

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:** Jesús Navarrete 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 NetBeans

## IV. Desarrollo de la práctica:

### ANALIZADOR LÉXICO

#### **DESCRIPCIÓN DEL PROBLEMA:** Construcción de un Analizador Léxico

Un analizador léxico, también conocido como lexer o escáner, es una de las primeras y más fundamentales etapas en el proceso de compilación de un lenguaje de programación. Este componente tiene como principal objetivo transformar el código fuente, que generalmente consiste en una secuencia de caracteres, en una secuencia estructurada de tokens. Los tokens representan las unidades básicas y esenciales del lenguaje, tales como palabras reservadas, identificadores, operadores matemáticos y lógicos, delimitadores como paréntesis o llaves, y literales como números o cadenas de texto.

El proceso de análisis léxico no solo consiste en identificar y categorizar estas unidades, sino también en ignorar elementos que no contribuyen directamente a la semántica del lenguaje, como espacios en blanco y comentarios. Esto permite que las etapas posteriores del compilador trabajen con una representación más abstracta y manejable del código fuente.

La construcción de un analizador léxico enfrenta varios desafíos clave. Uno de ellos es la definición clara y precisa de las reglas léxicas que describen los tokens válidos del lenguaje. Estas reglas suelen expresarse mediante expresiones regulares, una herramienta poderosa para capturar patrones en cadenas de texto. Además, es crucial implementar un mecanismo eficiente para procesar el código fuente, ya que un lexer mal diseñado puede convertirse en un cuello de botella en el proceso de compilación.

Para simplificar y automatizar la implementación de analizadores léxicos, se utilizan herramientas como JFlex. JFlex es una librería que permite a los desarrolladores especificar las reglas léxicas de su lenguaje de programación utilizando una sintaxis basada en expresiones regulares. A partir de estas reglas, JFlex genera código Java que implementa el analizador léxico. Esto no solo reduce el tiempo y el esfuerzo de desarrollo, sino que también garantiza que el lexer sea eficiente y robusto.

En el contexto de este proyecto, se busca diseñar un analizador léxico para un lenguaje de programación definido específicamente, con una sintaxis y semántica particulares. Este analizador tendrá la responsabilidad de leer el código fuente proporcionado por el usuario y producir una secuencia de tokens que puedan ser utilizados por las etapas posteriores del compilador, como el analizador sintáctico y el analizador semántico. Estas etapas adicionales utilizarán los tokens para verificar la estructura y el significado del programa, asegurando que sea válido según las reglas del lenguaje y que cumpla con las expectativas del desarrollador, la implementación de un analizador léxico es un paso crucial en el diseño de un compilador. Su correcta ejecución no solo facilita el análisis posterior del código, sino que también permite detectar errores tempranos en el proceso de compilación, mejorando la calidad del software final y la experiencia del usuario.

## EXPLICACIÓN DEL CONTENIDO DE LA TABLA DE TOKENS

### ¿Qué es?

La tabla de tokens es una estructura fundamental en el desarrollo de un analizador léxico. Consiste en un listado de elementos léxicos (tokens), que representan los componentes básicos de un lenguaje de programación, junto con sus respectivos lexemas, que son las representaciones textuales de estos elementos en el código fuente.

### ¿Para qué sirve?

La tabla de tokens tiene múltiples propósitos dentro del proyecto de un compilador o intérprete, entre ellos:

- **Definir los componentes léxicos del lenguaje:** Agrupa operadores, palabras reservadas, símbolos, tipos de datos, y otros elementos que el analizador léxico debe reconocer.
- **Facilitar el reconocimiento del código fuente:** Permite transformar las cadenas de caracteres del programa en una secuencia de tokens, simplificando su procesamiento.
- **Estandarizar el análisis:** Cada token está asociado a una categoría semántica, ayudando a las etapas posteriores del compilador (sintáctica y semántica) a trabajar con datos consistentes y uniformes.

### Importancia dentro del proyecto:

En el contexto del desarrollo de un analizador léxico, la tabla de tokens es esencial por las siguientes razones:

1. **Estructuración del lenguaje:** Define de manera precisa los elementos permitidos en el lenguaje, estableciendo sus reglas y delimitaciones.
2. **Optimización del análisis:** Al categorizar cada elemento del código fuente, se reduce la complejidad del análisis sintáctico y semántico, permitiendo que el compilador trabaje de manera más eficiente.
3. **Prevención de errores:** Facilita la detección de errores léxicos, como el uso de elementos no válidos, y contribuye a generar mensajes de error más claros para el usuario.

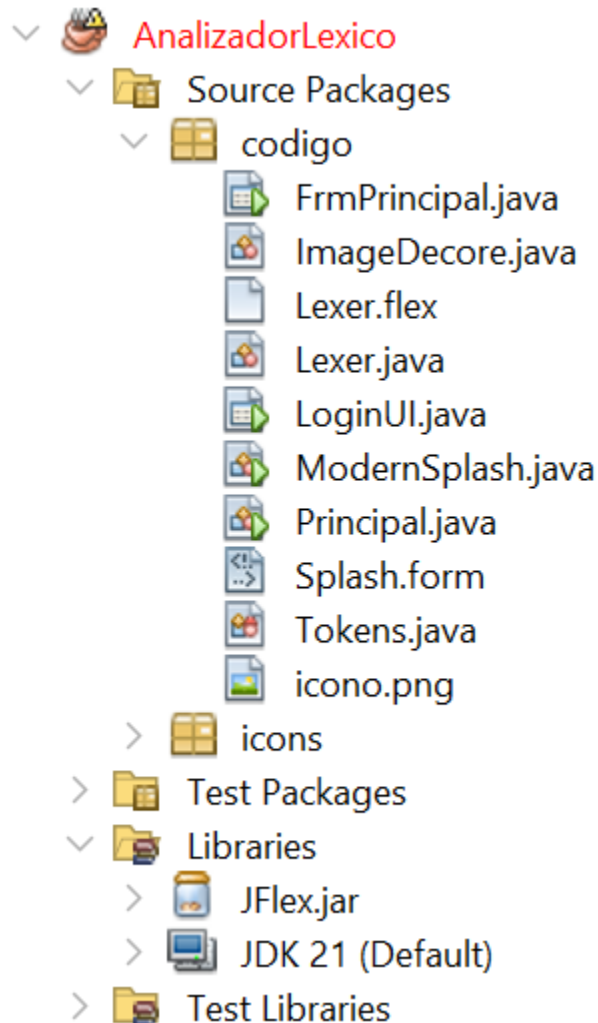


TABLA DE TOKENS			
	# TOKEN	TOKEN	LEXEMA
Operadores Binarios	1	operadorSuma	'+'
	2	operadorResta	'-'
	3	operadorMultiplicacion	'*'
	4	operadorDivision	'/'
	5	operadorModulo	'%'
	6	operadorPotencia	'^'
	7	operadorRaiz	'Rcuad'
Operador Unario	8	asignacion	'=>'
	9	incremento	'++'
	10	decremento	'--'
	11	operadorPositivo	'Pos'
Operadores Comparacion	12	operadorNegativo	'Neg'
	13	comparacionIgualdad	'=='
	14	mayorQue	'>'
	15	menorQue	'<'
	16	mayorIgual	'>=<'
Operador Logico	17	menorIgual	'<=>'
	18	operadorY	'&'
	19	operadorO	' '
	20	operadorNo	'!'
Alfabeto	21	letrasMin	'[a-z]'
	22	letrasMay	'[A-Z]'
	23	numero	'[0-9]'
Comentarios	24	inicioComentarioMult	'(:'
	25	finalComentarioMult	'):'
	26	comentarioLinea	')=)'
Tipos de Datos	27	valorEntero	'Ent'
	28	valorFlotante	'Flot'
	29	valorBooleano	'Boo'
	30	cadena	'Cad'
	31	carácter	'Cars'
Simbolos Especiales	32	finLinea	';-'
	33	saltoLinea	'\s'
	34	inicioTexto	'<<'
	35	finalTexto	'>>'
	36	parentIzq	'('
	37	parentDer	')'
	38	llaveIzq	'{'
	39	llaveDer	'}'
	40	corchetIzq	'['
	41	corchetDer	
	42	concatenar	'::'
	43	arreglo	'[]-[]'
	44	puntoAcceso	'.'
	45	puntoDecmial	'.'
	46	dosPuntos	'::'
	47	espacio	' '
	48	tabulacion	' '
	49	comillaDoble	'"'
	50	comillaSimple	'''
	51	coma	'.'
	52	guionBajo	'_'
	53	guionMedio	'_'
	54	diagonalInvertido	'\'
	55	diagonal	'/'
	56	signoAdmiracionAbre	'!'
	57	signoAdmiracionCierra	'!'
	58	gato	'#'
	59	pesos	'\$'



Palabras reservadas	60	condicionalif	'si'
	61	condicionalElse	'siNo'
	62	cicloFor	'ciclo'
	63	cicloWhile	'while'
	64	cicloDo	'hacer'
	65	funcionSwitch	'segun'
	66	funcionCase	'caso'
	67	funcionSalirCase	'salir'
	68	predeterminado	'predeterminado'
	69	detener	'Detener'
	70	constante	'Constante'
	71	variable	'Variable'
	72	funcion	'funcion'
	73	clase	'Clase'
	74	imprimir	'Imprimir'
	75	importar	'Importar'
	76	funcionLeer	'Leer'
	77	funcionIntroducirDatos	'IntroducirD'
Constantes	78	pi	'pi'
	79	euler	'E'

## Estructura Del Proyecto





## DESCRIPCIÓN DE CADA UNO DE LOS ARCHIVOS GENERADOS EN EL PROYECTO

### Archivo Lexer.flex

El archivo Lexer.flex es un componente clave en el desarrollo de un compilador o intérprete. Este archivo define el analizador léxico, una herramienta que descompone el código fuente en unidades fundamentales llamadas *tokens*. Un *token* es una representación simbólica de elementos como operadores, palabras reservadas, identificadores, números, o símbolos especiales.

El propósito principal de este archivo es procesar y clasificar el código fuente de manera sistemática, eliminando elementos irrelevantes como espacios y comentarios, para facilitar el análisis sintáctico en las etapas posteriores del procesamiento del lenguaje. Además, al manejar errores mediante el token ERROR, el analizador contribuye a detectar entradas no válidas en el código, mejorando la eficiencia del sistema.

Este archivo actúa como el primer filtro del procesamiento del lenguaje, estableciendo una base sólida para que el resto del sistema interprete o compile el código fuente de manera eficiente y precisa. Es esencial para automatizar el reconocimiento de los elementos del lenguaje y garantizar que el código fuente sea correctamente comprendido, minimizando errores y facilitando el desarrollo general del proyecto.

#### Código Realizado:

```
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
14 /* Espacios en blanco */
15 {espacio} { /*Ignore*/ }
16
17 /* Comentarios */
18 "/*".* { /*Ignore*/ }
19
20 /* Salto de línea */
21 "\n" { return Línea; }
22
23 /* Comillas */
24 "\"" { lexeme=yytext(); return comillaDoble; }
25 "'" { lexeme=yytext(); return comillaSimple; }
26
27 /* Tipos de datos */
28 "Ent" { lexeme=yytext(); return valorEntero; }
29 "Flot" { lexeme=yytext(); return valorFlotante; }
30 "Boo" { lexeme=yytext(); return valorBooleano; }
31 "Cad" { lexeme=yytext(); return cadena; }
32 "Cars" { lexeme=yytext(); return carácter; }
33
```



```
34  /* Palabras reservadas */
35  "si" {lexeme=yytext(); return condicionalIf;}
36  "siNo" {lexeme=yytext(); return condicionalElse;}
37  "ciclo" {lexeme=yytext(); return cicloFor;}
38  "mientras" {lexeme=yytext(); return cicloWhile;}
39  "hacer" {lexeme=yytext(); return cicloDo;}
40  "segun" {lexeme=yytext(); return funcionSwitch;}
41  "caso" {lexeme=yytext(); return funcionCase;}
42  "salir" {lexeme=yytext(); return funcionSalirCase;}
43  "predeterminado" {lexeme=yytext(); return predeterminado;}
44  "Detener" {lexeme=yytext(); return detener;}
45  "Constante" {lexeme=yytext(); return constante;}
46  "Variable" {lexeme=yytext(); return variable;}
47  "funcion" {lexeme=yytext(); return funcion;}
48  "Clase" {lexeme=yytext(); return clase;}
49  "imprimir" {lexeme=yytext(); return imprimir;}
50  "Leer" {lexeme=yytext(); return funcionLeer;}
51  "IntroducirD" {lexeme=yytext(); return funcionIntroducirDatos;}
52
53  /* Operadores matemáticos */
54  "+" {lexeme=yytext(); return operadorSuma;}
55  "-" {lexeme=yytext(); return operadorResta;}
56  "*" {lexeme=yytext(); return operadorMultiplicacion;}
57  "/" {lexeme=yytext(); return operadorDivision;}
58  "%" {lexeme=yytext(); return operadorModulo;}
59  "^" {lexeme=yytext(); return operadorPotencia;}
60  "Rcuad" {lexeme=yytext(); return operadorRaiz;}
61
62  /* Operadores de asignación y comparación */
63  "=>" {lexeme=yytext(); return asignacion;}
64  "==" {lexeme=yytext(); return comparacionIgualdad;}
65  ">" {lexeme=yytext(); return mayorQue;}
66  "<" {lexeme=yytext(); return menorQue;}
67
68  "<" {lexeme=yytext(); return menorQue;}
69  ">=" {lexeme=yytext(); return mayorIgual;}
70  "<=" {lexeme=yytext(); return menorIgual;}
71
72  /* Operadores lógicos */
73  "&" {lexeme=yytext(); return operadorY;}
74  "|" {lexeme=yytext(); return operadorO;}
75  "!" {lexeme=yytext(); return operadorNo;}
76
77  /* Operadores incrementales */
78  "++" {lexeme=yytext(); return incremento;}
79  "--" {lexeme=yytext(); return decremento;}
80
81  /* Caracteres especiales */
82  "(" {lexeme=yytext(); return parentIzq;}
83  ")" {lexeme=yytext(); return parentDer;}
84  "[" {lexeme=yytext(); return llaveIzq;}
85  "]" {lexeme=yytext(); return llaveDer;}
86  "{" {lexeme=yytext(); return corchetIzq;}
87  "}" {lexeme=yytext(); return corchetDer;}
88  ";" {lexeme=yytext(); return finLinea;}
89  "<<" {lexeme=yytext(); return inicioTexto;}
90  ">>" {lexeme=yytext(); return finalTexto;}
91  "." {lexeme=yytext(); return puntoDecimal;}
92  ":" {lexeme=yytext(); return dosPuntos;}
93  " " {lexeme=yytext(); return espacio;}
94
95  /* Identificadores y números */
96  {L}({L}|{D})* {lexeme=yytext(); return Identificador;}
97  {"-"({D}+)"|{D}+ {lexeme=yytext(); return Numero;}
98
99  /* Errores */
100  . {return ERROR;}
```

## Archivo Lexer.java (Generado automáticamente)

El archivo Lexer.java es una clase generada automáticamente por la herramienta JFlex, diseñada para construir un analizador léxico o escáner. Este archivo se encarga de identificar los distintos componentes léxicos o tokens de un lenguaje definido en una especificación, en este caso el archivo Lexer.flex. Los tokens pueden representar palabras clave, identificadores, operadores, literales, y otros elementos sintácticos básicos del lenguaje.

