



MANUAL DE PRÁCTICAS



Nombre de la práctica	ANALIZADOR SINTACTICO (UNIDAD 5)			No.	5
Asignatura:	LENGUAJES Y AUTÓMATAS I	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES-3501	Duración de la práctica (Hrs)	5 horas

NOMBRE DEL ALUMNO: Vanesa Hernández Martínez

GRUPO: 3501

I. Competencia(s) específica(s):

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

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

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

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

Laboratorio de cómputo y equipo de cómputo personal.

III. Material empleado:

- Equipo de cómputo
- Software para desarrollo : Apache Netbeans IDE 2

ANALIZADOR LÉXICO

Descripción del problema

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

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

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

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

¿Qué es un analizador léxico?

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

Principales Funciones

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

Para qué Sirve

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

Importancia y Usos

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

Tokens

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

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

1. Operadores

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

2. Signos y constantes matemáticas

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

3. Delimitadores

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

4. Comentarios

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

5. Tipos de datos

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

6. Estructuras de control

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

7. Declaraciones y entrada/salida

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



8. Componentes de cadenas

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

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

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

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

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

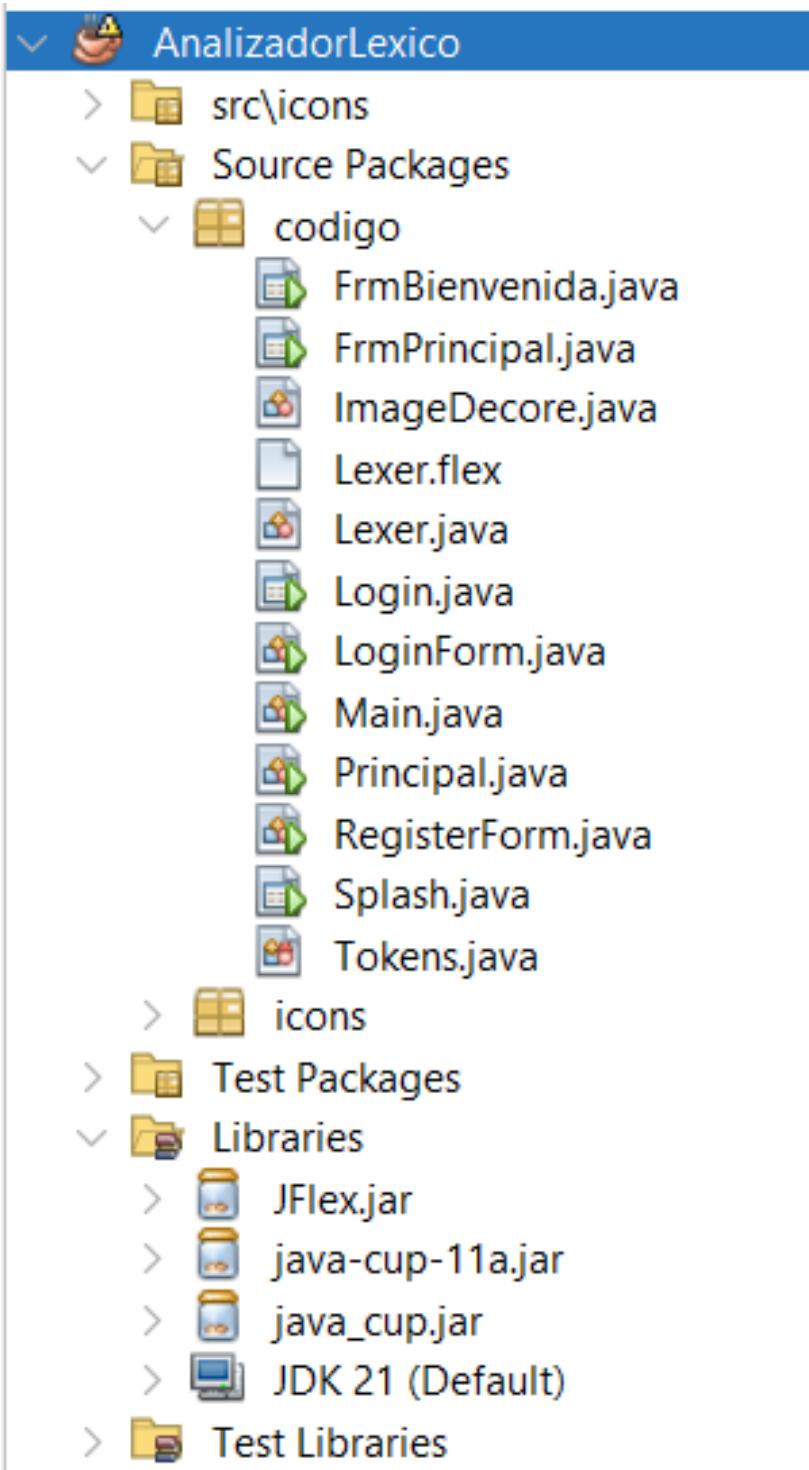


		Comando	Descripción
Comentarios	35	comentario	"#"
	36	inicioComentario	"{"
	37	finComentario	"}"
Tipos de Datos	38	entero	"ent"
	39	decimal	"dec"
	40	verdadero	"v"
	41	falso	"f"
	42	cadena	"cadena"
	43	sí	"sí"
Estructuras de control	44	siNo	"siNo"
	45	siNoSi	"siNoSi"
	46	cicloRepite	"repite"
	47	mientras	"mientras"
	48	hacer	"hacer"
	49	seleccionador	"seleccionar"
	50	caso	"caso"
	51	predeterminado	"predeterminado"
	52	detener	"detener"
	53		
	54	metodo	"metodo"
	55	arreglo	"arreglo"
Declaraciones	56	funcion	"fun"
	57	clase	"clase"
	58	procedimiento	"proc"
	59	principal	"principal"
	60	biblioteca	"biblioteca"
	61	imprimir	imprime
Entrada y salida de Datos	62	leer	leer
	63	importar	"importa"
	64	retornar	"retorna"
	65	nulo	"nulo"
	66	salto de linea	"\n"
Componentes de una cadena	67	caracteresEspeciales	":","\$"
	68	letrasMayusculas	[A-Z]
	69	letrasMinusculas	[a-z]
	70	digitos	[0-9]

Estructura del proyecto

El proyecto este compuesto por los siguientes archivos:

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



Lexer.flex

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

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

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



MANUAL DE PRÁCTICAS



```
36
37  /* OPERADORES RELACIONALES */
38
39  /* Operador Mayor */
40  ( ">" ) {lexeme=yytext(); return Mayor;}
41
42  /* Operador Menor */
43  ( "<" ) {lexeme=yytext(); return Menor;}
44
45  /* Operador Mayor o Igual */
46  ( ">=" ) {lexeme=yytext(); return MayorIgual;}
47
48  /* Operador Menor o Igual */
49  ( "<=" ) {lexeme=yytext(); return MenorIgual;}
50
51  /* Operador Comparación */
52  ( "==" ) {lexeme=yytext(); return Comparacion;}
53
54  /* Operador Distinto */
55  ( "!=" ) {lexeme=yytext(); return Distinto;}
56
57
58  /* OPERADORES LOGICOS */
59
60  /* Operador Y */
61  ( "&" ) {lexeme=yytext(); return Y;}
62
63  /* Operador O */
64  ( "|" ) {lexeme=yytext(); return O;}
65
66  /* Operador No */
67  ( "!" ) {lexeme=yytext(); return No;}
```

