if (!newUsername.isEmpty() && !newPassword.isEmpty()) {                             |  |
| 57 | LoginForm.addUser(newUsername, newPassword);  |  |
| 58 | JOptionPane.showMessageDialog(null, "Usuario registrado correctamente.");           |  |
| 59 | dispose(); // Cierra la ventana de registro   |  |
| 60 | new LoginForm().setVisible(true);   |  |
| 61 | } else {  |  |
| 62 | JOptionPane.showMessageDialog(null, "Por favor, complete todos los campos.");       |  |
| 63 | }   |  |
| 64 | });   |  |
| 65 | leftPanel.add(submitButton);  |  |
| 66 |   |  |
| 67 | // Panel derecho (color sólido)   |  |
| 68 | JPanel rightPanel = new JPanel();   |  |
| 69 | rightPanel.setBackground(Color.decode("#33B5E5"));                                  |  |
| 70 | rightPanel.setBounds(300, 0, 300, 400);   |  |
| 71 | rightPanel.setLayout(null);   |  |

|    |   |  |
|----|---|--|
| 74 | // Botón para regresar al login   |  |
| 75 | JButton backButton = new JButton("Regresar al Login");                            |  |
| 76 | backButton.setBounds(60, 150, 180, 40);   |  |
| 77 | backButton.setFocusPainted(false);  |  |
| 78 | backButton.setBackground(Color.WHITE);  |  |
| 79 | backButton.setForeground(Color.decode("#33B5E5"));                                |  |
| 80 | backButton.setFont(new Font("Arial", Font.BOLD, 14));                             |  |
| 81 | backButton.setBorder(BorderFactory.createLineBorder(Color.decode("#33B5E5"), 2)); |  |
| 82 | backButton.addActionListener(e -> {   |  |
| 83 | new LoginForm().setVisible(true); // Crear y mostrar la ventana LoginForm         |  |
| 84 | dispose(); // Cierra la ventana de registro                                       |  |
| 85 | });   |  |
| 86 | rightPanel.add(backButton);   |  |
| 87 |   |  |
| 88 | // Centrar la ventana   |  |
| 89 | setLocationRelativeTo(null);  |  |
| 90 |   |  |
| 91 | setVisible(true);   |  |
| 92 | }   |  |
| 93 |   |  |
| 94 | public static void main(String[] args) {  |  |
| 95 | // Crear la ventana de registro al iniciar el programa                            |  |
| 96 | new RegisterForm().setVisible(true);  |  |
| 97 | }   |  |
| 98 | }   |  |

## FrmBienvenida.java

Este programa en Java crea una ventana gráfica con Swing que muestra un mensaje de bienvenida. La clase FrmBienvenida extiende JFrame y utiliza un panel con fondo azul (jPanel1) que contiene una etiqueta (jLabel1) con el texto "Bienvenidos". El diseño centra el texto en la ventana. Al ejecutarse, la ventana se muestra centrada en la pantalla y está configurada para cerrarse al salir.

```
2 package codigo;
3
4 public class FrmBienvenida extends javax.swing.JFrame {
5
6     public FrmBienvenida() {
7         initComponents();
8         this.setLocationRelativeTo(null); // Centrar pantalla
9     }
10
11
12     @SuppressWarnings("unchecked")
13     // <editor-fold defaultstate="collapsed" desc="Generated Code">
14     private void initComponents() {
15
16         jPanel1 = new javax.swing.JPanel();
17         jLabel1 = new javax.swing.JLabel();
18
19         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
20
21         jPanel1.setBackground(new java.awt.Color(51, 153, 255));
22
23         jLabel1.setFont(new java.awt.Font("Dialog", 1, 36)); // NOI18N
24         jLabel1.setForeground(new java.awt.Color(255, 255, 255));
25         jLabel1.setText("Bienvenidos");
26
27         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
28         jPanel1.setLayout(jPanel1Layout);
29         jPanel1Layout.setHorizontalGroup(
30             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
31                 .addGroup(jPanel1Layout.createSequentialGroup()
32                     .addGap(91, 91, 91)
33                     .addComponent(jLabel1)
34                     .addGap(97, Short.MAX_VALUE))
35             );
36         jPanel1Layout.setVerticalGroup(
```

```

37         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
38             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
39                 .addContainerGap(135, Short.MAX_VALUE)
40                 .addComponent(jLabel1)
41                 .addGap(118, 118, 118))
42         );
43
44         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
45         getContentPane().setLayout(layout);
46         layout.setHorizontalGroup(
47             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
48                 .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
49         );
50         layout.setVerticalGroup(
51             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
52                 .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
53         );
54
55         pack();
56     } // </editor-fold>
57
58     /**
59     * @param args the command line arguments
60     */
61     public static void main(String args[]) {
62         // Set the Nimbus look and feel */
63         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
64         // If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
65         // For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
66         //
67         try {
68             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
69                 if ("Nimbus".equals(info.getName())) {
70                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
71                     break;
72                 }
73             }
74         }