El analizador léxico es una parte esencial de un compilador o intérprete, ya que actúa como el primer paso en el análisis del código fuente. Este archivo tiene la responsabilidad de leer el flujo de entrada de caracteres y convertirlo en una secuencia organizada de tokens que el analizador sintáctico puede entender. Su importancia radica en garantizar que el código fuente se procese de manera eficiente y que cualquier error léxico se detecte desde etapas tempranas del análisis. Este proceso es fundamental para garantizar la correcta interpretación y ejecución del código fuente.

### Código Generado:

```
1  /* The following code was generated by JFlex 1.4.3 on 01/01/25, 12:03 */
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 01/01/25, 12:03 from the specification file
10  * <tt>C:/Users/jesus/Downloads/AnalizadorLexico/src/codigo/Lexer.flex</tt>
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[l] is the state in the DFA for the lexical state l
25       * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
26       * at the beginning of a line
27       * l is of the form l = 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 =
37      "\1\0\1\3\1\16\2\0\1\3\2\0\1\4\7\1\42\1\65\1\71"+
38      "\1\60\1\22\1\40\1\66\1\45\1\50\1\21\1\17\1\3\1\20"+
39      "\1\64\1\15\1\2\1\46\1\57\1\37\1\31\1\32\2\0\1\44"+
40      "\1\54\1\55\1\74\1\52\1\53\2\44\1\77\2\44\1\100\1\44"+
41      "\1\35\1\44\1\33\1\44\1\24\3\44\1\75\4\44\1\62\1\67"+
42      "\1\63\1\23\1\1\1\0\1\27\1\76\1\25\1\30\1\10\1\7"+
43      "\1\36\1\14\1\4\2\43\1\11\1\72\1\5\1\34\1\73\1\43"+
44      "\1\56\1\12\1\6\1\26\1\43\1\13\3\43\1\61\1\41\1\51"+
45      "\43\0\1\70\uff5e\0";
46
47  /**
48   * Translates characters to character classes
49   */
50  private static final char [] ZZ_CMAP = zzUnpackCMap(ZZ_CMAP_PACKED);
51
52  /**
53   * Translates DFA states to action switch labels.
54   */
55  private static final int [] ZZ_ACTION = zzUnpackAction();
56
57  private static final String ZZ_ACTION_PACKED_0 =
58      "\1\0\1\1\1\2\1\3\1\4\7\5\1\6\1\7"+
59      "\1\10\1\11\1\12\1\13\1\14\1\5\1\1\1\15"+
60      "\2\14\1\16\1\17\1\20\1\21\1\14\1\22\1\23"+
61      "\1\24\1\25\4\14\1\1\1\26\1\27\1\30\1\31"+
62      "\1\32\1\33\1\34\1\35\1\36\1\37\2\5\4\14"+
63      "\1\40\1\41\1\40\1\42\2\40\1\43\4\40\1\4"+
64      "\1\44\1\45\3\40\1\46\1\47\1\0\1\50\1\40"+
65      "\1\51\1\40\1\52\3\0\1\53\1\0\6\40"+
66      "\1\0\1\54\1\0\2\1\40\1\55\1\56\1\57\1\60"+

```

```

67      "\1\0\1\61\1\62\1\63\1\64\1\40\1\65\1\40"+
68      "\1\66\2\40\1\67\1\0\10\40\1\70\5\40\1\71"+
69      "\1\41\1\72\1\40\1\73\1\40\1\0\7\40\1\74"+
70      "\1\40\1\75\1\76\1\77\1\100\1\101\1\102\1\40"+
71      "\1\103\20\40\1\104\3\40\1\105\5\40\1\106\1\40"+
72      "\1\107\1\40\1\110\1\111\1\112\5\40\1\113\2\40"+
73      "\1\114";
74
75  private static int [] zzUnpackAction() {
76      int [] result = new int[208];
77      int offset = 0;
78      offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);
79      return result;
80  }
81
82  private static int zzUnpackAction(String packed, int offset, int [] result) {
83      int i = 0; /* index in packed string */
84      int j = offset; /* index in unpacked array */
85      int l = packed.length();
86      while (i < l) {
87          int count = packed.charAt(i++);
88          int value = packed.charAt(i++);
89          do result[j++] = value; while (--count > 0);
90      }
91      return j;
92  }
93
94
95  /**
96   * Translates a state to a row index in the transition table
97   */
98  private static final int [] ZZ_ROWMAP = zzUnpackRowMap();
99