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



MANUAL DE PRÁCTICAS



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

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



MANUAL DE PRÁCTICAS



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

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



```
245 /* Retornar */
246 ( "retorna" ) {lexeme=yytext(); return Retornar;}
247
248 /* Metodo */
249 ( "metodo" ) {lexeme=yytext(); return Metodo;}
250
251 /* Nulo */
252 ( "nulo" ) {lexeme=yytext(); return Nulo;}
253
254 /* COMPONENTES DE UNA CADENA */
255
256 /* Caracteres Especiales */
257 ( ":" | "$" ) {lexeme=yytext(); return CaracteresEspeciales;}
258
259 /* Fin de Linea */
260 ( ";" ) {lexeme=yytext(); return P_coma;}
261
262 /* Marcador de inicio de algoritmo */
263 ( "main" ) {lexeme=yytext(); return Main;}
264
265 /* Constantes */
266 [A-Z_]+ {lexeme = yytext(); return Constante; }
267
268 /* Nombres de clases */
269 [A-Z][a-zA-Z0-9_]* {lexeme = yytext(); return NombreClase; }
270
271 /* Identificador */
272 (L)(D)* {lexeme=yytext(); return Identificador;}
273
274 /* Numero */
275 ("{"D}+")" | {D}+ {lexeme=yytext(); return Numero;}
276
277 /* Texto entre comillas dobles */
278 ("["^"\\"\\]*("\\\\.[^"\\"\\]*")")\\" { lexeme = yytext(); return Texto;}
```

```
280 /* Texto entre comillas simples */
281 ('[^'\\\"]*("\\\\.[^'\\\"]*)')\'' {lexeme = yytext(); return Texto;}
282
283 /* Error de análisis */
284 . {return ERROR;}
```

Lexer.java

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

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

```
1  /* The following code was generated by JFlex 1.4.3 on 04/01/25, 04:08 */
2
3  package codigo;
4  import static codigo.Tokens.*;
5
6  /**
7   * This class is a scanner generated by
8   * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9   * on 04/01/25, 04:08 from the specification file
10  * <ttoC:/Users/vanes/OneDrive/Documentos/hh/VANE_HP/hp/Nueva carpeta/UNIVERSIDAD/SEMESTRE 5/LENGUAJES Y AUTOMATAS I/Proyecto/AnalizadorLexico/AnalizadorLexico/src/codigo/Lexer.
11  */
12  class Lexer {
13
14  /** This character denotes the end of file */
15  public static final int YYEOF = -1;
16
17  /** initial size of the lookahead buffer */
18  private static final int ZZ_BUFSIZE = 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 =
```



```
36 private static final String ZZ_CMAP_PACKED =
37 "U1\0\A3\1\6\2\0\A1\3\2\0\0\A1\3\1\25\1\53\1\55" +
38 "U1\66\1\30\1\26\1\54\1\A5\1\4\0\A22\1\20\1\3\1\21" +
39 "U1\35\1\15\1\2\1\66\1\A0\A1\24\1\17\1\23\1\7\0\A0" +
40 "U1\4\1\A4\1\43\1\1\1\A4\1\4\1\1\41\1\37\1\36" +
41 "U2\1\A4\0\A1\3\1\A3\1\0\A0\A3\1\3\1\A0\A1\33" +
42 "U1\64\1\A7\1\56\1\10\1\7\1\A1\1\A1\4\1\A2\1\A1\11" +
43 "U1\63\1\A5\1\61\1\62\1\A1\1\A3\1\A2\1\A1\65\1\60" +
44 "U1\13\1\2\1\A3\1\51\1\27\1\52\1\43\0\1\71\35\0\1\67" +
45 "\uff40\0";
```

```
46 /**
47 * Translates characters to character classes
48 */
49 private static final char [] ZZ_CMAP = zzUnpackCMap(ZZ_CMAP_PACKED);
```

```
50 /**
51 * Translates DFA states to action switch labels
52 */
53 private static final int [] ZZ_ACTION = zzUnpackAction();
```

```
54 private static final String ZZ_ACTION_PACKED_0 =
```

```
"U1\0\A1\1\2\1\3\1\4\2\2\1\5\1\2" +
"U1\6\1\7\1\10\1\11\1\2\1\13\1\14\1\15" +
"U1\16\1\17\1\20\1\21\1\2\1\22\1\2\1\23" +
"U1\24\1\25\1\26\1\27\1\30\1\31\1\32\1\33" +
"U1\34\1\2\2\1\35\1\2\1\36\1\37\1\40\1\41" +
"U1\2\1\4\1\3\1\43\1\2\1\4\1\4\1\45" +
"U1\4\1\A7\1\50\1\51\1\52\1\53\1\54\1\52" +
"U1\5\1\A2\1\A0\1\72\1\56\1\1\2\1\57\1\60" +
"U1\0\1\2\1\61\1\72\1\62\1\63\1\64\1\4\1\2" +
"U1\6\1\3\1\2\1\66\1\10\2\1\67\1\1\2\1\70" +
"U2\2\1\7\1\5\1\1\72\1\73\1\2\1\74\1\75" +
"U7\2\1\7\1\2\1\1\77\1\2\1\100\1\101\3\2" +
"U1\102";
```

```
72 private static int [] zzUnpackAction() {
73     int [] result = new int[169];
74     int offset = 0;
75     offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);
76     return result;
77 }
```

```
78 /**
79 * Translates a state to a row index in the transition table
80 * @param packed packed string
81 * @param offset index in packed string
82 * @param result packed.length();
83 */
84 private static int zzUnpackAction(String packed, int offset, int [] result) {
85     int i = 0; /* index in packed string */
86     int j = offset; /* index in unpacked array */
87     int l = packed.length();
88     while (i < l) {
89         int count = packed.charAt(i++);
90         int value = packed.charAt(i++);
91         do result[j++] = value; while (--count > 0);
92     }
93     return j;
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 "U0\0\0\72\0\164\0\256\0\350\0\0\0122\0\0\015c\0\164" +
102 "U0\0196\0\0\01d0\0\0\020a\0\0\0244\0\0\027e\0\0\02b\0\0\02f2\0\0\72" +
103 "U0\72\0\0\032c\0\0\0366\0\0\03a\0\0\03da\0\0\72\0\0\72\0\0\72" +
104 "U0\2\0\0\0414\0\0\044\0\0\048\0\0\048\0\0\046\0\0\046\0\0\72" +
105 "U0\72\0\0\72\0\0\72\0\0\72\0\0\72\0\0\72\0\0\72\0\0\72" +
106 "U0\053\0\0\164\0\0\0570\0\0\05aa\0\0\0564\0\0\072\0\0\72\0\0\72" +
107 "U0\72\0\0\061\0\0\164\0\0\0658\0\0\0692\0\0\06cc\0\0\0706\0\0\0740" +
108 "U0\077a\0\0\07b4\0\0\07ee\0\0\0828\0\0\0862\0\0\072\0\0\72\0\0\72" +
109 "U0\72\0\0\72\0\0\72\0\0\72\0\0\72\0\0\089\0\0\08d6" +
110 "U0\0910\0\0\164\0\0\094a\0\0\0984\0\0\09be\0\0\09f\0\0\0a32\0\0\0a6c" +
```