```

```

75         } catch (ClassNotFoundException ex) {
76             java.util.logging.Logger.getLogger(FrmBienvenida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
77         } catch (InstantiationException ex) {
78             java.util.logging.Logger.getLogger(FrmBienvenida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
79         } catch (IllegalAccessException ex) {
80             java.util.logging.Logger.getLogger(FrmBienvenida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
81         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
82             java.util.logging.Logger.getLogger(FrmBienvenida.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
83         }
84     } //</editor-fold>
85     //</editor-fold>
86
87     /* Create and display the form */
88     java.awt.EventQueue.invokeLater(new Runnable() {
89         public void run() {
90             new FrmBienvenida().setVisible(true);
91         }
92     });
93
94     // Variables declaration - do not modify
95     private javax.swing.JLabel jLabel1;
96     private javax.swing.JPanel jPanel1;
97     // End of variables declaration
98 }

```

## Main.java

Este código comienza mostrando una pantalla de presentación (conocida como splash screen) durante cinco segundos antes de cargar la pantalla principal de inicio de sesión. Para lograr esto, utiliza un hilo separado que permite ejecutar la lógica del splash sin bloquear el resto de la aplicación.

El flujo funciona de la siguiente manera: al iniciar la aplicación, se crea un objeto que define una tarea (Runnable) donde primero se muestra la ventana del splash. Esta ventana se mantiene visible mientras el programa se detiene momentáneamente durante cinco segundos utilizando `Thread.sleep`. Pasado este tiempo, la ventana del splash se cierra y se muestra la ventana principal de inicio de sesión. Este hilo de ejecución se inicia en paralelo, permitiendo que la aplicación sea responsiva y manejando la secuencia de forma ordenada.

```
2
3 import java.util.logging.Level;
4 import java.util.logging.Logger;
5
6 public class Main {
7
8     public static void main(String[] args) {
9
10
11         Runnable mRun = () -> {
12
13             // JFrame que funge como splash
14             Splash mSplash = new Splash();
15             mSplash.setVisible(true);
16
17             try {
18                 Thread.sleep(5000); // 5000 milisegundos equivale a 5 segundos.
19             } catch (InterruptedException ex) {
20                 Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
21             }
22
23             mSplash.dispose();
24
25             // JFrame que funge como pantalla principal
26             LoginForm mLoginForm = new LoginForm();
27             mLoginForm.setVisible(true);
28
29         };
30
31         Thread mHiloSplash = new Thread(mRun);
32         mHiloSplash.start();
33
34
35
36
37 }
```

PANTALLAS RESULTANTES CON PRUEBAS DEL RECONOCIMIENTO DE LOS  
TOKENS

Pantallas de ejecución

Splash.java





## LoginForm.java

Login

¿Aún no tienes una cuenta?