```



```
100 private static final String ZZ_ROWMAP_PACKED_0 =
101     "\0\0\0\101\0\202\0\303\0\0104\0\0145\0\202\0\0186"+
102     "\0\01c7\0\0208\0\0249\0\028a\0\02cb\0\030c\0\034d\0\101"+
103     "\0\101\0\101\0\038e\0\03cf\0\0410\0\0451\0\0492\0\04d3"+
104     "\0\0514\0\101\0\101\0\101\0\202\0\0555\0\0596\0\101"+
105     "\0\05d7\0\0618\0\0659\0\069a\0\06db\0\071c\0\075d\0\101"+
106     "\0\079e\0\101\0\101\0\101\0\101\0\101\0\101\0\101"+
107     "\0\07df\0\0820\0\0861\0\08a2\0\08e3\0\0924\0\202\0\303"+
108     "\0\0965\0\202\0\09a6\0\09e7\0\0a28\0\0a69\0\0aaa\0\0aeb"+
109     "\0\0b2c\0\0b6d\0\101\0\101\0\0bae\0\0bef\0\0c30\0\101"+
110     "\0\101\0\0c71\0\101\0\0cb2\0\202\0\0cf3\0\0d34\0\101"+
111     "\0\0d75\0\0db6\0\0df7\0\101\0\0e38\0\0e79\0\0eba\0\0efb"+
112     "\0\0f3c\0\0f7d\0\0f8e\0\0fff\0\101\0\1040\0\1081\0\10c2"+
113     "\0\1103\0\1144\0\1185\0\11c6\0\1207\0\1248\0\1289\0\12ca"+
114     "\0\130b\0\134c\0\138d\0\13ce\0\140f\0\1450\0\1491\0\101"+
115     "\0\202\0\202\0\101\0\14d2\0\101\0\101\0\101\0\202"+
116     "\0\1513\0\202\0\1554\0\202\0\1595\0\15d6\0\101\0\1617"+
117     "\0\1658\0\1699\0\16da\0\171b\0\175c\0\179d\0\17de\0\181f"+
118     "\0\202\0\1860\0\18a1\0\18e2\0\1923\0\1964\0\202\0\101"+
119     "\0\202\0\19a5\0\202\0\19e6\0\1a27\0\1a68\0\1aa9\0\1aea"+
120     "\0\1b2b\0\1b6c\0\1bad\0\1bee\0\202\0\1c2f\0\202\0\202"+
121     "\0\202\0\202\0\202\0\202\0\1c70\0\101\0\1cb1\0\1cf2"+
122     "\0\1d33\0\1d74\0\1db5\0\1df6\0\1e37\0\1e78\0\1eb9\0\1efa"+
123     "\0\1f3b\0\1f7c\0\1fbd\0\1ffe\0\203f\0\2080\0\202\0\20c1"+
124     "\0\2102\0\2143\0\202\0\2184\0\21c5\0\2206\0\2247\0\2288"+
125     "\0\202\0\22c9\0\202\0\230a\0\202\0\202\0\202\0\234b"+
126     "\0\238c\0\23cd\0\240e\0\244f\0\202\0\2490\0\24d1\0\202";
127
128 private static int [] zzUnpackRowMap() {
129     int [] result = new int[208];
130     int offset = 0;
131     offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);
132     return result;
```

```
133 }
134
135 private static int zzUnpackRowMap(String packed, int offset, int [] result) {
136     int i = 0; /* index in packed string */
137     int j = offset; /* index in unpacked array */
138     int l = packed.length();
139     while (i < l) {
140         int high = packed.charAt(i++) << 16;
141         result[j++] = high | packed.charAt(i++);
142     }
143     return j;
144 }
145
146 /**
147  * The transition table of the DFA
148  */
149 private static final int [] ZZ_TRANS = zzUnpackTrans();
150
151 private static final String ZZ_TRANS_PACKED_0 =
152     "\1\2\1\3\1\4\1\5\1\6\1\7\1\10\1\11"+
153     "\1\12\1\13\1\14\1\15\1\16\1\17"+
154     "\1\20\1\21\1\22\1\23\1\24\1\25\1\26"+
155     "\1\27\1\28\1\29\1\30\1\31\1\32\1\33\1\34"+
156     "\1\35\1\36\1\37\1\38\1\39\1\40\1\41\1\42"+
157     "\1\43\1\44\1\45\1\46\1\47\1\48\1\49\1\50\1\51"+
158     "\1\52\1\53\1\54\1\55\1\56\1\57\1\58\1\59\1\60\1\61"+
159     "\1\62\1\63\1\64\1\65\1\66\1\67\1\68\1\69\1\70\1\71\1\72\1\73\1\74\1\75\1\76\1\77\1\78\1\79\1\80\1\81\1\82\1\83\1\84\1\85\1\86\1\87\1\88\1\89\1\90\1\91\1\92\1\93\1\94\1\95\1\96\1\97\1\98\1\99\1\100\1\101\1\102\1\103\1\104\1\105\1\106\1\107\1\108\1\109\1\110\1\111\1\112\1\113\1\114\1\115\1\116\1\117\1\118\1\119\1\120\1\121\1\122\1\123\1\124\1\125\1\126\1\127\1\128\1\129\1\130\1\131\1\132\1\133\1\134\1\135\1\136\1\137\1\138\1\139\1\140\1\141\1\142\1\143\1\144\1\145\1\146\1\147\1\148\1\149\1\150\1\151\1\152\1\153\1\154\1\155\1\156\1\157\1\158\1\159\1\160\1\161\1\162\1\163\1\164\1\165\1\166\1\167\1\168\1\169\1\170\1\171\1\172\1\173\1\174\1\175\1\176\1\177\1\178\1\179\1\180\1\181\1\182\1\183\1\184\1\185\1\186\1\187\1\188\1\189\1\190\1\191\1\192\1\193\1\194\1\195\1\196\1\197\1\198\1\199\1\200\1\201\1\202\1\203\1\204\1\205\1\206\1\207\1\208\1\209\1\210\1\211\1\212\1\213\1\214\1\215\1\216\1\217\1\218\1\219\1\220\1\221\1\222\1\223\1\224\1\225\1\226\1\227\1\228\1\229\1\230\1\231\1\232\1\233\1\234\1\235\1\236\1\237\1\238\1\239\1\240\1\241\1\242\1\243\1\244\1\245\1\246\1\247\1\248\1\249\1\250\1\251\1\252\1\253\1\254\1\255\1\256\1\257\1\258\1\259\1\260\1\261\1\262\1\263\1\264\1\265\1\266\1\267\1\268\1\269\1\270\1\271\1\272\1\273\1\274\1\275\1\276\1\277\1\278\1\279\1\280\1\281\1\282\1\283\1\284\1\285\1\286\1\287\1\288\1\289\1\290\1\291\1\292\1\293\1\294\1\295\1\296\1\297\1\298\1\299\1\300\1\301\1\302\1\303\1\304\1\305\1\306\1\307\1\308\1\309\1\310\1\311\1\312\1\313\1\314\1\315\1\316\1\317\1\318\1\319\1\320\1\321\1\322\1\323\1\324\1\325\1\326\1\327\1\328\1\329\1\330\1\331\1\332\1\333\1\334\1\335\1\336\1\337\1\338\1\339\1\340\1\341\1\342\1\343\1\344\1\345\1\346\1\347\1\348\1\349\1\350\1\351\1\352\1\353\1\354\1\355\1\356\1\357\1\358\1\359\1\360\1\361\1\362\1\363\1\364\1\365\1\366\1\367\1\368\1\369\1\370\1\371\1\372\1\373\1\374\1\375\1\376\1\377\1\378\1\379\1\380\1\381\1\382\1\383\1\384\1\385\1\386\1\387\1\388\1\389\1\390\1\391\1\392\1\393\1\394\1\395\1\396\1\397\1\398\1\399\1\400\1\401\1\402\1\403\1\404\1\405\1\406\1\407\1\408\1\409\1\410\1\411\1\412\1\413\1\414\1\415\1\416\1\417\1\418\1\419\1\420\1\421\1\422\1\423\1\424\1\425\1\426\1\427\1\428\1\429\1\430\1\431\1\432\1\433\1\434\1\435\1\436\1\437\1\438\1\439\1\440\1\441\1\442\1\443\1\444\1\445\1\446\1\447\1\448\1\449\1\450\1\451\1\452\1\453\1\454\1\455\1\456\1\457\1\458\1\459\1\460\1\461\1\462\1\463\1\464\1\465\1\466\1\467\1\468\1\469\1\470\1\471\1\472\1\473\1\474\1\475\1\476\1\477\1\478\1\479\1\480\1\481\1\482\1\483\1\484\1\485\1\486\1\487\1\488\1\489\1\490\1\491\1\492\1\493\1\494\1\495\1\496\1\497\1\498\1\499\1\500\1\501\1\502\1\503\1\504\1\505\1\506\1\507\1\508\1\509\1\510\1\511\1\512\1\513\1\514\1\515\1\516\1\517\1\518\1\519\1\520\1\521\1\522\1\523\1\524\1\525\1\526\1\527\1\528\1\529\1\530\1\531\1\532\1\533\1\534\1\535\1\536\1\537\1\538\1\539\1\540\1\541\1\542\1\543\1\544\1\545\1\546\1\547\1\548\1\549\1\550\1\551\1\552\1\553\1\554\1\555\1\556\1\557\1\558\1\559\1\560\1\561\1\562\1\563\1\564\1\565\1\566\1\567\1\568\1\569\1\570\1\571\1\572\1\573\1\574\1\575\1\576\1\577\1\578\1\579\1\580\1\581\1\582\1\583\1\584\1\585\1\586\1\587\1\588\1\589\1\590\1\591\1\592\1\593\1\594\1\595\1\596\1\597\1\598\1\599\1\600\1\601\1\602\1\603\1\604\1\605\1\606\1\607\1\608\1\609\1\610\1\611\1\612\1\613\1\614\1\615\1\616\1\617\1\618\1\619\1\620\1\621\1\622\1\623\1\624\1\625\1\626\1\627\1\628\1\629\1\630\1\631\1\632\1\633\1\634\1\635\1\636\1\637\1\638\1\639\1\640\1\641\1\642\1\643\1\644\1\645\1\646\1\647\1\648\1\649\1\650\1\651\1\652\1\653\1\654\1\655\1\656\1\657\1\658\1\659\1\660\1\661\1\662\1\663\1\664\1\665\1\666\1\667\1\668\1\669\1\670\1\671\1\672\1\673\1\674\1\675\1\676\1\677\1\678\1\679\1\680\1\681\1\682\1\683\1\684\1\685\1\686\1\687\1\688\1\689\1\690\1\691\1\692\1\693\1\694\1\695\1\696\1\697\1\698\1\699\1\700\1\701\1\702\1\703\1\704\1\705\1\706\1\707\1\708\1\709\1\710\1\711\1\712\1\713\1\714\1\715\1\716\1\717\1\718\1\719\1\720\1\721\1\722\1\723\1\724\1\725\1\726\1\727\1\728\1\729\1\730\1\731\1\732\1\733\1\734\1\735\1\736\1\737\1\738\1\739\1\740\1\741\1\742\1\743\1\744\1\745\1\746\1\747\1\748\1\749\1\750\1\751\1\752\1\753\1\754\1\755\1\756\1\757\1\758\1\759\1\760\1\761\1\762\1\763\1\764\1\765\1\766\1\767\1\768\1\769\1\770\1\771\1\772\1\773\1\774\1\775\1\776\1\777\1\778\1\779\1\780\1\781\1\782\1\783\1\784\1\785\1\786\1\787\1\788\1\789\1\790\1\791\1\792\1\793\1\794\1\795\1\796\1\797\1\798\1\799\1\800\1\801\1\802\1\803\1\804\1\805\1\806\1\807\1\808\1\809\1\810\1\811\1\812\1\813\1\814\1\815\1\816\1\817\1\818\1\819\1\820\1\821\1\822\1\823\1\824\1\825\1\826\1\827\1\828\1\829\1\830\1\831\1\832\1\833\1\834\1\835\1\836\1\837\1\838\1\839\1\840\1\841\1\842\1\843\1\844\1\845\1\846\1\847\1\848\1\849\1\850\1\851\1\852\1\853\1\854\1\855\1\856\1\857\1\858\1\859\1\860\1\861\1\862\1\863\1\864\1\865\1\866\1\867\1\868\1\869\1\870\1\871\1\872\1\873\1\874\1\875\1\876\1\877\1\878\1\879\1\880\1\881\1\882\1\883\1\884\1\885\1\886\1\887\1\888\1\889\1\890\1\891\1\892\1\893\1\894\1\895\1\896\1\897\1\898\1\899\1\900\1\901\1\902\1\903\1\904\1\905\1\906\1\907\1\908\1\909\1\910\1\911\1\912\1\913\1\914\1\915\1\916\1\917\1\918\1\919\1\920\1\921\1\922\1\923\1\924\1\925\1\926\1\927\1\928\1\929\1\930\1\931\1\932\1\933\1\934\1\935\1\936\1\937\1\938\1\939\1\940\1\941\1\942\1\943\1\944\1\945\1\946\1\947\1\948\1\949\1\950\1\951\1\952\1\953\1\954\1\955\1\956\1\957\1\958\1\959\1\960\1\961\1\962\1\963\1\964\1\965\1\966\1\967\1\968\1\969\1\970\1\971\1\972\1\973\1\974\1\975\1\976\1\977\1\978\1\979\1\980\1\981\1\982\1\983\1\984\1\985\1\986\1\987\1\988\1\989\1\990\1\991\1\992\1\993\1\994\1\995\1\996\1\997\1\998\1\999\1\1000\1\1001\1\1002\1\1003\1\1004\1\1005\1\1006\1\1007\1\1008\1\1009\1\1010\1\1011\1\1012\1\1013\1\1014\1\1015\1\1016\1\1017\1\1018\1\1019\1\1020\1\1021\1\1022\1\1023\1\1024\1\1025\1\1026\1\1027\1\1028\1\1029\1\1030\1\1031\1\1032\1\1033\1\1034\1\1035\1\1036\1\1037\1\1038\1\1039\1\1040\1\1041\1\1042\1\1043\1\1044\1\1045\1\1046\1\1047\1\1048\1\1049\1\1050\1\1051\1\1052\1\1053\1\1054\1\1055\1\1056\1\1057\1\1058\1\1059\1\1060\1\1061\1\1062\1\1063\1\1064\1\1065\1\1066\1\1067\1\1068\1\1069\1\1070\1\1071\1\1072\1\1073\1\1074\1\1075\1\1076\1\1077\1\1078\1\1079\1\1080\1\1081\1\1082\1\1083\1\1084\1\1085\1\1086\1\1087\1\1088\1\1089\1\1090\1\1091\1\1092\1\1093\1\1094\1\1095\1\1096\1\1097\1\1098\1\1099\1\1100\1\1101\1\1102\1\1103\1\1104\1\1105\1\1106\1\1107\1\1108\1\1109\1\1110\1\1111\1\1112\1\1113\1\1114\1\1115\1\1116\1\1117\1\1118\1\1119\1\1120\1\1121\1\1122\1\1123\1\1124\1\1125\1\1126\1\1127\1\1128\1\1129\1\1130\1\1131\1\1132\1\1133\1\1134\1\1135\1\1136\1\1137\1\1138\1\1139\1\1140\1\1141\1\1142\1\1143\1\1144\1\1145\1\1146\1\1147\1\1148\1\1149\1\1150\1\1151\1\1152\1\1153\1\1154\1\1155\1\1156\1\1157\1\1158\1\1159\1\1160\1\1161\1\1162\1\1163\1\1164\1\1165\1\1166\1\1167\1\1168\1\1169\1\1170\1\1171\1\1172\1\1173\1\1174\1\1175\1\1176\1\1177\1\1178\1\1179\1\1180\1\1181\1\1182\1\1183\1\1184\1\1185\1\1186\1\1187\1\1188\1\1189\1\1190\1\1191\1\1192\1\1193\1\1194\1\1195\1\1196\1\1197\1\1198\1\1199\1\1200\1\1201\1\1202\1\1203\1\1204\1\1205\1\1206\1\1207\1\1208\1\1209\1\1210\1\1211\1\1212\1\1213\1\1214\1\1215\1\1216\1\1217\1\1218\1\1219\1\1220\1\1221\1\1222\1\1223\1\1224\1\1225\1\1226\1\1227\1\1228\1\1229\1\1230\1\1231\1\1232\1\1233\1\1234\1\1235\1\1236\1\1237\1\1238\1\1239\1\1240\1\1241\1\1242\1\1243\1\1244\1\1245\1\1246\1\1247\1\1248\1\1249\1\1250\1\1251\1\1252\1\1253\1\1254\1\1255\1\1256\1\1257\1\1258\1\1259\1\1260\1\1261\1\1262\1\1263\1\1264\1\1265\1\1266\1\1267\1\1268\1\1269\1\1270\1\1271\1\1272\1\1273\1\1274\1\1275\1\1276\1\1277\1\1278\1\1279\1\1280\1\1281\1\1282\1\1283\1\1284\1\1285\1\1286\1\1287\1\1288\1\1289\1\1290\1\1291\1\1292\1\1293\1\1294\1\1295\1\1296\1\1297\1\1298\1\1299\1\1300\1\1301\1\1302\1\1303\1\1304\1\1305\1\1306\1\1307\1\1308\1\1309\1\1310\1\1311\1\1312\1\1313\1\1314\1\1315\1\1316\1\1317\1\1318\1\1319\1\1320\1\1321\1\1322\1\1323\1\1324\1\1325\1\1326\1\1327\1\1328\1\1329\1\1330\1\1331\1\1332\1\1333\1\1334\1\1335\1\1336\1\1337\1\1338\1\1339\1\1340\1\1341\1\1342\1\1343\1\1344\1\1345\1\1346\1\1347\1\1348\1\1349\1\1350\1\1351\1\1352\1\1353\1\1354\1\1355\1\1356\1\1357\1\1358\1\1359\1\1360\1\1361\1\1362\1\1363\1\1364\1\1365\1\1366\1\1367\1\1368\1\1369\1\1370\1\1371\1\1372\1\1373\1\1374\1\1375\1\1376\1\1377\1\1378\1\1379\1\1380\1\1381\1\1382\1\1383\1\1384\1\1385\1\1386\1\1387\1\1388\1\1389\1\1390\1\1391\1\1392\1\1393\1\1394\1\1395\1\1396\1\1397\1\1398\1\1399\1\1400\1\1401\1\1402\1\1403\1\1404\1\1405\1\1406\1\1407\1\1408\1\1409\1\1410\1\1411\1\1412\1\1413\1\1414\1\1415\1\1416\1\1417\1\1418\1\1419\1\1420\1\1421\1\1422\1\1423\1\1424\1\1425\1\1426\1\1427\1\1428\1\1429\1\1430\1\1431\1\1432\1\1433\1\1434\1\1435\1\1436\1\1437\1\1438\1\1439\1\1440\1\1441\1\1442\1\1443\1\1444\1\1445\1\1446\1\1447\1\1448\1\1449\1\1450\1\1451\1\1452\1\1453\1\1454\1\1455\1\1456\1\1457\1\1458\1\1459\1\1460\1\1461\1\1462\1\1463\1\1464\1\1465\1\1466\1\1467\1\1468\1\1469\1\1470\1\1471\1\1472\1\1473\1\1474\1\1475\1\1476\1\1477\1\1478\1\1479\1\1480\1\1481\1\1482\1\1483\1\1484\1\1485\1\1486\1\1487\1\1488\1\1489\1\1490\1\1491\1\1492\1\1493\1\1494\1\1495\1\1496\1\1497\1\1498\1\1499\1\1500\1\1501\1\1502\1\1503\1\1504\1\1505\1\1506\1\1507\1\1508\1\1509\1\1510\1\1511\1\1512\1\1513\1\1514\1\1515\1\1516\1\1517\1\1518\1\1519\1\1520\1\1521\1\1522\1\1523\1\1524\1\1525\1\1526\1\1527\1\1528\1\1529\1\1530\1\1531\1\1532\1\1533\1\1534\1\1535\1\1536\1\1537\1\1538\1\1539\1\1540\1\1541\1\1542\1\1543\1\1544\1\1545\1\1546\1\1547\1\1548\1\1549\1\1550\1\1551\1\1552\1\1553\1\1554\1\1555\1\1556\1\1557\1\1558\1\1559\1\1560\1\1561\1\1562\1\1563\1\1564\1\1565\1\1566\1\1567\1\1568\1\1569\1\1570\1\1571\1\1572\1\1573\1\1574\1\1575\1\1576\1\1577\1\1578\1\1579\1\1580\1\1581\1\1582\1\1583\1\1584\1\1585\1\1586\1\1587\1\1588\1\1589\1\1590\1\1591\1\1592\1\1593\1\1594\1\1595\1\1596\1\1597\1\1598\1\1599\1\1600\1\1601\1\1602\1\1603\1\1604\1\1605\1\1606\1\1607\1\1608\1\1609\1\1610\1\1611\1\1612\1\1613\1\1614\1\1615\1\1616\1\1617\1\1618\1\1619\1\1620\1\1621\1\1622\1\1623\1\1624\1\1625\1\1626\1\1627\1\1628\1\1629\1\1630\1\1631\1\1632\1\1633\1\1634\1\163
```



166	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
167	"\1\0\5\67\1\74\3\67\7\0\5\67\2\0\4\67"+	
168	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
169	"\1\0\1\75\3\67\1\76\4\67\7\0\3\67\1\77"+	
170	"\1\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+	
171	"\7\67\1\0\2\67\1\0\1\0\67\1\100\7\0\5\67"+	
172	"\2\0\4\67\4\0\2\67\5\0\5\67\13\0\7\67"+	
173	"\1\0\2\67\1\0\11\67\7\0\3\67\1\101\1\67"+	
174	"\2\0\4\67\4\0\2\67\5\0\5\67\13\0\7\67"+	
175	"\15\0\1\102\102\0\1\103\101\0\1\104\61\0\2\67"+	
176	"\1\0\11\67\7\0\1\67\1\105\3\67\2\0\4\67"+	
177	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
178	"\1\0\1\106\10\67\7\0\3\67\1\107\1\67\2\0"+	
179	"\4\67\4\0\2\67\5\0\5\67\13\0\7\67\3\0"+	
180	"\1\110\1\111\7\0\1\112\1\113\4\7\0\2\67\1\0"+	
181	"\11\67\7\0\5\67\2\0\1\67\1\114\2\67\4\0"+	
182	"\2\67\5\0\5\67\13\0\5\67\1\115\1\67\1\0"+	
183	"\2\67\1\0\4\67\1\116\4\67\7\0\5\67\2\0"+	
184	"\4\67\4\0\2\67\5\0\5\67\13\0\7\67\3\0"+	
185	"\1\117\5\0\1\120\6\1\0\1\121\25\0\1\122\5\2\0"+	
186	"\1\123\2\7\0\1\124\6\1\0\1\125\5\0\2\67\1\0"+	
187	"\1\67\1\126\7\67\7\0\5\67\2\0\4\67\4\0"+	
188	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
189	"\5\67\1\127\3\67\7\0\5\67\2\0\4\67\4\0"+	
190	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
191	"\1\1\67\7\0\5\67\2\0\1\67\1\130\2\67\4\0"+	
192	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
193	"\5\67\1\131\3\67\7\0\3\67\1\132\1\67\2\0"+	
194	"\1\67\1\133\2\67\4\0\2\67\5\0\5\67\13\0"+	
195	"\7\67\2\0\0\1\134\7\1\0\1\135\152\0\1\136\16\0"+	
196	"\2\67\1\0\1\137\1\0\1\67\7\0\5\67\2\0\4\67"+	
197	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
198	"\1\0\1\1\67\7\0\5\67\2\0\4\67\4\0\2\67"+	
199	"\5\0\4\67\1\140\13\0\7\67\1\0\2\67\1\0"+	
200	"\4\67\1\141\4\67\7\0\5\67\2\0\4\67\4\0"+	
201	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
202	"\1\1\67\7\0\3\67\1\142\1\67\2\0\4\67\4\0"+	
203	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
204	"\1\67\1\143\7\67\7\0\5\67\2\0\4\67\4\0"+	
205	"\2\67\5\0\5\67\13\0\1\144\6\67\1\0\2\67"+	
206	"\1\0\4\67\1\145\4\67\7\0\5\67\2\0\4\67"+	
207	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
208	"\1\0\2\67\1\72\6\67\7\0\5\67\2\0\4\67"+	
209	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
210	"\1\0\1\67\1\146\7\67\7\0\5\67\2\0\4\67"+	
211	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
212	"\1\0\6\67\1\147\2\67\7\0\5\67\2\0\4\67"+	
213	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
214	"\1\0\1\1\67\7\0\5\67\2\0\2\67\1\150\1\67"+	
215	"\4\0\2\67\5\0\5\67\13\0\7\67\1\0\2\67"+	
216	"\1\0\1\1\67\7\0\5\67\2\0\3\67\1\151\1\0"+	
217	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
218	"\5\67\1\152\3\67\7\0\5\67\2\0\4\67\4\0"+	
219	"\2\67\5\0\5\67\13\0\7\67\1\0\2\67\1\0"+	
220	"\1\153\10\67\7\0\5\67\2\0\4\67\4\0\2\67"+	
221	"\5\0\5\67\13\0\7\67\1\0\2\67\1\0\1\1\67"+	
222	"\7\0\1\67\1\154\3\67\2\0\4\67\4\0\2\67"+	
223	"\5\0\5\67\13\0\7\67\1\6\102\1\0\6\2\102\1\0"+	
224	"\2\67\1\0\1\1\67\7\0\2\67\1\155\2\67\2\0"+	
225	"\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+	
226	"\2\67\1\0\1\1\67\7\0\1\67\1\156\3\67\2\0"+	
227	"\4\67\4\0\2\67\5\0\5\67\13\0\7\67\1\0"+	
228	"\2\67\1\0\6\67\1\157\2\67\7\0\5\67\2\0"+	
229	"\4\67\4\0\2\67\5\0\5\67\13\0\7\67\3\7\0"+	
230	"\1\160\14\2\0\2\67\1\0\1\67\1\161\2\67\7\0"+	
231	"\5\67\2\0\4\67\4\0\2\67\5\0\5\67\13\0"+	