MANUAL DE PRÁCTICAS



```
108     "\u0\uaa6\u0\uaeao\u0\uobla\u0\u164\u0\u0b54\u0\u0b8e\u0\u0bc8\u0\u0c02"+  
109     "\u0\u0c3c\u0\u0c76\u0\u0cb0\u0\u0cea\u0\u0d24\u0\u164\u0\u164\u0\u0dse"+  
110     "\u0\u0d98\u0\u164\u0\u0dd2\u0\u0e0c\u0\u0e46\u0\u0e80\u0\u0eba\u0\u0ef4"+  
111     "\u0\u0f2e\u0\u164\u0\u0f68\u0\u0fa2\u0\u0fdc\u0\u016\u0\u1050"+  
112     "\u0\u164\u0\u72\u0\u108a\u0\u164\u0\u0c0\u0\u0fe6\u0\u1138\u0\u1172"+  
113     "\u0\u1lac\u0\u1le6\u0\u1220\u0\u125a\u0\u164\u0\u294\u0\u12ce\u0\u1308"+  
114     "\u0\u1342\u0\u137e\u0\u13b6\u0\u13f0\u0\u142a\u0\u146\u0\u164\u0\u149e"+  
115     "\u0\u14d8\u0\u164\u0\u1512\u0\u154c\u0\u158c\u0\u15c0\u0\u15fa\u0\u164e"+  
116     "\u0\u164\u0\u1634\u0\u164\u0\u164\u0\u166\u0\u168a\u0\u16e2\u0\u171c"+  
117     "\u0\u1756\u0\u1790\u0\uca1\u0\u164\u0\u1804\u0\u183e\u0\u164\u0\u1878"+  
118     "\u0\u1bb2\u0\u18ec\u0\u1926\u0\u164\u0\u164\u0\u1960\u0\u199a\u0\u19d4"+  
119     "\u0\u164";  
120  
121     private static int [] zzUnpackRowMap() {  
122         int [] result = new int[169];  
123         int offset = 0;  
124         offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);  
125         return result;  
126     }  
127  
128     private static int zzUnpackRowMap(String packed, int offset, int [] result) {  
129         int i = 0; /* index in packed string */  
130         int j = offset; /* index in unpacked array */  
131         int l = packed.length();  
132         while (i < l) {  
133             int high = packed.charAt(i++) << 16;  
134             result[j++] = high | packed.charAt(i++);  
135         }  
136         return j;  
137     }  
138  
139     /**  
140      * The transition table of the DFA  
141      */  
142     private static final int [] ZZ_TRANS = zzUnpackTrans();
```

```
144 private static final String ZZ_TRANS_PACKED_C =  
145 "\\"1\2\0\1\3\1\4\1\5\1\6\1\7\1\8\1\9\1\0\" +  
146 "\\"1\1\1\1\1\1\2\1\1\3\1\4\1\5\1\6\1\7\1\8\1\9\1\0\" +  
147 "\\"1\2\0\1\1\2\1\1\2\2\1\2\3\1\4\1\5\1\6\1\7\1\8\1\9\1\0\" +  
148 "\\"1\3\0\1\3\1\2\2\2\3\1\3\3\1\3\4\2\3\1\3\5\4\" +  
149 "\\"1\3\6\1\1\3\1\3\7\1\4\0\1\4\1\4\2\1\3\1\4\4\" +  
150 "\\"1\4\5\1\1\4\6\1\4\7\1\5\0\1\1\5\1\5\2\1\3\1\5\3\" +  
151 "\\"1\5\4\1\1\5\5\1\3\1\5\1\5\7\1\6\0\1\1\7\3\0\" +  
152 "\\"2\2\3\1\0\1\1\3\1\5\0\1\3\2\1\3\1\0\1\3\1\1\0\" +  
153 "\\"1\0\3\1\0\1\1\3\1\4\7\2\0\1\1\5\1\2\0\1\1\5\4\0\" +  
154 "\\"2\2\3\1\0\1\1\3\1\6\2\2\1\3\1\6\3\8\1\3\1\0\\" +  
155 "\\"3\3\1\0\1\0\1\1\3\1\0\5\1\3\1\6\4\2\3\1\5\0\" +  
156 "\\"2\3\1\0\1\0\1\3\1\5\0\1\3\1\0\1\3\1\0\1\0\\" +  
157 "\\"3\1\2\1\6\5\5\0\2\3\1\0\1\3\1\3\1\6\3\3\0\" +  
158 "\\"1\1\6\7\3\2\1\5\0\1\3\1\1\0\1\3\1\1\0\1\0\3\0\" +  
159 "\\"5\0\2\1\3\1\0\4\1\3\1\7\0\4\3\1\5\0\3\3\0\" +  
160 "\\"1\1\0\3\7\3\1\2\1\3\1\0\3\5\0\2\3\1\0\1\1\7\1\" +  
161 "\\"3\1\3\1\7\2\1\4\3\1\5\0\3\3\1\0\1\7\0\3\1\1\0\" +  
162 "\\"1\0\3\1\5\0\2\1\2\1\0\1\0\3\1\7\3\1\5\0\3\3\0\" +  
163 "\\"1\0\1\0\7\3\1\1\0\1\0\3\5\0\2\3\1\0\1\1\2\3\0\" +  
164 "\\"1\5\0\1\1\3\1\7\4\1\3\1\0\7\3\1\1\0\1\0\1\0\3\" +  
165 "\\"2\1\0\1\1\7\5\4\0\1\1\7\6\0\0\1\7\7\3\0\1\0\1\00\" +  
166 "\\"3\3\0\1\1\0\1\7\3\0\1\1\0\2\3\0\1\0\1\0\3\6\5\0\1\10\4\" +  
167 "\\"1\0\1\0\1\0\5\6\4\1\0\1\0\6\5\3\0\2\3\1\0\1\0\4\3\0\" +  
168 "\\"1\1\0\7\4\1\3\1\5\0\1\3\1\1\0\1\1\0\1\3\1\0\1\0\7\3\0\" +  
169 "\\"1\1\0\1\0\3\1\5\0\2\3\1\0\1\1\0\1\1\3\1\5\0\3\3\0\" +  
170 "\\"1\0\1\3\1\3\1\1\1\4\3\1\1\1\2\3\1\0\1\0\3\5\0\" +  
171 "\\"2\3\1\0\1\3\1\5\0\3\3\1\0\1\3\1\4\0\3\1\1\3\3\0\" +  
172 "\\"2\3\1\1\0\1\0\3\2\5\0\1\1\1\4\5\1\0\2\3\1\0\1\0\7\0\" +  
173 "\\"1\4\3\1\1\5\4\3\1\5\0\3\3\1\0\1\7\3\1\0\1\0\7\0\" +  
174 "\\"1\0\3\1\5\0\2\3\1\0\1\1\3\1\5\0\1\1\3\1\1\1\6\0\" +  
175 "\\"1\2\3\1\0\1\0\9\1\1\0\1\0\2\3\1\0\1\0\9\2\3\1\0\1\0\7\0\" +  
176 "\\"1\1\3\1\5\0\1\1\7\7\3\1\2\3\1\0\1\3\1\1\0\1\0\3\3\0\" +  
177 "\\"5\0\2\3\1\0\1\2\0\1\0\3\5\0\5\0\3\3\1\0\0\7\0\" +  
178 "\\"7\3\1\1\0\1\0\3\5\0\2\3\1\0\1\0\1\2\1\0\1\0\3\0\"
```