Nuevo Registro

Ingresa Usuario:

Ingresa Contraseña:

Entrar

## RegisterForm.java

Nuevo Registro

Nuevo usuario:

Nueva contraseña:

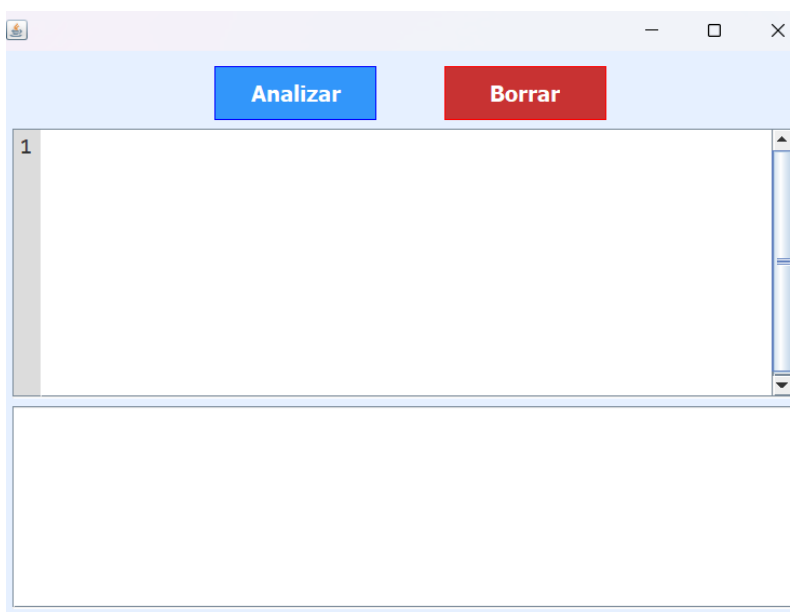
Registrar

Regresar al Login

## FrmBienvenida.java

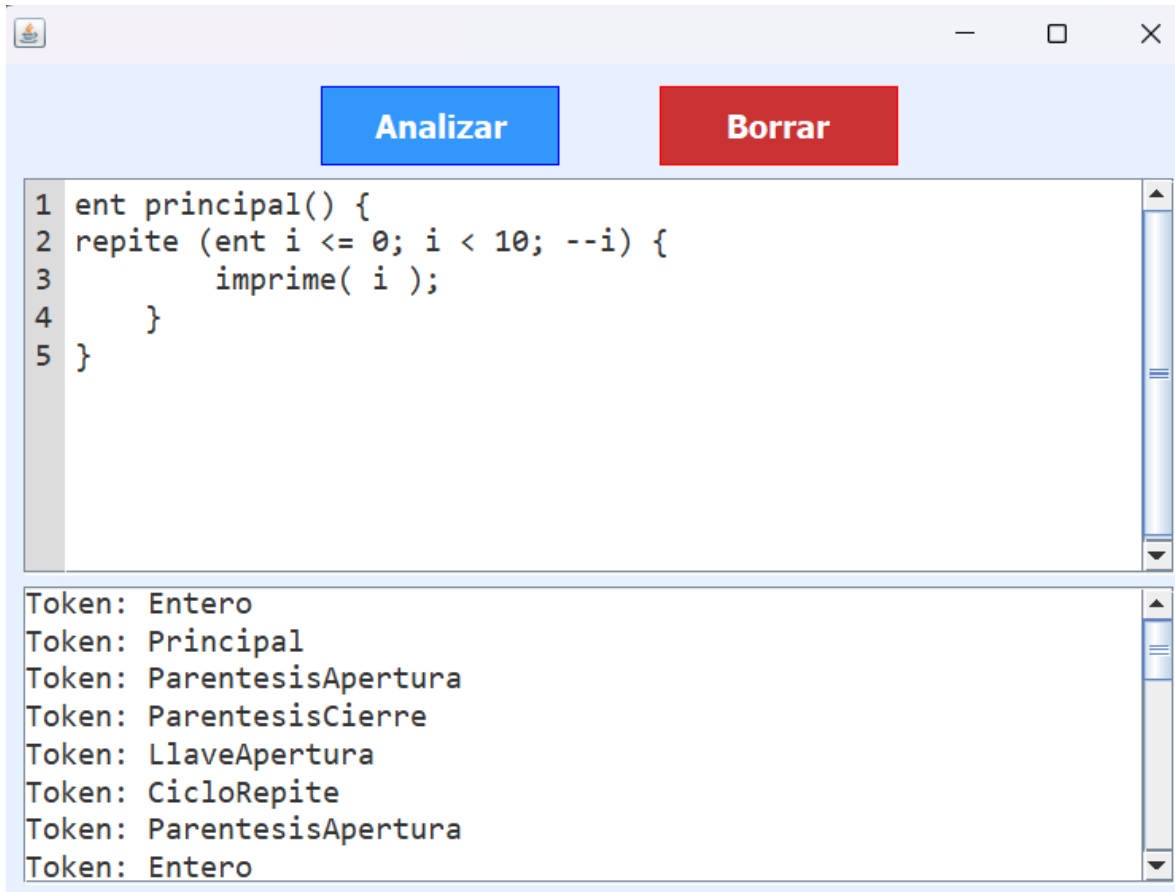


## FrmPrincipal.java





## Ejemplo del análisis de un código:



The screenshot shows a software interface for code analysis. At the top, there are two buttons: 'Analizar' (blue) and 'Borrar' (red). Below these is a text area containing the following code:

```
1 ent principal() {  
2 repite (ent i <= 0; i < 10; --i) {  
3     imprime( i );  
4 }  
5 }
```

Below the code area is a list of tokens identified by the tool:

- Token: Entero
- Token: Principal
- Token: ParentesisApertura
- Token: ParentesisCierre
- Token: LlaveApertura
- Token: CicloRepite
- Token: ParentesisApertura
- Token: Entero

## V. Conclusiones:

# Conclusiones

El objetivo principal del desarrollo de este proyecto es crear un compilador, enfocado en la fase del análisis léxico, que permite identificar y clasificar los tokens presentes en un código fuente. El analizador léxico descompone el código en unidades mínimas de significado, como palabras reservadas, identificadores, operadores y símbolos, sirviendo como base para las fases posteriores del compilador, como el análisis sintáctico y semántico. Este proceso no solo garantiza la correcta interpretación del código, sino también su estructura y claridad, esenciales en el desarrollo de lenguajes de programación.

Para la implementación del proyecto se utilizó el lenguaje de programación **Java**, desarrollado en el entorno **NetBeans** por su facilidad para manejar proyectos complejos. Asimismo, se incorporaron las librerías especializadas **JFlex**, **JCup**, y **JCup 11a**.