232	"\17\67\110\12\67\110\11\67\17\0\5\67\12\0"+	
233	"\13\67\11\162\4\0\12\67\15\0\5\67\113\0\7\67"+	
234	"\132\0\1\163\5\0\0\1\164\145\0\1\165\7\0\1\166"+	
235	"\102\0\1\167\13\0\12\67\11\0\12\67\11\70\6\67"+	
236	"\17\0\1\5\67\12\0\1\4\67\14\0\12\67\15\0\1\5\67"+	
237	"\113\0\1\7\67\11\0\12\67\11\0\11\67\17\0\1\5\67"+	
238	"\12\0\1\1\67\11\7\12\67\14\0\12\67\15\0\1\5\67"+	
239	"\113\0\1\7\67\11\0\12\67\11\0\11\67\17\0\1\5\67"+	
240	"\12\0\1\1\67\11\7\12\67\14\0\12\67\15\0\1\5\67"+	
241	"\113\0\1\7\67\11\0\12\67\11\0\11\67\17\0\1\3\67"+	
242	"\11\173\1\1\67\12\0\1\4\67\14\0\12\67\15\0\1\5\67"+	
243	"\113\0\1\7\67\11\0\12\67\11\0\11\67\17\0\1\4\67"+	
244	"\11\174\12\0\1\4\67\14\0\12\67\15\0\1\4\67\11\75"+	
245	"\113\0\1\7\67\11\0\12\67\11\0\1\1\67\11\76\7\67"+	
246	"\17\0\1\5\67\12\0\1\4\67\14\0\12\67\15\0\1\5\67"+	
247	"\113\0\1\7\67\15\7\0\1\1\77\14\1\0\1\1200\1\0\12\67"+	
248	"\11\0\1\4\67\11\1201\1\4\67\17\0\1\5\67\12\0\1\4\67"+	
249	"\14\0\12\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67"+	
250	"\11\0\1\4\67\11\202\1\4\67\17\0\1\5\67\12\0\1\4\67"+	
251	"\14\0\12\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67"+	
252	"\11\0\12\67\11\203\1\6\67\17\0\1\5\67\12\0\1\4\67"+	
253	"\14\0\12\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67"+	
254	"\11\0\1\1\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67"+	
255	"\15\0\1\4\67\11\204\113\0\1\7\67\11\0\12\67\11\0"+	
256	"\12\1\67\11\205\1\6\67\17\0\1\5\67\12\0\1\4\67\14\0"+	
257	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
258	"\11\1\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67\15\0"+	
259	"\15\67\113\0\1\1\67\11\206\1\5\67\11\0\12\67\11\0"+	
260	"\14\67\11\207\1\4\67\17\0\1\5\67\12\0\1\4\67\14\0"+	
261	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
262	"\11\1\67\17\0\1\1\67\11\210\1\3\67\12\0\1\4\67\14\0"+	
263	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
264	"\14\67\11\72\1\4\67\17\0\1\5\67\12\0\1\4\67\14\0"+	
265	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
266	"\11\1\67\17\0\1\5\67\12\0\1\1\67\11\211\1\2\67\14\0"+	
267	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
268	"\11\1\67\17\0\1\2\67\11\212\1\2\67\12\0\1\4\67\14\0"+	
269	"\12\1\67\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0"+	
270	"\11\213\1\0\1\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67"+	
271	"\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0\1\5\67"+	
272	"\11\147\1\3\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67"+	
273	"\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0\1\4\67"+	
274	"\11\214\1\4\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67"+	
275	"\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0\1\1\67"+	
276	"\17\0\1\3\67\11\215\1\1\67\12\0\1\4\67\14\0\12\67"+	
277	"\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0\1\5\67"+	
278	"\11\216\1\3\67\17\0\1\5\67\12\0\1\4\67\14\0\12\67"+	
279	"\15\0\1\5\67\113\0\1\7\67\11\0\12\67\11\0\1\1\67"+	
280	"\17\0\1\5\67\12\0\1\1\67\11\217\1\2\67\14\0\12\67"+	
281	"\15\0\1\5\67\113\0\1\7\67\12\0\1\1\64\1\4\5\0\1\220"+	
282	"\13\1\0\12\67\11\0\12\67\11\221\1\6\67\17\0\1\5\67"+	
283	"\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
284	"\11\0\12\67\11\0\1\6\67\11\222\1\2\67\17\0\1\5\67"+	
285	"\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
286	"\11\0\12\67\11\0\1\6\67\11\223\1\2\67\17\0\1\5\67"+	
287	"\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
288	"\11\0\12\67\11\0\1\6\67\11\224\1\2\67\17\0\1\5\67"+	
289	"\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
290	"\162\1\0\1\225\1\17\0\12\67\11\0\1\1\67\11\226\1\7\67"+	
291	"\17\0\1\5\67\12\0\1\4\67\14\0\12\67\15\0\1\5\67"+	
292	"\113\0\1\7\67\11\0\12\67\11\0\1\1\67\17\0\1\4\67"+	
293	"\11\227\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
294	"\17\1\67\11\0\12\67\11\0\1\4\67\11\230\1\4\67\17\0\1\7\67"+	
295	"\15\67\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	
296	"\17\1\67\11\0\12\67\11\0\1\1\231\1\0\1\67\17\0\1\5\67"+	
297	"\12\0\1\4\67\14\0\12\67\15\0\1\5\67\113\0\1\7\67"+	



298	"1110121671101111671710156712101467"+	
299	"11012167151014671112321131017167110"+	
300	"12167110111167171015671210146711233"+	
301	"12167141012167151014671112341131017167"+	
302	"110121671101111671710156712101467"+	
303	"11012167151014671112351131017167110"+	
304	"121671101112361101671710156712101467"+	
305	"1101216715101567113101716711012167"+	
306	"1101116711237171671710156712101467"+	
307	"1101216715101567113101716711012167"+	
308	"110111671710156712101467141012167"+	
309	"1510146711240113101716711012167110"+	
310	"1116717101567121014671410121671510"+	
311	"116711241113101716711012167110111167"+	
312	"1710146711242121014671410121671510"+	
313	"15671131017167110121671101111671710"+	
314	"15671210116711243121671410121671510"+	
315	"1567113101716711012167110146711244"+	
316	"146717101567121014671410121671510"+	
317	"15671131017167110121671101216711245"+	
318	"166717101567121014671410121671510"+	
319	"1567113101716716310112461161012167110"+	
320	"1216711247166717101567121014671410"+	
321	"1216715101567113101716711012167110"+	
322	"1467111250146717101567121014671410"+	
323	"1216715101567113101716711012167110"+	
324	"116711251176717101567121014671410"+	
325	"1216715101567113101716711012167110"+	
326	"11167171013167112521167121014671410"+	
327	"1216715101567113101716711012167110"+	
328	"11167171015671210146711253121671410"+	
329	"1216715101567113101716711012167110"+	
330	"1116717101567121014671410121671510"+	
331	"146711254113101716711012167110131255"+	
332	"11016717101567121014671410121671510"+	
333	"15671131017167110121671101111671710"+	
334	"156712101167111256121671410121671510"+	
335	"15671131017167110121671101111671710"+	
336	"131671125711167121014671410121671510"+	
337	"15671131017167110121671101111671710"+	
338	"1567121014671410121671510146711260"+	
339	"11310171671101216711012167112611667"+	
340	"171015671210146714101216715101567"+	
341	"1131017167110121671101467112621467"+	
342	"171015671210146714101216715101567"+	
343	"11310171671101216711011116717101567"+	
344	"1210146714101216715101567113101467"+	
345	"11263121671101216711011116717101467"+	
346	"11264121014671410121671510156711310"+	
347	"176711012167110121671126516671710"+	
348	"1567121014671410121671510156711310"+	
349	"176711012167110111167171015671210"+	
350	"14671410121671510156711310112661667"+	
351	"11012167110111167112671716717101567"+	
352	"12101467141012167151015671131017167"+	
353	"11012167110111167112701716717101567"+	
354	"12101467141012167151015671131017167"+	
355	"11012167110111167171013167112711167"+	
356	"12101467141012167151015671131017167"+	
357	"110121671101467112721467171015671210"+	
358	"12101467141012167151015671131017167"+	
359	"110121671101111671710156712101467"+	
360	"14101216715101467112731131017167110"+	
361	"1216711015671127413167171015671210"+	
362	"1467141012167151015671131017167110"+	
363	"1216711011116717101216711275121671210"+	



```
364 "4674026750567130767110"+
365 "267101167703671276167200"+
366 "4674026750567130767110"+
367 "26710127710677056720467"+
368 "4026750567130767110267"+
369 "102671300677056720467"+
370 "4026750567130767110267"+
371 "106713012677056720467"+
372 "4026750567130767110267"+
373 "101167705672046740267"+
374 "504671302130767110267110"+
375 "46713034677056720467400"+
376 "26750567130767110267110"+
377 "1167701167130436720467400"+
378 "26750567130767110267110"+
379 "1167705672046740267500"+
380 "46713051307671102671101167"+
381 "70567204674026750467"+
382 "13061307671102671104671307"+
383 "467705672046740267500"+
384 "5671307671102671101167700"+
385 "5672046740267505671300"+
386 "13106711026711013111067700"+
387 "5672046740267505671300"+
388 "7671102671101312106770567"+
389 "204674026750567130767"+
390 "1026711011677056720467"+
391 "402675046711313130767110"+
392 "2671101167131476770567200"+
393 "4674026750567130767110"+
394 "26711011677056720467400"+
395 "2675056713026711315467110"+
396 "267110116770367113161167200"+
```

```
397 "4674026750567130767110"+
398 "267110116770467131720467"+
399 "4026750567130767110267"+
400 "10116770567204671320267"+
401 "4026750567130767";
402
403 private static int [] zzUnpackTrans() {
404     int [] result = new int[9490];
405     int offset = 0;
406     offset = zzUnpackTrans(zz_TRANS_PACKED_0, offset, result);
407     return result;
408 }
409
410 private static int zzUnpackTrans(String packed, int offset, int [] result) {
411     int i = 0; /* index in packed string */
412     int j = offset; /* index in unpacked array */
413     int l = packed.length();
414     while (i < l) {
415         int count = packed.charAt(i++);
416         int value = packed.charAt(i++);
417         value--;
418         do result[j++] = value; while (--count > 0);
419     }
420     return j;
421 }
422
423
424 /* error codes */
425 private static final int zz_UNKNOWN_ERROR = 0;
426 private static final int zz_NO_MATCH = 1;
427 private static final int zz_PUSHBACK_2BIG = 2;
428
429 /* error messages for the codes above */
```





```

430 private static final String ZZ_ERROR_MSG[] = {
431     "Unkown internal scanner error",
432     "Error: could not match input",
433     "Error: pushback value was too large"
434 };
435
436 /**
437  * ZZ_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
438  */
439 private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();
440
441 private static final String ZZ_ATTRIBUTE_PACKED_0 =
442     "\1\0\1\1\1\15\1\3\1\1\7\1\3\1\1\3\1\1\1\1"+
443     "\7\1\1\1\1\1\1\1\22\1\2\1\1\3\1\2\1\1"+
444     "\1\0\1\1\3\1\1\0\1\1\1\3\0\1\1\1\0"+
445     "\6\1\1\0\1\1\1\0\2\1\1\1\1\2\1\1\1\1"+
446     "\1\0\3\1\1\7\1\1\1\1\1\0\17\1\1\1\1\4\1"+
447     "\1\0\20\1\1\1\52\1\1";
448
449 private static int [] zzUnpackAttribute() {
450     int [] result = new int[208];
451     int offset = 0;
452     offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
453     return result;
454 }
455
456 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
457     int i = 0; /* index in packed string */
458     int j = offset; /* index in unpacked array */
459     int l = packed.length();
460     while (i < l) {
461         int count = packed.charAt(i++);
462         int value = packed.charAt(i++);

```

```

463         do result[j++] = value; while (--count > 0);
464     }
465     return j;
466 }
467
468 /** the input device */
469 private java.io.Reader zzReader;
470
471 /** the current state of the DFA */
472 private int zzState;
473
474 /** the current lexical state */
475 private int zzLexicalState = YYINITIAL;
476
477 /** this buffer contains the current text to be matched and is
478  * the source of the yytext() string */
479 private char zzBuffer[] = new char[ZZ_BUFFER_SIZE];
480
481 /** the textposition at the last accepting state */
482 private int zzMarkedPos;
483
484 /** the current text position in the buffer */
485 private int zzCurrentPos;
486
487 /** startRead marks the beginning of the yytext() string in the buffer */
488 private int zzStartRead;
489
490 /** endRead marks the last character in the buffer, that has been read
491  * from input */
492 private int zzEndRead;
493
494 /** number of newlines encountered up to the start of the matched text */
495 private int yyline;

```



```

496
497 /** the number of characters up to the start of the matched text */
498 private int vychar;
499
500 /**
501  * the number of characters from the last newline up to the start of the
502  * matched text
503  */
504 private int vycolumn;
505
506 /**
507  * zzAtBOL == true <=> the scanner is currently at the beginning of a line
508  */
509 private boolean zzAtBOL = true;
510
511 /** zzAtEOF == true <=> the scanner is at the EOF */
512 private boolean zzAtEOF;
513
514 /** denotes if the user-EOF-code has already been executed */
515 private boolean zzEOFDone;
516
517 /* user code: */
518 public String lexeme;
519
520
521 /**
522  * Creates a new scanner
523  * There is also a java.io.InputStream version of this constructor.
524  *
525  * @param in the java.io.Reader to read input from.
526  */
527 Lexer(java.io.Reader in) {
528     this.zzReader = in;

```

```

529 }
530
531 /**
532  * Creates a new scanner.
533  * There is also java.io.Reader version of this constructor.
534  *
535  * @param in the java.io.InputStream to read input from.
536  */
537 Lexer(java.io.InputStream in) {
538     this(new java.io.InputStreamReader(in));
539 }
540
541 /**
542  * Unpacks the compressed character translation table.
543  *
544  * @param packed the packed character translation table
545  * @return the unpacked character translation table
546  */
547 private static char [] zzUnpackCMap(String packed) {
548     char [] map = new char[0x10000];
549     int i = 0; /* index in packed string */
550     int j = 0; /* index in unpacked array */
551     while (i < 166) {
552         int count = packed.charAt(i++);
553         char value = packed.charAt(i++);
554         do map[j++] = value; while (--count > 0);
555     }
556     return map;
557 }
558
559
560 /**
561  * Refills the input buffer.