178 "X\7\3\11\0\10\3\1\5\0\2\3\1\0\1\121\10\3"+
179 "X\5\0\3\3\1\0\7\3\11\0\10\3\1\0\2\3"+
180 "X\1\0\2\3\1\0\3\6\3\1\15\0\3\3\1\0\7\3"+
181 "X\1\0\10\3\1\0\2\3\1\0\11\3\1\0\3\3"+
182 "X\1\0\7\3\11\0\4\3\1\122\3\1\3\5\0\2\3"+
183 "X\1\0\5\3\1\123\3\3\1\5\0\3\3\1\0\7\3"+
184 "X\1\0\10\3\1\0\2\3\1\0\2\3\1\24\3"+
185 "X\1\0\3\3\1\0\7\3\11\0\10\3\1\0\2\3"+
186 "X\1\0\6\3\1\125\2\3\1\5\0\2\3\1\0\7\3"+
187 "X\1\0\10\3\1\0\2\3\1\0\4\3\1\126\4\3"+
188 "X\5\0\3\3\1\0\7\3\11\0\10\3\1\0\2\3"+
189 "X\1\0\11\3\1\0\3\3\1\0\3\3\1\127\3\3"+
190 "X\1\0\10\3\1\0\2\3\1\0\5\3\1\130\3\3"+
191 "X\5\0\3\3\1\0\7\3\11\0\10\3\1\0\2\3"+
192 "X\1\0\1\1\3\1\0\5\3\1\0\3\1\0\7\3\1\11\0"+
193 "X\1\0\3\5\0\2\3\1\0\11\3\1\0\3\3\1\0\7\3"+
194 "X\7\3\11\0\1\3\1\132\6\3\1\0\6\1\6\7\5\1\0"+
195 "X\5\3\7\5\1\0\2\3\1\0\2\3\1\133\6\3\1\15\0"+
196 "X\3\3\1\0\7\3\11\0\4\3\1\134\3\3\5\0"+
197 "X\2\3\1\0\1\135\10\3\1\5\0\3\3\1\0\7\3"+
198 "X\1\0\10\3\1\0\2\3\1\0\11\3\1\5\0\3\3"+
199 "X\1\0\2\3\1\136\4\3\11\0\10\3\1\0\2\3"+
200 "X\1\0\11\3\1\5\0\3\3\1\0\5\3\1\137\1\3"+
201 "X\1\0\10\3\1\0\1\140\70\0\2\3\1\0\2\3"+
202 "X\1\141\6\3\1\5\0\3\1\0\3\1\0\7\3\1\11\0\1\3"+
203 "X\1\142\6\3\1\5\0\2\3\1\0\6\3\1\143\2\3"+
204 "X\5\0\3\3\1\0\7\3\11\0\10\3\1\0\2\3"+
205 "X\1\0\1\144\3\1\3\1\145\4\3\15\0\3\3\1\0"+
206 "X\7\3\11\0\10\3\1\0\2\3\1\0\11\3\1\5\0"+
207 "X\4\3\1\5\0\3\3\1\0\1\140\70\0\2\3\1\0\2\3"+
208 "X\2\3\1\0\11\3\1\5\0\3\3\1\0\7\3\1\11\0"+
209 "X\6\3\1\147\1\3\1\5\0\2\3\1\0\1\13\1\5\0"+
210 "X\1\150\2\3\1\0\7\3\11\0\13\3\1\151\4\3"+
211 "X\5\0\2\3\1\0\11\3\1\5\0\3\3\1\0\7\3"+
212 "X\1\0\3\3\1\152\4\3\5\0\2\3\1\0\4\3"+