- **JFlex** es una herramienta generadora de analizadores léxicos basada en autómatas finitos, que facilita la definición y detección de patrones específicos en el código fuente mediante expresiones regulares.
- **JCup** y su versión **JCup 11a** se emplearon para generar parsers basados en gramáticas, permitiendo definir las reglas sintácticas necesarias para analizar el código de manera más estructurada.

Estas herramientas trabajan en conjunto para garantizar que el proyecto sea robusto y eficiente. En términos generales, el analizador léxico desarrollado no solo valida las cadenas de texto del código fuente, sino que también optimiza el proceso de análisis, conectando el lenguaje humano con las operaciones internas del compilador. Esto representa un avance fundamental en la construcción de sistemas computacionales precisos y confiables.

En el desarrollo del proyecto no he tenido muchas complicaciones, aunque a veces puede ser tedioso el desarrollar las instrucciones y los tokens de cada uno. Este proyecto me hizo darme cuenta de todo lo que conlleva un lenguaje de programación, y me ha hecho pensar mucho para no olvidar palabras reservadas o partes importantes de un código de programación, ya que el desarrollar los tokens simples en donde hay símbolos es relativamente fácil, pero si se requiere hacer un identificador, nombre de clase o dígito es más complejo realizar esto ya que conlleva aplicar algunos conocimientos de las unidades pasadas para desarrollar un patrón que permita generar estas palabras más complejas.