```





```

562 *
563 * @return <code>false</code>, iff there was new input.
564 *
565 * @exception java.io.IOException if any I/O-Error occurs
566 */
567 private boolean zzRefill() throws java.io.IOException {
568
569     /* first: make room (if you can) */
570     if (zzStartRead > 0) {
571         System.arraycopy(zzBuffer, zzStartRead,
572             zzBuffer, 0,
573             zzEndRead-zzStartRead);
574
575         /* translate stored positions */
576         zzEndRead -= zzStartRead;
577         zzCurrentPos -= zzStartRead;
578         zzMarkedPos -= zzStartRead;
579         zzStartRead = 0;
580     }
581
582     /* is the buffer big enough? */
583     if (zzCurrentPos >= zzBuffer.length) {
584         /* if not: blow it up */
585         char newBuffer[] = new char[zzCurrentPos*2];
586         System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
587         zzBuffer = newBuffer;
588     }
589
590     /* finally: fill the buffer with new input */
591     int numRead = zzReader.read(zzBuffer, zzEndRead,
592         zzBuffer.length-zzEndRead);
593
594     if (numRead > 0) {
595         zzEndRead += numRead;
596         return false;
597     }
598     // unlikely but not impossible: read 0 characters, but not at end of stream
599     if (numRead == 0) {
600         int c = zzReader.read();
601         if (c == -1) {
602             return true;
603         } else {
604             zzBuffer[zzEndRead++] = (char) c;
605             return false;
606         }
607     }
608
609     // numRead < 0
610     return true;
611 }
612
613 /**
614  * Closes the input stream.
615  */
616 public final void yyclose() throws java.io.IOException {
617     zzAtEOF = true; /* indicate end of file */
618     zzEndRead = zzStartRead; /* invalidate buffer */
619
620     if (zzReader != null)
621         zzReader.close();
622 }
623
624 /**
625  * Resets the scanner to read from a new input stream.
626  */

```



```

628      * Does not close the old reader.
629      *
630      * All internal variables are reset, the old input stream
631      * <b>cannot</b> be reused (internal buffer is discarded and lost).
632      * Lexical state is set to <tt>ZZ_INITIAL</tt>.
633      *
634      * @param reader the new input stream
635      */
636      public final void yyreset(java.io.Reader reader) {
637          zzReader = reader;
638          zzAtBOL = true;
639          zzAtEOF = false;
640          zzEOFDone = false;
641          zzEndRead = zzStartRead = 0;
642          zzCurrentPos = zzMarkedPos = 0;
643          yyline = yychar = yycolumn = 0;
644          zzLexicalState = YYINITIAL;
645      }
646
647
648      /**
649       * Returns the current lexical state.
650       */
651      public final int yystate() {
652          return zzLexicalState;
653      }
654
655
656      /**
657       * Enters a new lexical state
658       *
659       * @param newState the new lexical state
660       */

```

```

661      public final void yybegin(int newState) {
662          zzLexicalState = newState;
663      }
664
665
666      /**
667       * Returns the text matched by the current regular expression.
668       */
669      public final String yytext() {
670          return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
671      }
672
673
674      /**
675       * Returns the character at position <tt>pos</tt> from the
676       * matched text.
677       *
678       * It is equivalent to yytext().charAt(pos), but faster
679       *
680       * @param pos the position of the character to fetch.
681       *           A value from 0 to yylength()-1.
682       *
683       * @return the character at position pos
684       */
685      public final char yycharat(int pos) {
686          return zzBuffer[zzStartRead+pos];
687      }
688
689
690      /**
691       * Returns the length of the matched text region.
692       */
693      public final int yylength() {

```



```

694         return zzMarkedPos-zzStartRead;
695     }
696
697
698     /**
699     * Reports an error that occurred while scanning.
700     *
701     * In a wellformed scanner (no or only correct usage of
702     * yypushback(int) and a match-all fallback rule) this method
703     * will only be called with things that "Can't Possibly Happen".
704     * If this method is called, something is seriously wrong
705     * (e.g. a JFlex bug producing a faulty scanner etc.).
706     *
707     * Usual syntax/scanner level error handling should be done
708     * in error fallback rules.
709     *
710     * @param   errorCode   the code of the error message to display
711     */
712     private void zzScanError(int errorCode) {
713         String message;
714         try {
715             message = ZZ_ERROR_MSG[errorCode];
716         }
717         catch (ArrayIndexOutOfBoundsException e) {
718             message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
719         }
720
721         throw new Error(message);
722     }
723
724
725     /**
726     * Pushes the specified amount of characters back into the input stream.

```

```

727     *
728     * They will be read again by then next call of the scanning method
729     *
730     * @param number   the number of characters to be read again.
731     *                 This number must not be greater than yylength()!
732     */
733     public void yypushback(int number) {
734         if ( number > yylength() )
735             zzScanError(ZZ_PUSHBACK_2BIG);
736
737         zzMarkedPos -= number;
738     }
739
740
741     /**
742     * Resumes scanning until the next regular expression is matched,
743     * the end of input is encountered or an I/O-Error occurs.
744     *
745     * @return        the next token
746     * @exception     java.io.IOException if any I/O-Error occurs
747     */
748     public Tokens yylex() throws java.io.IOException {
749         int zzInput;
750         int zzAction;
751
752         // cached fields:
753         int zzCurrentPosL;
754         int zzMarkedPosL;
755         int zzEndReadL = zzEndRead;
756         char [] zzBufferL = zzBuffer;
757         char [] zzCMapL = ZZ_CMAP;
758
759         int [] zzTransL = ZZ_TRANS;

```



```

760 int [] zzRowMapL = ZZ_ROWMAP;
761 int [] zzAttrL = ZZ_ATTRIBUTE;
762
763 while (true) {
764     zzMarkedPosL = zzMarkedPos;
765
766     zzAction = -1;
767
768     zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
769
770     zzState = ZZ_LEXSTATE[zzLexicalState];
771
772
773     zzForAction: {
774         while (true) {
775
776             if (zzCurrentPosL < zzEndReadL)
777                 zzInput = zzBufferL[zzCurrentPosL++];
778             else if (zzAtEOF) {
779                 zzInput = YYEOF;
780                 break zzForAction;
781             }
782             else {
783                 // store back cached positions
784                 zzCurrentPos = zzCurrentPosL;
785                 zzMarkedPos = zzMarkedPosL;
786                 boolean eof = zzRefill();
787                 // get translated positions and possibly new buffer
788                 zzCurrentPosL = zzCurrentPos;
789                 zzMarkedPosL = zzMarkedPos;
790                 zzBufferL = zzBuffer;
791                 zzEndReadL = zzEndRead;
792                 if (eof) {
793                     zzInput = YYEOF;
794                     break zzForAction;
795                 }
796                 else {
797                     zzInput = zzBufferL[zzCurrentPosL++];
798                 }
799             }
800             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
801             if (zzNext == -1) break zzForAction;
802             zzState = zzNext;
803
804             int zzAttributes = zzAttrL[zzState];
805             if ( ( zzAttributes & 1 ) == 1 ) {
806                 zzAction = zzState;
807                 zzMarkedPosL = zzCurrentPosL;
808                 if ( ( zzAttributes & 8 ) == 8 ) break zzForAction;
809             }
810
811         }
812     }
813
814     // store back cached position
815     zzMarkedPos = zzMarkedPosL;
816
817     switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
818     case 20:
819         { return parentDer;
820         }
821     case 77: break;
822     case 21:
823         { return llaveDer;
824         }
825     case 78: break;

```



```
826 case 10:
827 { return operadorModulo;
828 }
829 case 79: break;
830 case 32:
831 { lexeme=yytext(); return Identificador;
832 }
833 case 80: break;
834 case 22:
835 { return pesos;
836 }
837 case 81: break;
838 case 54:
839 { return cadena;
840 }
841 case 82: break;
842 case 69:
843 { return detener;
844 }
845 case 83: break;
846 case 31:
847 { return gato;
848 }
849 case 84: break;
850 case 3:
851 { return numero;
852 }
853 case 85: break;
854 case 2:
855 { return guionBajo;
856 }
857 case 86: break;
858 case 67:
```

```
859 { return arreglo;
860 }
861 case 87: break;
862 case 11:
863 { return operadorPotencia;
864 }
865 case 88: break;
866 case 72:
867 { return importar;
868 }
869 case 89: break;
870 case 34:
871 { lexeme=yytext(); return Reservadas;
872 }
873 case 90: break;
874 case 75:
875 { return funcionIntroducirDatos;
876 }
877 case 91: break;
878 case 36:
879 { return incremento;
880 }
881 case 92: break;
882 case 24:
883 { return corchetIzq;
884 }
885 case 93: break;
886 case 17:
887 { return operadorNo;
888 }
889 case 94: break;
890 case 4:
891 { /*Ignore*/
```



```
892 |         }  
893 |         case 95: break;  
894 |         case 8:  
895 |         { return operadorResta;  
896 |         }  
897 |         case 96: break;  
898 |         case 60:  
899 |         { return funcionLeer;  
900 |         }  
901 |         case 97: break;  
902 |         case 46:  
903 |         { return operadorPositivo;  
904 |         }  
905 |         case 98: break;  
906 |         case 14:  
907 |         { return menorQue;  
908 |         }  
909 |         case 99: break;  
910 |         case 5:  
911 |         { return letrasMin;  
912 |         }  
913 |         case 100: break;  
914 |         case 70:  
915 |         { return cicloWhile;  
916 |         }  
917 |         case 101: break;  
918 |         case 53:  
919 |         { return valorBooleano;  
920 |         }  
921 |         case 102: break;  
922 |         case 65:  
923 |         { return cicloFor;  
924 |         }
```

```
925 |         case 103: break;  
926 |         case 42:  
927 |         { return inicioTexto;  
928 |         }  
929 |         case 104: break;  
930 |         case 66:  
931 |         { return clase;  
932 |         }  
933 |         case 105: break;  
934 |         case 1:  
935 |         { return ERROR;  
936 |         }  
937 |         case 106: break;  
938 |         case 19:  
939 |         { return dosPuntos;  
940 |         }  
941 |         case 107: break;  
942 |         case 28:  
943 |         { return comillaSimple;  
944 |         }  
945 |         case 108: break;  
946 |         case 76:  
947 |         { return predeterminado;  
948 |         }  
949 |         case 109: break;  
950 |         case 50:  
951 |         { return concatenar;  
952 |         }  
953 |         case 110: break;  
954 |         case 56:  
955 |         { return condicionalElse;  
956 |         }  
957 |         case 111: break;
```



```
958 | case 39:
959 |     { return asignacion;
960 |     }
961 | case 112: break;
962 | case 44:
963 |     { return saltoLineas;
964 |     }
965 | case 113: break;
966 | case 63:
967 |     { return cicloDo;
968 |     }
969 | case 114: break;
970 | case 49:
971 |     { return inicioComentarioMult;
972 |     }
973 | case 115: break;
974 | case 37:
975 |     { return decremento;
976 |     }
977 | case 116: break;
978 | case 13:
979 |     { return mayorQue;
980 |     }
981 | case 117: break;
982 | case 64:
983 |     { return operadorRaiz;
984 |     }
985 | case 118: break;
986 | case 35:
987 |     { return condicionalIf;
988 |     }
989 | case 119: break;
990 | case 57:
```

```
991 |     { return funcionCase;
992 |     }
993 | case 120: break;
994 | case 16:
995 |     { return operadorO;
996 |     }
997 | case 121: break;
998 | case 43:
999 |     { return finalComentarioMult;
1000 |     }
1001 | case 122: break;
1002 | case 74:
1003 |     { return constante;
1004 |     }
1005 | case 123: break;
1006 | case 25:
1007 |     { return corchetDer;
1008 |     }
1009 | case 124: break;
1010 | case 58:
1011 |     { return valorFlotante;
1012 |     }
1013 | case 125: break;
1014 | case 51:
1015 |     { return comentarioLineas;
1016 |     }
1017 | case 126: break;
1018 | case 47:
1019 |     { return operadorNegativo;
1020 |     }
1021 | case 127: break;
1022 | case 26:
1023 |     { return puntoAcceso;
```



```
1024 | }  
1025 | case 128: break;  
1026 | case 18:  
1027 | { return parentIzq;  
1028 | }  
1029 | case 129: break;  
1030 | case 23:  
1031 | { return llaveIzq;  
1032 | }  
1033 | case 130: break;  
1034 | case 40:  
1035 | { return finalTexto;  
1036 | }  
1037 | case 131: break;  
1038 | case 33:  
1039 | { lexeme=yytext(); return Numero;  
1040 | }  
1041 | case 132: break;  
1042 | case 48:  
1043 | { return menorIgual;  
1044 | }  
1045 | case 133: break;  
1046 | case 27:  
1047 | { return comillaDoble;  
1048 | }  
1049 | case 134: break;  
1050 | case 7:  
1051 | { return operadorSuma;  
1052 | }  
1053 | case 135: break;  
1054 | case 12:  
1055 | { return letrasMay;  
1056 | }
```

```
1057 | case 136: break;  
1058 | case 45:  
1059 | { return mayorIgual;  
1060 | }  
1061 | case 137: break;  
1062 | case 62:  
1063 | { return funcionSalirCase;  
1064 | }  
1065 | case 138: break;  
1066 | case 68:  
1067 | { return funcion;  
1068 | }  
1069 | case 139: break;  
1070 | case 29:  
1071 | { return diagonalInvertido;  
1072 | }  
1073 | case 140: break;  
1074 | case 30:  
1075 | { return signoAdmiracionAbre;  
1076 | }  
1077 | case 141: break;  
1078 | case 71:  
1079 | { return variable;  
1080 | }  
1081 | case 142: break;  
1082 | case 73:  
1083 | { return imprimir;  
1084 | }  
1085 | case 143: break;  
1086 | case 15:  
1087 | { return operadorY;  
1088 | }  
1089 | case 144: break;
```





```
1090 case 59:
1091 { return carácter;
1092 }
1093 case 145: break;
1094 case 38:
1095 { return comparacionIgualdad;
1096 }
1097 case 146: break;
1098 case 41:
1099 { return pi;
1100 }
1101 case 147: break;
1102 case 6:
1103 { return operadorDivision;
1104 }
1105 case 148: break;
1106 case 9:
1107 { return operadorMultiplicacion;
1108 }
1109 case 149: break;
1110 case 61:
1111 { return funcionSwitch;
1112 }
1113 case 150: break;
1114 case 52:
1115 { return valorEntero;
1116 }
1117 case 151: break;
1118 case 55:
1119 { return finLinea;
1120 }
1121 case 152: break;
1122 default:
```

```
1123 if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
1124     zzAtEOF = true;
1125     return null;
1126 }
1127 else {
1128     zzScanError(ZZ_NO_MATCH);
1129 }
1130 }
1131 }
1132 }
1133 }
1134 }
1135 }
1136 }
```

## Archivo Tokens.java

El archivo Tokens.java es una pieza fundamental en la implementación de este analizador léxico, ya que define una enumeración que clasifica los diferentes tipos de tokens que el analizador puede identificar en el código fuente. Los tokens representan las unidades léxicas básicas del lenguaje, como palabras reservadas, operadores, identificadores, números y símbolos especiales, así como elementos estructurales como llaves, paréntesis y comentarios. Además, incluye un token genérico para errores, lo que permite gestionar y reportar entradas no reconocidas.