212 "X\1\1\0\3\3\1\152\4\3\5\0\2\3\1\0\4\3"+
213 "X\1\6\3\4\3\15\0\3\3\1\0\7\3\11\0\10\3"+
214 "X\5\0\2\3\1\0\1\13\1\5\0\1\153\2\3\1\0"+
215 "X\7\3\11\0\10\3\1\0\2\3\1\0\11\3\1\5\0"+
216 "X\3\3\1\0\7\3\11\0\3\1\3\1\154\4\3\5\0"+
217 "X\2\3\1\0\4\3\1\155\4\3\15\0\3\3\1\0"+
218 "X\7\3\11\0\10\3\1\0\2\3\1\0\10\5\3\1\125"+
219 "X\3\3\1\0\5\3\1\0\1\0\7\3\1\11\0\10\3\5\0"+
220 "X\2\3\1\0\4\3\1\156\4\3\15\0\3\3\1\0"+
221 "X\7\3\11\0\10\3\1\0\2\3\1\0\11\3\1\5\0"+
222 "X\3\3\1\0\7\3\11\0\3\1\3\1\157\4\3\15\0"+
223 "X\2\3\1\0\1\16\0\10\3\1\5\0\3\1\10\7\3"+
224 "X\1\0\10\3\1\0\2\3\1\0\11\3\1\5\0\2\3"+
225 "X\1\161\1\0\7\3\11\0\10\3\6\0\1\140\4\3\0"+
226 "X\1\162\24\0\2\3\1\0\14\3\1\163\4\3\15\0"+
227 "X\3\3\1\0\7\3\11\0\10\3\6\0\2\3\1\0\7\3"+
228 "X\1\3\15\0\3\3\1\0\7\3\11\0\3\3\1\164"+
229 "X\4\3\5\0\2\3\1\0\1\3\1\165\7\3\1\5\0"+
230 "X\3\3\1\0\7\3\11\0\10\3\5\0\2\3\1\0\7\3"+
231 "X\1\3\15\0\3\3\1\0\7\3\11\0\166\7\3"+
232 "X\5\0\2\3\1\0\1\3\1\167\7\3\1\5\0\3\3"+
233 "X\1\0\7\3\11\0\10\3\5\0\2\3\1\0\5\3"+
234 "X\1\170\3\3\1\5\0\3\3\1\0\7\3\11\0\10\3"+
235 "X\5\0\2\3\1\0\1\171\10\3\1\5\0\3\3\1\0"+
236 "X\7\3\11\0\10\3\1\0\2\3\1\0\11\3\1\5\0"+
237 "X\1\172\2\3\1\0\7\3\11\0\10\3\5\0\2\3"+
238 "X\1\0\11\3\1\5\0\3\3\1\0\2\3\1\173\4\3"+
239 "X\1\0\10\3\1\0\2\3\1\0\11\3\1\5\0\2\3"+
240 "X\1\0\7\3\11\0\10\3\5\0\2\3\1\0\5\3"+
241 "X\1\0\11\3\1\5\0\1\175\2\3\1\0\7\3\1\10"+
242 "X\1\0\3\5\0\2\3\1\0\11\3\1\5\0\1\176\2\3"+
243 "X\1\0\7\3\11\0\10\3\5\0\2\3\1\0\2\3"+
244 "X\1\177\6\3\1\5\0\3\3\1\0\7\3\11\0\10\3"+
245 "X\5\0\2\3\1\0\1\3\1\200\7\3\1\5\0\3\3"+
246 "X\1\0\7\3\11\0\10\3\5\0\2\3\1\0\11\3"+



```

247 "1\5\0\3\3\1\0\7\3\1\1\0\1\3\1\1\2\0\1\%3"+  

248 "\5\0\2\3\1\0\1\3\1\1\0\1\2\0\2\4\3\1\5\0\3\3"+  

249 "\1\2\0\3\1\3\1\1\0\1\0\3\0\2\1\3\1\0\2\1\3"+  

250 "\1\2\0\3\1\3\1\5\0\3\1\1\0\7\3\1\1\0\1\0\3"+  

251 "\5\0\2\3\1\3\1\0\1\2\0\4\2\0\3\1\5\0\3\1\3\1\0"+  

252 "\1\3\1\1\0\1\0\3\1\5\0\1\2\3\1\1\0\1\1\3\1\5\0"+  

253 "\3\1\1\0\3\1\3\1\1\0\1\5\3\1\2\0\5\2\3\5\0"+  

254 "\2\3\1\1\0\2\3\1\2\0\6\6\3\1\5\0\3\1\1\0"+  

255 "\1\3\1\1\0\1\0\3\1\5\0\2\3\1\1\0\1\2\0\7\1\0\3"+  

256 "\1\5\0\3\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\2\3\2"+  

257 "\1\0\1\1\3\1\5\0\3\1\3\1\0\7\3\1\1\0\1\3\1\0"+  

258 "\1\2\1\0\1\3\1\5\0\1\2\3\1\0\1\1\3\1\2\1\1\3"+  

259 "\1\5\0\3\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\2\3\2"+  

260 "\1\0\1\3\1\1\2\2\4\3\1\5\0\3\3\1\0\1\0\7\3\3"+  

261 "\1\1\0\1\0\3\1\5\0\2\2\3\1\1\0\4\3\1\1\2\1\3\1\3"+  

262 "\1\5\0\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\2\3\2"+  

263 "\1\0\1\2\1\1\0\1\3\1\5\0\3\1\3\1\1\0\7\3\1\1\0"+  

264 "\1\0\3\1\5\0\2\1\3\1\1\0\1\3\1\2\1\5\6\3\1\5\0"+  

265 "\3\1\1\0\1\0\7\3\1\1\0\1\0\3\5\0\1\2\3\1\1\0"+  

266 "\1\1\3\1\5\0\1\2\1\6\2\3\1\1\0\7\3\1\1\0\1\0\3\3"+  

267 "\3\0\2\3\1\1\0\1\1\3\1\5\0\3\1\3\1\1\0\1\0\7\3\3"+  

268 "\1\1\0\1\3\1\1\2\1\7\4\3\1\5\0\2\3\1\1\0\1\0\4\3\3"+  

269 "\1\2\2\0\1\3\1\5\0\3\1\3\1\1\0\7\3\1\1\0\1\0\3\3"+  

270 "\1\5\0\2\3\1\1\0\1\1\3\1\5\0\1\1\3\1\1\2\2\1\1\3\3"+  

271 "\1\1\0\1\7\3\1\1\0\1\0\3\5\0\1\2\3\1\1\0\1\2\2\2\1\3\3"+  

272 "\1\0\3\1\5\0\3\3\1\1\0\7\3\1\1\0\1\0\1\0\3\5\0\1\0\7\3\3"+  

273 "\2\3\1\1\0\1\1\3\1\5\0\1\3\1\1\2\2\3\1\1\3\1\1\0\1\0\7\3\3"+  

274 "\1\3\1\1\0\1\0\1\0\3\5\0\2\2\3\1\1\0\1\1\3\1\5\0\1\0\7\3\3"+  

275 "\1\2\2\4\1\2\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\0\2\3\3"+  

276 "\1\0\1\1\1\3\1\5\0\1\3\1\3\1\0\7\3\1\1\0\1\0\4\3\3"+  

277 "\1\2\2\5\1\3\1\5\0\1\2\2\3\1\1\0\4\3\1\1\2\2\1\1\3\3"+  

278 "\1\5\0\1\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\0\2\3\3"+  

279 "\1\0\1\1\3\1\5\0\1\3\1\1\2\2\1\1\3\1\1\0\1\0\7\3\3"+  

280 "\1\1\0\1\0\3\1\5\0\1\2\2\3\1\1\0\2\3\1\1\2\3\0\1\0\7\3\3"+  

281 "\1\5\0\2\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\0\2\3\3"+  

282 "\1\0\1\1\3\1\5\0\1\3\1\1\0\7\3\1\1\0\1\0\3\3"+

283 "\1\2\3\1\4\3\1\5\0\1\2\3\1\1\0\1\1\3\1\5\0\1\1\3\3"+  

284 "\1\2\3\2\1\3\1\1\0\7\3\1\1\0\1\0\3\1\5\0\1\2\3\3"+  

285 "\1\1\0\1\1\3\1\5\0\1\2\3\1\1\0\7\3\1\1\0\1\2\3\1\1\0\1\0\7\3\3"+  

286 "\1\0\3\1\5\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\2\3\1\1\0\1\0\7\3\3"+  

287 "\1\3\1\1\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\2\3\1\1\0\1\0\7\3\3"+  

288 "\1\3\1\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\2\3\1\1\0\1\0\7\3\1\1\0\1\0\7\3\3"+  

289 "\1\0\3\1\5\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\7\3\3"+  

290 "\1\3\1\1\0\1\0\7\3\1\1\0\1\0\3\1\5\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\7\3\3"+  

291 "\1\3\1\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\3\1\5\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\0\7\3\3"+  

292 "\1\0\3\1\5\0\1\2\3\1\1\0\1\1\3\1\5\0\1\3\1\1\0\1\0\7\3\1\1\0\1\0\7\3\3"+  

293 "\1\3\1\1\0\1\5\3\1\1\2\4\0\2\3\1\5\0\1\2\3\1\1\0\1\0\7\3\1\1\0\1\2\3\1\1\0\1\0\7\3\3"+  

294 "\1\1\3\1\1\5\0\1\3\1\1\0\7\3\1\1\0\1\1\3\1\1\2\4\1"+  

295 "\1\3\1\5\0\1\2\3\1\1\0\1\1\3\1\1\0\1\1\3\1\1\2\4\2"+  

296 "\1\1\3\1\1\0\1\3\1\1\0\1\1\3\1\1\0\1\1\3\1\1\0\1\1\3\1\1\0\1\0\7\3\3"+  

297 "\1\4\4\1\0\1\3\1\5\0\3\3\1\1\0\1\7\3\1\1\0\1\0\1\0\3\3"+  

298 "\1\0\2\3\1\1\0\1\1\3\1\5\0\1\3\1\1\0\1\2\3\1\1\0\1\1\3\3"+  

299 "\1\1\0\1\7\3\1\1\0\1\0\3\5\0\1\2\3\1\1\0\1\1\3\3"+  

300 "\1\5\0\1\2\4\5\2\3\1\1\0\7\3\1\1\0\1\0\3\5\0\1\0\7\3\3"+  

301 "\1\3\1\1\0\1\1\3\1\1\2\4\6\7\3\1\1\5\0\1\3\1\1\0\1\0\7\3\3"+  

302 "\1\3\1\1\0\1\0\1\3\1\1\0\2\3\1\1\0\1\1\3\1\1\3\5\0\1\0\7\3\3"+  

303 "\1\3\1\1\2\4\7\1\3\1\1\0\7\3\1\1\0\1\1\3\1\1\0\1\0\7\3\3"+  

304 "\1\3\1\1\0\1\1\3\1\1\3\1\5\0\3\3\1\1\0\7\3\1\1\0\1\1\3\1\1\0\1\0\7\3\3"+  

305 "\1\2\5\0\1\3\1\5\0\2\3\1\1\0\1\1\3\1\1\3\5\0\1\0\7\3\3"+  

306 "\1\0\1\3\1\1\0\1\3\1\3\1\1\0\7\3\1\1\0\1\3\1\1\0\1\3\3"+  

307

308 □ private static int [] zzUnpackTrans () {  

309     int [] result = new int [6670];  

310     int offset = 0;  

311     offset = zzUnpackTrans (zz_TRANS_PACKED_0 , offset , result);  

312     return result;  

313 }  

314  

315 □ private static int zzUnpackTrans (String packed , int offset , int [] result) {  

316     int i = 0; /* index in packed string */  

317     int j = offset; /* index in unpacked array */  

318 }