Este archivo actúa como una referencia central para el análisis y clasificación de los elementos léxicos, estableciendo una relación clara entre el código fuente y los componentes del lenguaje que se está interpretando o compilando. Esta enumeración facilita la implementación de reglas de análisis, mejora la legibilidad del código y contribuye a mantener una estructura ordenada y extensible para futuras actualizaciones o ampliaciones del lenguaje soportado por el analizador.



## Código Realizado:

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package codigo;
7
8
9  public enum Tokens {
10     Linea,
11     Reservadas,
12     operadorSuma,
13     operadorResta,
14     operadorMultiplicacion,
15     operadorDivision,
16     operadorModulo,
17     operadorPotencia,
18     operadorRaiz,
19     asignacion,
20     incremento,
21     decremento,
22     operadorPositivo,
23     operadorNegativo,
24     comparacionIgualdad,
25     mayorQue,
26     menorQue,
27     mayorIgual,
28     menorIgual,
29     operadorY,
30     operadorO,
31     operadorNo,
32     letrasMin,
33     letrasMay,
```

```
34     numero,
35     inicioComentarioMult,
36     finalComentarioMult,
37     comentarioLinea,
38     valorEntero,
39     valorFlotante,
40     valorBooleano,
41     cadena,
42     carácter,
43     finLinea,
44
45     inicioTexto,
46     finalTexto,
47     parentIzq,
48     parentDer,
49     llaveIzq,
50     llaveDer,
51     corchetIzq,
52     corchetDer,
53     concatenar,
54     arreglo,
55
56     puntoDecimal,
57     dosPuntos,
58     espacio,
59     tabulacion,
60     comillaDoble,
61     comillaSimple,
62     coma,
63     guionBajo,
64     guionMedio,
65     diagonalInvertido,
66     diagonal ,
```



```
67 signoAdmiracionAbre,  
68 signoAdmiracionCierra,  
69 gato,  
70 pesos,  
71 condicionalIf,  
72 condicionalElse,  
73 cicloFor,  
74 cicloWhile,  
75 cicloDo,  
76 funcionSwitch,  
77 funcionCase,  
78 funcionSalirCase,  
79 predeterminado,  
80 detener,  
81 constante,  
82 variable,  
83 funcion,  
84 clase,  
85 imprimir,  
86 importar,  
87 funcionLeer,  
88 funcionIntroducirDatos,  
89 pi,  
90 euler,  
91 Numero,  
92 Identificador,  
93 ERROR  
94 )
```

## Archivo Principal.java

El archivo Principal.java actúa como el punto de entrada para la ejecución del analizador léxico en Java. Este archivo combina la inicialización de la interfaz gráfica con la ejecución del proceso que genera el analizador léxico a partir de una especificación escrita en JFlex. La clase implementa un splash screen mediante un hilo separado, que sirve como una introducción visual antes de cargar la interfaz principal del sistema, representada por el componente LoginUI.

Este archivo es un coordinador del flujo de ejecución del programa. Pues gestiona la experiencia del usuario al proporcionar un inicio visualmente atractivo y funcional. Además también automatiza la generación del archivo del analizador léxico mediante la herramienta JFlex, asegurando que el componente encargado del análisis de tokens esté disponible y actualizado. Esto lo convierte en una pieza esencial para la interacción fluida entre el usuario y el sistema, además de garantizar la eficiencia técnica del proceso de análisis léxico.



## Código Realizado:

```
1 package codigo;
2
3
4 import java.io.File;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7
8 public class Principal {
9
10     public static void main(String[] args) {
11
12         // Implementación del Runnable para el Splash y la pantalla principal
13         Runnable mRun = () -> {
14             // JFrame que funge como splash
15             ModernSplash mSplash = new ModernSplash();
16             mSplash.setVisible(true);
17
18             try {
19                 Thread.sleep(10000); // 5000 milisegundos equivale a 5 segundos.
20             } catch (InterruptedException ex) {
21                 Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
22             }
23
24             mSplash.dispose();
25
26             // JFrame que funge como pantalla principal
27             LoginUI mFrmPrincipal = new LoginUI(); // Renombrado de FrmBienvenida
28             mFrmPrincipal.setVisible(true);
29         };
30
31         // Creación e inicio del hilo para el Splash
32         Thread mHiloSplash = new Thread(mRun);
33         mHiloSplash.start();
34
35         // Ejecución del método para generar el Lexer
36         String ruta = "C:/Users/jesus/Downloads/AnalizadorLexico/src/codigo/Lexer.flex";
37         generarLexer(ruta);
38     }
39
40     public static void generarLexer(String ruta) {
41         File archivo = new File(ruta);
42         JFlex.Main.generate(archivo);
43     }
44 }
```

### Archivo ModernSplash.java (Splash del proyecto)

El splash de este proyecto, llamado ModernSplash, es una pantalla inicial diseñada para ofrecer una experiencia visual atractiva mientras se cargan los recursos necesarios del sistema prácticamente es una ventana inicial moderna e interactiva que combina varias animaciones y efectos visuales para ofrecer una experiencia de bienvenida atractiva al usuario.. Este componente destaca por su diseño moderno, que incluye un fondo degradado en tonos azules, un título animado que realiza un efecto de caída y rebote simulado con física, y un icono que flota suavemente gracias a una animación sinusoidal. Además, incorpora una barra de progreso estilizada con colores dinámicos y mensajes de estado que informan al usuario sobre el avance de la carga, como "Preparando recursos..." y "Cargando módulos...".

La finalidad principal de este splash es mejorar la interacción inicial con el usuario, proporcionando un entorno visual que no solo entretiene, sino que también comunica claramente el progreso mientras se inicializan los módulos del proyecto. Este tipo de elemento no solo ayuda a gestionar la percepción del tiempo de carga, sino que también refuerza la identidad visual del proyecto, haciendo que la primera impresión sea profesional y agradable.



## Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class ModernSplash extends JFrame {
8     private JProgressBar progressBar;
9     private JLabel titleLabel, loadingLabel, iconLabel;
10    private Timer progressTimer, fadeTimer, bounceTimer, colorTimer, floatTimer;
11    private float opacity = 0.0f;
12    private Color[] gradientColors = {new Color(76, 175, 80)};
13    private int gradientIndex = 0;
14
15    // Variables para animación del título
16    private int titleYPosition = 50; // Posición inicial
17    private int targetY = 170; // Posición final
18    private double velocity = 0; // Velocidad inicial
19    private double gravity = 1.2; // Simula la gravedad
20    private int bouncesRemaining = 3; // Número de rebotes
21    private double dampingFactor = 0.6; // Factor de amortiguación
22
23    // Variables para animación flotante del icono
24    private int iconBaseY = 30; // Posición base del icono
25    private int floatAmplitude = 10; // Amplitud de la flotación
26    private double floatAngle = 0; // Ángulo para la animación sinusoidal
27
28    public ModernSplash() {
29        initComponents();
30    }
31
32    private void initComponents() {
33        // Configuración principal del JFrame
34        setUndecorated(true);
35        setSize(400, 300);
36        setOpacity(0.0f);
37        setLocationRelativeTo(null);
38        setLayout(null);
39
40        // Fondo personalizado
41        JPanel backgroundPanel = new JPanel() {
42            @Override
43            protected void paintComponent(Graphics g) {
44                super.paintComponent(g);
45                Graphics2D g2d = (Graphics2D) g;
46                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
47
48                // Degradado del fondo
49                GradientPaint gradient = new GradientPaint(0, 0, new Color(30, 136, 229), getWidth(), getHeight(), new Color(63, 81, 181));
50                g2d.setPaint(gradient);
51                g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
52            }
53        };
54        backgroundPanel.setBounds(0, 0, 400, 300);
55        backgroundPanel.setLayout(null);
56        add(backgroundPanel);
57
58        // Icono
59        iconLabel = new JLabel(new ImageIcon("C:\\Users\\jesus\\Downloads\\AnalizadorLexico\\src\\codigo\\icono.png"));
60        iconLabel.setBounds(125, iconBaseY, 150, 150);
61        iconLabel.setHorizontalAlignment(SwingConstants.CENTER);
62        backgroundPanel.add(iconLabel);
63    }
```



```

64 // Título con animación
65 titleLabel = new JLabel("Token-Visor");
66 titleLabel.setBounds(50, titleYPosition, 300, 40);
67 titleLabel.setFont(new Font("Roboto", Font.BOLD, 24));
68 titleLabel.setForeground(Color.WHITE);
69 titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
70 backgroundPanel.add(titleLabel);
71
72 // Barra de progreso
73 progressBar = new JProgressBar() {
74     @Override
75     protected void paintComponent(Graphics g) {
76         super.paintComponent(g);
77         Graphics2D g2d = (Graphics2D) g;
78         g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
79
80         int progressWidth = (int) (getWidth() * (getValue() / 100.0));
81         g2d.setColor(gradientColors[gradientIndex]); // Color dinámico
82         g2d.fillRoundRect(0, 0, progressWidth, getHeight(), 10, 10);
83     }
84 };
85 progressBar.setBounds(50, 230, 300, 10);
86 progressBar.setValue(0);
87 backgroundPanel.add(progressBar);
88
89 // Texto de carga
90 loadingLabel = new JLabel("Iniciando...");
91 loadingLabel.setBounds(50, 250, 300, 20);
92 loadingLabel.setFont(new Font("Roboto", Font.PLAIN, 14));
93 loadingLabel.setForeground(Color.WHITE);
94 loadingLabel.setHorizontalAlignment(SwingConstants.CENTER);
95 backgroundPanel.add(loadingLabel);
96

```

```

97 // Iniciar animaciones
98 startAnimations();
99
100
101 private void startAnimations() {
102     // Fade-in de la ventana
103     fadeTimer = new Timer(30, (ActionEvent e) -> {
104         opacity += 0.02f;
105         if (opacity >= 1.0f) {
106             setOpacity(1.0f);
107             fadeTimer.stop();
108             startFloatAnimation(); // Inicia la animación flotante del icono
109             startBounceAnimation(); // Inicia la animación del título
110         } else {
111             setOpacity(opacity);
112         }
113     });
114     fadeTimer.start();
115
116     // Cambio de color dinámico para la barra de progreso
117     colorTimer = new Timer(2000, (ActionEvent e) -> {
118         gradientIndex = (gradientIndex + 1) % gradientColors.length;
119         progressBar.repaint();
120     });
121     colorTimer.start();
122 }
123
124 private void startFloatAnimation() {
125     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez
126         // Animación sinusoidal para "flotar"
127         floatAngle += 0.05; // Incremento más pequeño para ralentizar el movimiento
128         int offsetY = (int) (Math.sin(floatAngle) * floatAmplitude);
129         iconLabel.setBounds(125, iconBaseY + offsetY, 150, 150);
130     });
131     floatTimer.start();
132 }

```



```
130 }
131     floatTimer.start();
132 }
133
134
135 private void startBounceAnimation() {
136     bounceTimer = new Timer(20, (ActionEvent e) -> {
137         velocity += gravity; // Incrementa velocidad por efecto de "gravedad"
138         titleYPosition += velocity; // Actualiza posición del título
139
140         // Detecta colisión con el suelo virtual
141         if (titleYPosition >= targetY) {
142             titleYPosition = targetY; // Ajusta la posición al suelo
143             velocity = -velocity * dampingFactor; // Rebote con amortiguación
144             bouncesRemaining--;
145
146             // Si no quedan más rebotes, detiene la animación
147             if (bouncesRemaining == 0) {
148                 bounceTimer.stop();
149                 titleYPosition = targetY; // Asegura posición final
150                 startProgressBar(); // Inicia la barra de progreso
151             }
152         }
153
154         // Actualiza la posición del título en el componente
155         titleLabel.setBounds(50, titleYPosition, 300, 40);
156     });
157
158     bounceTimer.start();
159 }
```

```
160
161 private void startProgressBar() {
162     progressTimer = new Timer(50, (ActionEvent e) -> {
163         int value = progressBar.getValue();
164         if (value < 100) {
165             progressBar.setValue(value + 1);
166
167             // Cambiar texto dinámicamente
168             if (value < 30) {
169                 loadingLabel.setText("Preparando recursos...");
170             } else if (value < 60) {
171                 loadingLabel.setText("Cargando módulos...");
172             } else if (value < 90) {
173                 loadingLabel.setText("Finalizando configuración...");
174             } else {
175                 loadingLabel.setText("¡Listo para comenzar!");
176             }
177             } else {
178                 progressTimer.stop();
179                 startFadeOut();
180             }
181         });
182     progressTimer.start();
183 }
184
185 private void startFadeOut() {
186     fadeTimer = new Timer(30, (ActionEvent e) -> {
187         opacity -= 0.02f;
188         if (opacity <= 0.0f) {
189             setOpacity(0.0f);
190             fadeTimer.stop();
191             dispose(); // Cierra la ventana
192         } else {
193             setOpacity(opacity);
194         }
195     });
196     fadeTimer.start();
197 }
198
199 public static void main(String[] args) {
200     EventQueue.invokeLater(() -> {
201         ModernSplash splash = new ModernSplash();
202         splash.setVisible(true);
203     });
204 }
205
206 }
```



## Archivo LoginUI.java

El archivo LoginUI.java es una implementación de una interfaz gráfica de usuario (GUI) en Java diseñada para ofrecer una experiencia interactiva de inicio de sesión dentro del analizador léxico. Utilizando la biblioteca Swing, esta clase presenta una ventana dividida en dos paneles: un lado izquierdo visualmente atractivo con un ícono flotante animado y un mensaje de bienvenida, y un lado derecho funcional que contiene el formulario de inicio de sesión con campos para el usuario y la contraseña.

La importancia de este archivo radica en su combinación de funcionalidad y estética. Además de validar credenciales de acceso con una lógica sencilla, proporciona un diseño profesional que mejora la interacción del usuario mediante detalles como animaciones suaves y botones personalizados. Su estructura modular y enfoque en la usabilidad lo convierten en un componente clave del sistema, creando una primera impresión positiva y facilitando la transición hacia las funcionalidades principales del programa.

### Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.ActionEvent;
6
7 public class LoginUI extends JFrame {
8
9     private JTextField registerNameField;
10    private JPasswordField registerPasswordField;
11    private JPanel leftPanel;
12    private JLabel iconLabel;
13    private Timer floatTimer;
14    private int iconBaseY = 100; // Posición base del icono en el eje Y
15    private int floatAmplitude = 10; // Amplitud de la flotación
16    private double floatAngle = 0; // Ángulo para la animación sinusoidal
17
18    public LoginUI() {
19        initComponents();
20        startFloatAnimation(); // Inicia la animación flotante del icono
21    }
22
23    private void initComponents() {
24        setTitle("Login UI");
25        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26        setSize(800, 500);
27        setLocationRelativeTo(null);
28        setLayout(new GridLayout(1, 2));
29
30        // Panel izquierdo (Bienvenida)
31        leftPanel = new JPanel();
32        leftPanel.setBackground(new Color(0, 38, 95));
33        leftPanel.setLayout(null); // Posición absoluta para incluir el icono
```





```

35 JLabel welcomeLabel = new JLabel("<html><center>Nos Alegra Verte De Nuevo!<br>Inicia sesión para continuar</center></html>");
36 welcomeLabel.setForeground(Color.WHITE);
37 welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));
38 welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
39 welcomeLabel.setBounds(50, 300, 300, 100);
40 leftPanel.add(welcomeLabel);
41
42 // ícono flotante
43 iconLabel = new JLabel(new ImageIcon("C:\\Users\\jesus\\Downloads\\AnalizadorLexico\\src\\codigo\\icono.png"));
44 iconLabel.setBounds(0, iconBaseY, 150, 150); // La posición X se ajustará dinámicamente
45 leftPanel.add(iconLabel);
46
47 add(leftPanel);
48
49 // Panel derecho (Formulario de registro)
50 JPanel rightPanel = new JPanel();
51 rightPanel.setBackground(Color.WHITE);
52 rightPanel.setLayout(new GridBagLayout());
53
54 GridBagConstraints gbc = new GridBagConstraints();
55 gbc.insets = new Insets(10, 10, 10, 10);
56 gbc.gridx = 0;
57 gbc.anchor = GridBagConstraints.CENTER;
58
59 JLabel createAccountLabel = new JLabel("Inicio De Sesión");
60 createAccountLabel.setFont(new Font("Arial", Font.BOLD, 24));
61 createAccountLabel.setForeground(Color.BLACK);
62 gbc.gridy = 0;
63 rightPanel.add(createAccountLabel, gbc);
64
65 registerNameField = new JTextField(20);
66 registerNameField.setFont(new Font("Arial", Font.PLAIN, 14));
67 registerNameField.setBorder(BorderFactory.createTitledBorder("Usuario"));
68
69 gbc.gridy = 1;
70 rightPanel.add(registerNameField, gbc);
71
72 registerPasswordField = new JPasswordField(20);
73 registerPasswordField.setFont(new Font("Arial", Font.PLAIN, 14));
74 registerPasswordField.setBorder(BorderFactory.createTitledBorder("Contraseña"));
75 gbc.gridy = 2;
76 rightPanel.add(registerPasswordField, gbc);
77
78 CustomButton signUpButton = new CustomButton("Ingresar");
79 signUpButton.setPreferredSize(new Dimension(200, 50));
80 signUpButton.addActionListener(e -> {
81     String name = registerNameField.getText();
82     String password = new String(registerPasswordField.getPassword());
83
84     // Validación simple
85     if (name.equals("jesus") && password.equals("jesus123")) {
86         JOptionPane.showMessageDialog(this, "Inicio de sesión exitoso");
87         this.dispose(); // Cierra la ventana actual
88         FrmPrincipal mainFrame = new FrmPrincipal();
89         mainFrame.setVisible(true); // Abre la ventana principal
90     } else {
91         JOptionPane.showMessageDialog(this, "Credenciales inválidas. Intenta de nuevo.", "Error", JOptionPane.ERROR_MESSAGE);
92     }
93 });
94 gbc.gridy = 3;
95 rightPanel.add(signUpButton, gbc);
96
97 add(rightPanel);
98
99 private void startFloatAnimation() {
100     floatTimer = new Timer(30, (ActionEvent e) -> { // Intervalo más largo para mayor fluidez

```



```

101 // Animación sinusoidal para "flotar"
102 floatAngle += 0.05; // Incremento más pequeño para ralentizar el movimiento
103 int offsetY = (int) (Math.sin(floatAngle) * floatAmplitude);
104
105 // Recalcula la posición horizontal centrada
106 int iconX = (leftPanel.getWidth() - iconLabel.getWidth()) / 2;
107
108 // Aplica la posición centrada y el desplazamiento vertical
109 iconLabel.setBounds(iconX, iconBaseY + offsetY, 150, 150);
110 });
111 floatTimer.start();
112 }
113
114 public static void main(String[] args) {
115     SwingUtilities.invokeLater(() -> {
116         new LoginUI().setVisible(true);
117     });
118 }
119
120 // Clase personalizada para el botón
121 class CustomButton extends JButton {
122     private boolean isHovered = false;
123
124     public CustomButton(String text) {
125         super(text);
126         setContentAreaFilled(false);
127         setFocusPainted(false);
128         setBorderPainted(false);
129         setFont(new Font("Arial", Font.BOLD, 16));
130         setForeground(Color.WHITE);
131         setCursor(new Cursor(Cursor.HAND_CURSOR));
132
133         addMouseListener(new java.awt.event.MouseAdapter() {
134             @Override
135             public void mouseEntered(java.awt.event.MouseEvent e) {
136                 isHovered = true;
137                 repaint();
138             }
139
140             @Override
141             public void mouseExited(java.awt.event.MouseEvent e) {
142                 isHovered = false;
143                 repaint();
144             }
145         });
146     }
147
148     @Override
149     protected void paintComponent(Graphics g) {
150         Graphics2D g2d = (Graphics2D) g.create();
151         g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
152
153         Color topColor = isHovered ? new Color(0, 120, 215) : new Color(33, 97, 140);
154         Color bottomColor = isHovered ? new Color(0, 85, 170) : new Color(0, 51, 102);
155
156         GradientPaint gradient = new GradientPaint(0, 0, topColor, 0, getHeight(), bottomColor);
157         g2d.setPaint(gradient);
158         g2d.fillRoundRect(0, 0, getWidth(), getHeight(), 30, 30);
159
160         g2d.setColor(new Color(0, 0, 0, 50));
161         g2d.fillRoundRect(2, 2, getWidth() - 4, getHeight() - 4, 30, 30);
162
163         super.paintComponent(g2d);
164         g2d.dispose();
165     }
166
167     @Override
168     protected void paintBorder(Graphics g) {
169     }
170 }
171

```

## Archivo FrmPrincipal.java

El archivo FrmPrincipal.java representa la interfaz principal del analizador léxico, diseñada para simular un entorno de desarrollo integrado (IDE). Utilizando la biblioteca Swing, esta clase combina características interactivas y estéticas para ofrecer un espacio de trabajo funcional. Entre sus componentes destacan un área de entrada para escribir código, un panel de resultados donde se muestra el análisis realizado, una barra lateral que simula un explorador de archivos y una barra de estado que comunica el estado actual del programa.

### Código Realizado:

```
1 package codigo;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import java.io.*;
6 import java.util.logging.*;
7 import javax.swing.*;
8
9 public class FrmPrincipal extends javax.swing.JFrame {
10
11     public FrmPrincipal() {
12         initComponents();
13         this.setLocationRelativeTo(null);
14         this.setTitle("Analizador Léxico - IDE Simulado");
15     }
16
17     private void initComponents() {
18         // Panel principal
19         mainPanel = new JPanel(new BorderLayout());
20         mainPanel.setBackground(new Color(30, 30, 30)); // Fondo oscuro principal
21
22         // Barra lateral simulada (como árbol de archivos)
23         JPanel sidebar = new JPanel();
24         sidebar.setPreferredSize(new Dimension(150, 0));
25         sidebar.setBackground(new Color(40, 40, 40));
26
27         JLabel lblExplorer = new JLabel("Explorador");
28         lblExplorer.setForeground(new Color(200, 200, 200));
29         lblExplorer.setFont(new Font("Consolas", Font.BOLD, 14));
30         lblExplorer.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
31
32         JList<String> fileTree = new JList<>(new String[]{"archivo.txt", "output.log", "README.md"});
33         fileTree.setBackground(new Color(50, 50, 50));
```



```
34 fileTree.setForeground(new Color(220, 220, 220));
35 fileTree.setFont(new Font("Consolas", Font.PLAIN, 14));
36 fileTree.setSelectionBackground(new Color(70, 130, 180));
37 fileTree.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
38
39 sidebar.setLayout(new BorderLayout());
40 sidebar.add(lblExplorer, BorderLayout.NORTH);
41 sidebar.add(fileTree, BorderLayout.CENTER);
42
43 // Barra de pestañas simulada
44 tabBar = new JPanel(new FlowLayout(FlowLayout.LEFT, 5, 5));
45 tabBar.setBackground(new Color(45, 45, 48));
46 tabBar.setPreferredSize(new Dimension(0, 35));
47
48 tab1 = new JLabel("archivo.txt");
49 tab1.setForeground(new Color(220, 220, 220));
50 tab1.setFont(new Font("Consolas", Font.BOLD, 14));
51 tab1.setBorder(BorderFactory.createCompoundBorder(
52     BorderFactory.createMatteBorder(0, 0, 2, 0, new Color(97, 218, 251)),
53     BorderFactory.createEmptyBorder(0, 10, 0, 10));
54
55 tabBar.add(tab1);
56
57 // Panel para entrada y números de línea
58 JPanel editorPanel = new JPanel(new BorderLayout());
59 editorPanel.setBackground(new Color(28, 28, 28));
60
61 // Números de línea
62 lineNumberArea = new JTextArea("1\n2\n3\n4\n5\n");
63 lineNumberArea.setFont(new Font("Consolas", Font.PLAIN, 16));
64 lineNumberArea.setBackground(new Color(35, 35, 35));
65 lineNumberArea.setForeground(new Color(150, 150, 150));
66 lineNumberArea.setEditable(false);
67
68 lineNumberArea.setMargin(new Insets(5, 5, 5, 5));
69
70 // Área de entrada de código
71 txtEntrada = new JTextArea();
72 txtEntrada.setFont(new Font("Consolas", Font.PLAIN, 16));
73 txtEntrada.setBackground(new Color(40, 44, 52));
74 txtEntrada.setForeground(new Color(200, 200, 200));
75 txtEntrada.setCaretColor(new Color(97, 218, 251));
76 txtEntrada.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
77 txtEntrada.setText("// Escribe tu código aquí...");
78 txtEntrada.setLineWrap(true);
79 txtEntrada.setWrapStyleWord(true);
80
81 JScrollPane scrollEntrada = new JScrollPane(txtEntrada);
82 scrollEntrada.setRowHeaderView(lineNumberArea);
83 scrollEntrada.setBorder(BorderFactory.createLineBorder(new Color(60, 63, 65)));
84
85 editorPanel.add(scrollEntrada, BorderLayout.CENTER);
86
87 // Área de resultados
88 txtResultado = new JTextArea();
89 txtResultado.setFont(new Font("Consolas", Font.PLAIN, 16));
90 txtResultado.setBackground(new Color(35, 35, 35));
91 txtResultado.setForeground(new Color(200, 200, 200));
92 txtResultado.setCaretColor(new Color(97, 218, 251));
93 txtResultado.setEditable(false);
94 txtResultado.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
95 txtResultado.setLineWrap(true);
96 txtResultado.setWrapStyleWord(true);
97 txtResultado.setText("// Los resultados del análisis aparecerán aquí...");
98
99 JScrollPane scrollResultado = new JScrollPane(txtResultado);
100 scrollResultado.setBorder(BorderFactory.createLineBorder(new Color(60, 63, 65)));
```