```



MANUAL DE PRÁCTICAS



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



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

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



```
449  private static char [] zzUnpackMap(String packed) {
450      char [] map = new char[0x10000];
451      int i = 0; /* index in packed string */
452      int j = 0; /* index in unpacked array */
453      while (i < 162) {
454          int count = packed.charAt(i++);
455          char value = packed.charAt(i++);
456          do map[j++] = value; while (--count > 0);
457      }
458      return map;
459  }
460
461
462  /**
463   * Refills the input buffer.
464   *
465   * @return      <code>false</code>, iff there was new input.
466   *
467   * @exception  java.io.IOException if any I/O-Error occurs
468   */
469  private boolean zzRefill() throws java.io.IOException {
470
471      /* first: make room (if you can) */
472      if (zzStartRead > 0) {
473          System.arraycopy(zzBuffer, zzStartRead,
474                          zzBuffer, 0,
475                          zzEndRead-zzStartRead);
476
477          /* translate stored positions */
478          zzEndRead-= zzStartRead;
479          zzCurrentPos-= zzStartRead;
480          zzMarkedPos-= zzStartRead;
481          zzStartRead = 0;
482      }
483  }
```

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



MANUAL DE PRÁCTICAS



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

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



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

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



```
672     zzState = ZZ_LEXSTATE[zzLexicalState];
673
674     zzForAction: {
675         while (true) {
676
677             if (zzCurrentPosL < zzEndReadL)
678                 zzInput = zzBufferL[zzCurrentPosL++];
679             else if (zzAtEOF) {
680                 zzInput = YYEOF;
681                 break zzForAction;
682             }
683             else {
684                 // store back cached positions
685                 zzCurrentPos = zzCurrentPosL;
686                 zzMarkedPos = zzMarkedPosL;
687                 boolean eof = zzRefill();
688                 // get translated positions and possibly new buffer
689                 zzCurrentPosL = zzCurrentPos;
690                 zzMarkedPosL = zzMarkedPos;
691                 zzBufferL = zzBuffer;
692                 zzEndReadL = zzEndRead;
693                 if (eof) {
694                     zzInput = YYEOF;
695                     break zzForAction;
696                 }
697                 else {
698                     zzInput = zzBufferL[zzCurrentPosL++];
699                 }
700             }
701         }
702         int zzNext = zzTransL[zzRowMapL[zzState] + zzCMapL[zzInput]];
703         if (zzNext == -1) break zzForAction;
704         zzState = zzNext;
705
706         int zzAttributes = zzAttrL[zzState];
707
708         int zzAttributes = zzAttrL[zzState];
709         if ( (zzAttributes & 1) == 1 ) {
710             zzAction = zzState;
711             zzMarkedPosL = zzCurrentPosL;
712             if ( (zzAttributes & 8) == 8 ) break zzForAction;
713         }
714     }
715
716     // store back cached position
717     zzMarkedPos = zzMarkedPosL;
718
719     switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
720     case 2:
721         { lexeme=yytext(); return Identificador;
722         }
723     case 67: break;
724     case 13:
725         { return No;
726         }
727     case 68: break;
728     case 24:
729         { return LlaveApertura;
730         }
731     case 69: break;
732     case 38:
733         { return Asignacion;
734         }
735     case 70: break;
736     case 12:
737         { return Menor;
738         }
739     case 71: break;
740     case 28:
741         { return Comentario;
```

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



MANUAL DE PRÁCTICAS



```
743     case 72: break;
744     case 36:
745     { return InicioComentario;
746     }
747     case 73: break;
748     case 47:
749     { return Positivo;
750     }
751     case 74: break;
752     case 20:
753     { return ParentesisApertura;
754     }
755     case 75: break;
756     case 46:
757     { return Entero;
758     }
759     case 76: break;
760     case 57:
761     { return CicloRepite;
762     }
763     case 77: break;
764     case 63:
765     { return Principal;
766     }
767     case 78: break;
768     case 55:
769     { return Hacer;
770     }
771     case 79: break;
772     case 5:
773     { return Falso;
774     }
775     case 80: break;
776     case 51:
777     }
```

```
779     case 81: break;
780     case 65:
781     { return Seleccionador;
782     }
783     case 82: break;
784     case 34:
785     { lexeme=yytext(); return Reservadas;
786     }
787     case 83: break;
788     case 17:
789     { return Potencia;
790     }
791     case 84: break;
792     case 48:
793     { return Negativo;
794     }
795     case 85: break;
796     case 60:
797     { return Retornar;
798     }
799     case 86: break;
800     case 16:
801     { return Modulo;
802     }
803     case 87: break;
804     case 3:
805     { lexeme=yytext(); return Número;
806     }
807     case 88: break;
808     case 64:
809     { return Bibliotecas;
810     }
811     case 89: break;
812     case 61:
813     { return Detener;
814     }
```