```
100 // Botón analizar
101 btnAnalizar = new JButton("Analizar Código");
102 btnAnalizar.setFont(new Font("Consolas", Font.BOLD, 16));
103 btnAnalizar.setBackground(new Color(97, 218, 251));
104 btnAnalizar.setForeground(new Color(28, 28, 28));
105 btnAnalizar.setFocusPainted(false);
106 btnAnalizar.setBorder(BorderFactory.createCompoundBorder(
107     BorderFactory.createLineBorder(new Color(60, 63, 65)),
108     BorderFactory.createEmptyBorder(10, 20, 10, 20));
109
110
111 btnAnalizar.addActionListener(evt -> {
112     btnAnalizar.setText("Cargando...");
113     btnAnalizar.setEnabled(false);
114     btnAnalizar.setBackground(new Color(70, 130, 180));
115
116     Timer timer = new Timer(2000, e -> {
117         btnAnalizarActionPerformed(evt);
118         btnAnalizar.setText("Analizar Código");
119         btnAnalizar.setEnabled(true);
120         btnAnalizar.setBackground(new Color(97, 218, 251));
121     });
122     timer.setRepeats(false);
123     timer.start();
124 });
125
126 btnAnalizar.addMouseListener(new java.awt.event.MouseAdapter() {
127     @Override
128     public void mouseEntered(java.awt.event.MouseEvent evt) {
129         btnAnalizar.setBackground(new Color(135, 206, 250));
130     }
131
132     @Override
133     public void mouseExited(java.awt.event.MouseEvent evt) {
134         btnAnalizar.setBackground(new Color(97, 218, 251));
135     }
136 });
137
138 // Barra de estado inferior
139 statusBar = new JLabel("Modo: Edición");
140 statusBar.setFont(new Font("Consolas", Font.PLAIN, 14));
141 statusBar.setForeground(new Color(220, 220, 220));
142 statusBar.setBackground(new Color(35, 35, 35));
143 statusBar.setOpaque(true);
144 statusBar.setBorder(BorderFactory.createEmptyBorder(5, 10, 5, 10));
145
146 // Panel inferior
147 JPanel bottomPanel = new JPanel(new BorderLayout());
148 bottomPanel.setBackground(new Color(28, 28, 28));
149 bottomPanel.add(btnAnalizar, BorderLayout.EAST);
150 bottomPanel.add(statusBar, BorderLayout.CENTER);
151
152 // Divisor ajustable
153 JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT, editorPanel, scrollResultado);
154 splitPane.setDividerLocation(300);
155 splitPane.setResizeWeight(0.7);
156 splitPane.setContinuousLayout(true);
157 splitPane.setBorder(null);
158
159 mainPanel.add(sideBar, BorderLayout.WEST);
160 mainPanel.add(tabBar, BorderLayout.NORTH);
161 mainPanel.add(splitPane, BorderLayout.CENTER);
162 mainPanel.add(bottomPanel, BorderLayout.SOUTH);
163
164 getContentPane().add(mainPanel);
165 pack();
```

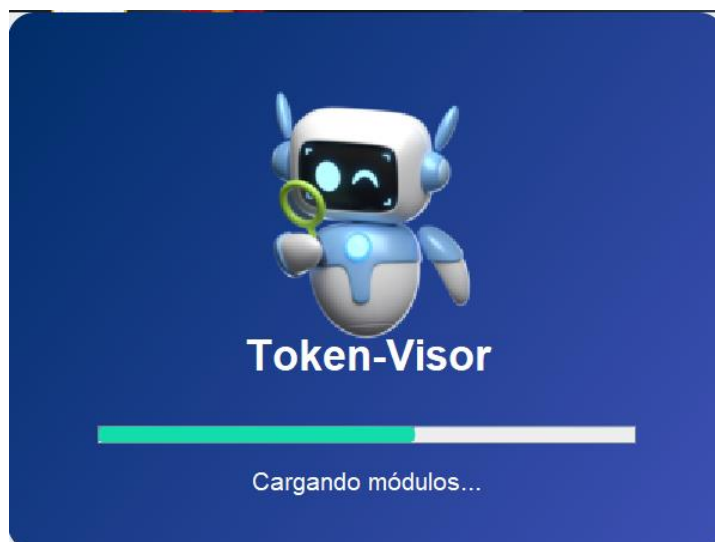


```
166 // Atajos de teclado
167 addKeyboardShortcuts();
168 }
169
170
171 private void addKeyboardShortcuts() {
172     txtEntrada.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW)
173         .put(KeyStroke.getKeyStroke(KeyEvent.VK_S, InputEvent.CTRL_DOWN_MASK), "save");
174     txtEntrada.getActionMap().put("save", new AbstractAction() {
175         @Override
176         public void actionPerformed(ActionEvent e) {
177             statusBar.setText("Archivo guardado.");
178         }
179     });
180
181     txtEntrada.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW)
182         .put(KeyStroke.getKeyStroke(KeyEvent.VK_R, InputEvent.CTRL_DOWN_MASK), "analyze");
183     txtEntrada.getActionMap().put("analyze", new AbstractAction() {
184         @Override
185         public void actionPerformed(ActionEvent e) {
186             btnAnalizarActionPerformed(null);
187         }
188     });
189 }
190
191 private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {
192     statusBar.setText("Analizando...");
193     File archivo = new File("archivo.txt");
194     try (PrintWriter escribir = new PrintWriter(archivo)) {
195         escribir.print(txtEntrada.getText());
196     } catch (FileNotFoundException ex) {
197         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
198     }
199
200     try (Reader lector = new BufferedReader(new FileReader("archivo.txt"))) {
201         Lexer lexer = new Lexer(lector);
202         StringBuilder resultado = new StringBuilder();
203         while (true) {
204             Tokens tokens = lexer.yylex();
205             if (tokens == null) {
206                 resultado.append("FIN");
207                 txtResultado.setText(resultado.toString());
208                 statusBar.setText("Análisis completado.");
209                 return;
210             }
211             switch (tokens) {
212                 case ERROR:
213                     resultado.append("Símbolo no definido\n");
214                     break;
215                 case Identificador:
216                 case Numero:
217                 case Reservadas:
218                     resultado.append(lexer.lexeme).append(" Es un ").append(tokens).append("\n");
219                     break;
220                 default:
221                     resultado.append("Token: ").append(tokens).append("\n");
222                     break;
223             }
224         }
225     } catch (IOException ex) {
226         Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
227     }
228 }
229
230 public static void main(String args[]) {
231     java.awt.EventQueue.invokeLater(() -> new FrmPrincipal().setVisible(true));
232 }
```

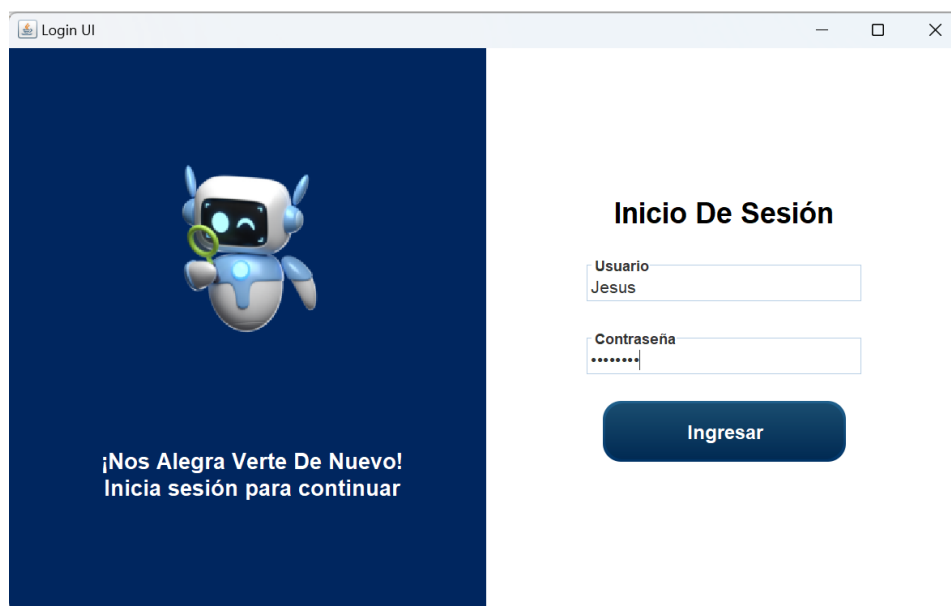
```
232 }
233
234 // Variables
235 private JPanel mainPanel;
236 private JPanel tabBar;
237 private JLabel tab1;
238 private JTextArea txtEntrada;
239 private JTextArea txtResultado;
240 private JTextArea lineNumberArea;
241 private JButton btnAnalizar;
242 private JLabel statusBar;
243 }
```

PANTALLAS RESULTANTES CON PRUEBAS DEL RECONOCIMIENTO DE LOS TOKENS

Pantalla Splash

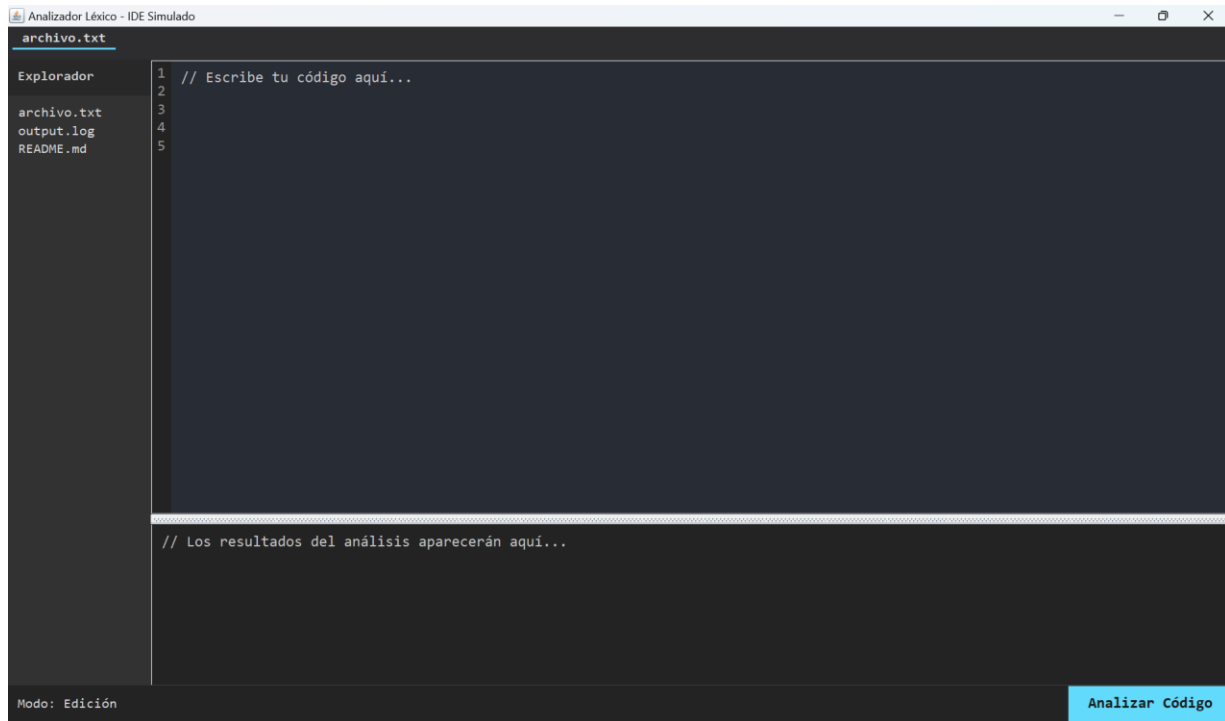


Pantalla Login

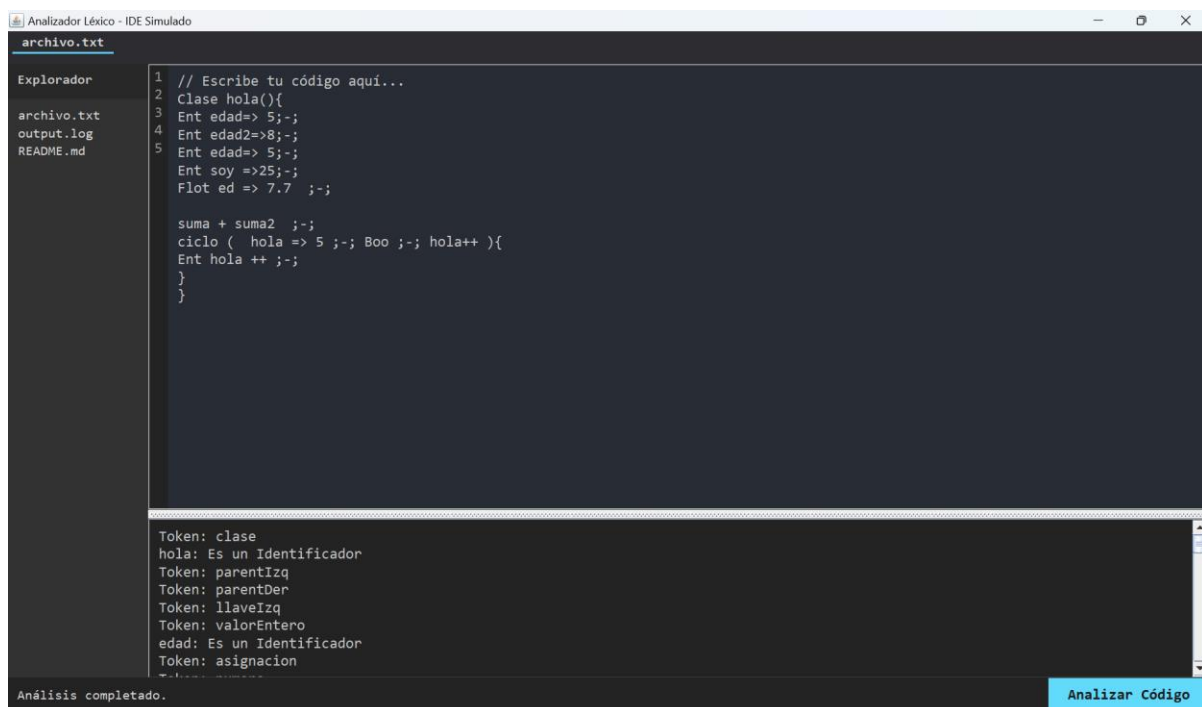




## Pantalla Principal



## Pantalla principal con demostración de la verificación de los tokens





## V. Conclusiones:

El objetivo principal del desarrollo de este proyecto fue implementar un analizador léxico que permitiera identificar y clasificar elementos sintácticos de un lenguaje de programación simulado. Este proyecto buscó no solo reforzar los conocimientos teóricos adquiridos sobre compiladores, sino también aplicar habilidades prácticas en el desarrollo de software, trabajando con herramientas y tecnologías modernas.

Para llevar a cabo el proyecto, se utilizó el lenguaje de programación **Java**, el cual proporcionó una plataforma sólida y orientada a objetos para estructurar el sistema. Dentro de las herramientas y librerías empleadas, destaca **JFlex**, que permitió generar un analizador léxico basado en expresiones regulares definidas en un archivo .flex. Esta herramienta automatizó gran parte del proceso de construcción del lexer, transformando reglas léxicas en código ejecutable. Asimismo, se usó **Swing**, la biblioteca gráfica de Java, para diseñar una interfaz de usuario amigable que simula las funcionalidades de un IDE básico. Componentes personalizados, como botones estilizados y paneles con efectos visuales, añadieron un toque profesional al diseño de la aplicación. Además, se incorporaron herramientas estándar de Java para gestionar archivos, manejar eventos y mostrar resultados en tiempo real.

Durante el desarrollo del proyecto uno de los principales desafíos fue diseñar reglas léxicas precisas en el archivo .flex, pues cualquier error mínimo en las expresiones regulares podía causar un mal funcionamiento del lexer, lo que podía provocar varias ejecuciones de prueba y corrección. Otro reto significativo fue la integración de la interfaz gráfica con la lógica del analizador léxico. Garantizar que las acciones del usuario en la interfaz fueran correctamente interpretadas por el sistema requería una sincronización cuidadosa entre la capa visual y el backend. Asimismo, implementar una experiencia de usuario intuitiva, con mensajes de error claros y una respuesta visual atractiva, representó un esfuerzo adicional.

A pesar de las dificultades, este proyecto me permitió desarrollar algunas habilidades importantes, como la escritura de expresiones regulares, la programación de interfaces gráficas intuitivas y agradables visualmente, así como la solución de problemas en tiempo real.