MANUAL DE PRÁCTICAS



```
816         }
817     case 35:
818         {
819             return Si;
820         }
821     case 91: break;
822     case 25:
823         {
824             return LlaveCierre;
825         }
826     case 92: break;
827     case 54:
828         {
829             return Caso;
830         }
831     case 93: break;
832     case 50:
833         {
834             return Nulo;
835         }
836     case 94: break;
837     case 44:
838         {
839             return Distinto;
840         }
841     case 26:
842         {
843             return ComillaDoble;
844         }
845     case 96: break;
846     case 33:
847         {
848             return AdmiracionInicio;
849         }
850     case 97: break;
851     case 29:
852         {
853             return Verdadero;
854         }
855     case 98: break;
856     case 27:
857         {
858             return ComillaSimple;
859         }
860     case 99: break;
861     case 8:
862         {
863             return Suma;
864         }
865     case 100: break;
866     case 49:
867         {
868             return Decimal;
869         }
870     case 101: break;
871     case 23:
872         {
873             return CorcheteCierre;
874         }
875     case 102: break;
876     case 59:
877         {
878             return Importar;
879         }
880     case 103: break;
881     case 66:
882         {
883             return Predeterminado;
884         }
885     case 104: break;
886     case 32:
887         {
888             return InterrogacionFin;
889         }
890     case 105: break;
891     case 52:
892         {
893             return Sino;
894         }
895     case 106: break;
896     case 18:
897         {
898             return Punto;
899         }
900     case 107: break;
```

```
848         case 27:
849             {
850                 return ComillaSimple;
851             }
852         case 99: break;
853         case 8:
854             {
855                 return Suma;
856             }
857         case 100: break;
858         case 49:
859             {
860                 return Decimal;
861             }
862         case 101: break;
863         case 23:
864             {
865                 return CorcheteCierre;
866             }
867         case 102: break;
868         case 59:
869             {
870                 return Importar;
871             }
872         case 103: break;
873         case 66:
874             {
875                 return Predeterminado;
876             }
877         case 104: break;
878         case 32:
879             {
880                 return InterrogacionFin;
881             }
882         case 105: break;
883         case 52:
884             {
885                 return Sino;
886             }
887         case 106: break;
888         case 18:
889             {
890                 return Punto;
891             }
892         case 107: break;
```



MANUAL DE PRÁCTICAS



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

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



```
955 |         case 125: break;
956 |         case 4:
957 |             { /*Ignore*/
958 |             }
959 |         case 126: break;
960 |         case 42:
961 |             { return MenorIgual;
962 |             }
963 |         case 127: break;
964 |         case 53:
965 |             { return Raiz;
966 |             }
967 |         case 128: break;
968 |         case 19:
969 |             { return Euler;
970 |             }
971 |         case 129: break;
972 |         case 7:
973 |             { return Igual;
974 |             }
975 |         case 130: break;
976 |         case 6:
977 |             { return Division;
978 |             }
979 |         case 131: break;
980 |         case 15:
981 |             { return O;
982 |             }
983 |         case 132: break;
984 |         default:
985 |             if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
986 |                 zzAtEOF = true;
987 |                 return null;
988 |             }
989 |             else {
990 |                 zzScanError(zz_NO_MATCH);
991 |             }
992 |         }
993 |     }
994 |
995 |
996 |
997 |
998 }
```

Tokens.java

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

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



MANUAL DE PRÁCTICAS



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

```
77     CaracteresEspeciales,  
78     P_coma,  
79     Main,  
80     Constante,  
81     NombreClase,  
82     Identificador,  
83     Numero,  
84     Texto,  
85     ERROR  
86     )
```

Principal.java

Este código tiene dos propósitos principales:

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

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

```
package codigo;

import java.io.File;

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

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

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



FrmPrincipal.java

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

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

Código:

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



MANUAL DE PRÁCTICAS



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

```
74 lineNumberPane.setBackground(new Color(220, 220, 220));
75 lineNumberPane.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
76
77 // Añadir JTextAreas al JScrollPane
78 JScrollPane scrollPane = new JScrollPane(txtEntrada);
79 scrollPane.setRowHeaderView(lineNumberPane);
80
81 // Botón Analizar
82 btnAnalizar.setFont(new Font("Tahoma", Font.BOLD, 16));
83 btnAnalizar.setText("Analizar");
84
85 btnAnalizar.setBackground(new Color(50, 150, 250));
86 btnAnalizar.setForeground(Color.WHITE);
87 btnAnalizar.setFocusPainted(false);
88 btnAnalizar.setBorder(BorderFactory.createCompoundBorder(
89     BorderFactory.createLineBorder(Color.BLUE, 1),
90     BorderFactory.createEmptyBorder(10, 20, 10, 20)
91 ));
92 btnAnalizar.addActionListener(new java.awt.event.ActionListener() {
93     public void actionPerformed(java.awt.event.ActionEvent evt) {
94         btnAnalizarActionPerformed(evt);
95     }
96 });
97
98 // Botón Borrar
99 btnBorrar.setFont(new Font("Tahoma", Font.BOLD, 16));
100 btnBorrar.setText("Borrar");
101 btnBorrar.setBackground(new Color(200, 50, 50));
102 btnBorrar.setForeground(Color.WHITE);
103 btnBorrar.setFocusPainted(false);
104 btnBorrar.setBorder(BorderFactory.createCompoundBorder(
105     BorderFactory.createLineBorder(Color.RED, 1),
106     BorderFactory.createEmptyBorder(5, 15, 5, 15)
107 ));
108 btnBorrar.addActionListener(new java.awt.event.ActionListener() {
109     public void actionPerformed(java.awt.event.ActionEvent evt) {
110         btnBorrarActionPerformed(evt);
111     }
112 });
```



MANUAL DE PRÁCTICAS



```
110         txtEntrada.setText("");
111     }
112   });
113 
114   // JTextArea para resultados
115   txtResultado.setColumns(20);
116   txtResultado.setRows(5);
117   txtResultado.setEditable(false);
118   txtResultado.setFont(new Font("Consolas", Font.PLAIN, 16));
119   jScrollPane.setviewportView(txtResultado);
120 
121   // Configurar Layout
122   javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
123   getContentPane().setLayout(layout);
124   layout.setHorizontalGroup(
125     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
126       .addGroup(layout.createSequentialGroup()
127         .addGap(150, 150, 150)
128         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
129           .addComponent(jScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 580, Short.MAX_VALUE)
130           .addComponent(scrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 580, Short.MAX_VALUE)
131           .addGroup(layout.createSequentialGroup()
132             .addGap(150, 150, 150)
133             .addComponent(btnAnalizar, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.swing.GroupLayout.PREFERRED_SIZE)
134             .addGap(50, 50, 50)
135             .addComponent(btnBorrar, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.swing.GroupLayout.PREFERRED_SIZE)
136             .addGap(0, 0, Short.MAX_VALUE))
137           .addGap(0, 0, Short.MAX_VALUE)))
138         .addGap(150, 150, 150)
139       )
140     .layout.setVerticalGroup(
141       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
142         .addGroup(layout.createSequentialGroup()
143           .addGap(150, 150, 150)
144           .addComponent(btnAnalizar, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
145           .addGap(40, 40, 40)
146           .addComponent(scrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 200, javax.swing.GroupLayout.PREFERRED_SIZE)
147           .addGap(40, 40, 40)
148           .addComponent(jScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)
149           .addGap(150, 150, 150)
150           .addComponent(btnBorrar, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
151           .addGap(150, 150, 150)
152         )
153       )
154     );
155 
156     pack();
157     updateLineNumbers();
158   }
159 
160   private void updateLineNumbers() {
161     String lineNumbers = new StringBuilder();
162     int lines = txtEntrada.getLineCount();
163     for (int i = 1; i <= lines; i++) {
164       lineNumbers.append(i).append("\n");
165     }
166     lineNumberPane.setText(lineNumbers.toString());
167   }
168 
169   private void btnAnalizarActionPerformed(java.awt.event.ActionEvent evt) {
170     File archivo = new File("archivo.txt");
171     PrintWriter escribir;
172     try {
173       escribir = new PrintWriter(archivo);
174       escribir.print(txtEntrada.getText());
175       escribir.close();
176     } catch (FileNotFoundException ex) {
177       Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
178     }
179 
180     try {
181       Reader lector = new BufferedReader(new FileReader("archivo.txt"));
182       Lexer lexer = new Lexer(lector);
183       String resultado = "";
184       while (true) {
185         resultado += lexer.lex();
186       }
187     } catch (IOException ex) {
188       Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
189     }
190   }
191 }
```



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



Splash.java

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

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

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

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



MANUAL DE PRÁCTICAS



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

```
77 private void ProgressBarInicado() {
78     String textoCompleto = "Program-Arte      ";
79     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
80     jLabel1.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
81     jLabel1.setFont(new java.awt.Font("Comic Sans MS", java.awt.Font.PLAIN, 32));
82     Color colorInicio = Color.decode("#F0E68C");
83     Color colorMedio = Color.decode("#FFD700");
84     Color colorFinal = Color.decode("#DAAS20");
85     // Inicia el progress bar a partir de un Timer
86     Timer mTimer = new Timer(30, (ActionEvent e) -> {
87         int progress = pbCarga.getValue() + 1;
88         if (progress <= 100) {
89             pbCarga.setValue(progress); // Valor del progress bar
90             pbCarga.setBackground(Color.white);
91             pbCarga.setStringPainted(true);
92             pbCarga.setString("Loading... " + progress + "%");
93
94             // Cambiar el color del progress bar según el progreso usando los colores definidos
95             if (progress <= 30) {
96                 pbCarga.setForeground(colorInicio);
97             } else if (progress <= 60) {
98                 pbCarga.setForeground(colorMedio);
99             } else {
100                 pbCarga.setForeground(colorFinal);
101             }
102             // Determinar cuántas letras mostrar según el progreso
103             int numLetras = progress / 7; // Cada % se muestra una letra
104
105             String textoActual = textoCompleto.substring(0, numLetras);
106             // Actualizar el texto de la etiqueta con las letras correspondientes
107             jLabel1.setText(textoActual);
108         }
109     });
110     mTimer.start();
111 }
112 }
```



```
115     @SuppressWarnings("unchecked")
116     [Generated Code]
117
118     private void pnlBackgroundMousePressed(java.awt.event.MouseEvent evt) {
119         mPoint = evt.getPoint(); // Obtener el punto de partida del movimiento
120         getComponentAt(mPoint);
121     }
122
123     private void pnlBackgroundMouseDragged(java.awt.event.MouseEvent evt) {
124         // Obtener las posiciones actuales del JFrame en X, Y
125         int CurrentX = this.getLocation().x;
126         int CurrentY = this.getLocation().y;
127         // Obtener el movimiento del del JFrame en X, Y
128         int MoveX = (CurrentX + evt.getX()) - (CurrentX + mPoint.x);
129         int MoveY = (CurrentY + evt.getY()) - (CurrentY + mPoint.y);
130
131         // Calcular las nuevas posiciones
132         int x = CurrentX + MoveX;
133         int y = CurrentY + MoveY;
134
135         // Asignar las posiciones al JFrame para generar el movimiento
136         this.setLocation(x, y);
137     }
138
139     private void pnlCerrarMouseClicked(java.awt.event.MouseEvent evt) {
140         System.exit(0);
141     }
142
143
144     public static void main(String args[]) {
145         /* Set the Nimbus look and feel */
146         /* Look and feel setting code (optional)
147          */
148         //
149         //
150         //
151         //
152         //
153         //
154
155
156         public static void main(String args[]) {
157             /* Set the Nimbus look and feel */
158             /* Look and feel setting code (optional)
159             */
160             //
161             //
162             //
163             //
164             //
165             //
166
167             /* Create and display the form */
168             java.awt.EventQueue.invokeLater(new Runnable() {
169                 public void run() {
170                     new Splash().setVisible(true);
171                 }
172             });
173         }
174
175
176         // Variables declaration - do not modify
177         private javax.swing.JButton jButton1;
178         private javax.swing.JLabel jLabel1;
179         private javax.swing.JPanel jPanel1;
180         private javax.swing.JProgressBar pbCarga;
181         private javax.swing.JPanel pnlBackground;
182         private javax.swing.JPanel pnlCerrar;
183         // End of variables declaration
184     }
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291 }
```

```
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291 }
```



LoginForm.java

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

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

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



```
74 JLabel userLabel = new JLabel("Ingresa Usuario:");
75 userLabel.setBounds(50, 100, 120, 25);
76 userLabel.setFont(new Font("Arial", Font.PLAIN, 14));
77 rightPanel.add(userLabel);
78
79 JTextField usernameField = new JTextField();
80 usernameField.setBounds(50, 130, 200, 30);
81 rightPanel.add(usernameField);
82
83 JLabel passLabel = new JLabel("Ingresa Contraseña:");
84 passLabel.setBounds(50, 180, 150, 25);
85 passLabel.setFont(new Font("Arial", Font.PLAIN, 14));
86 rightPanel.add(passLabel);
87
88 JPasswordField passwordField = new JPasswordField();
89 passwordField.setBounds(50, 210, 200, 30);
90 rightPanel.add(passwordField);
91
92 JButton loginButton = new JButton("Entrar");
93 loginButton.setBounds(50, 270, 200, 40);
94 loginButton.setFocusPainted(false);
95 loginButton.setBackground(Color.decode("#33B5E5"));
96 loginButton.setForeground(Color.WHITE);
97 loginButton.setFont(new Font("Arial", Font.BOLD, 14));
98 loginButton.setBorder(BorderFactory.createLineBorder(Color.decode("#33B5E5"), 2));
99 loginButton.addActionListener(e -> {
100     String username = usernameField.getText();
101     String password = new String(passwordField.getPassword());
102
103     // Validar las credenciales
104     // Validar las credenciales
105     if (users.containsKey(username) && users.get(username).equals(password)) {
106         JOptionPane.showMessageDialog(null, "Inicio de sesión exitoso!");
107         // Crear e iniciar la ventana de bienvenida
108         FrmBienvenida ventanaBienvenida = new FrmBienvenida();
109         ventanaBienvenida.setVisible(true);
110     } else {
111         JOptionPane.showMessageDialog(null, "Usuario o contraseña incorrectos");
112     }
113 }
114 )
115 );
116
117 rightPanel.add(loginButton);
118
119 // Centrar la ventana
120 setLocationRelativeTo(null);
121
122 setVisible(true);
123
124
125 public static void main(String[] args) {
126     new LoginForm();
127 }
128
129 // Método para agregar un usuario a la lista
130 public static void addUser(String username, String password) {
131     users.put(username, password);
132 }
133 }
```

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

RegisterForm.java

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

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



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

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



FrmBienvenida.java

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

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



MANUAL DE PRÁCTICAS



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

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

Main.java

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

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

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



PANTALLAS RESULTANTES CON PRUEBAS DEL RECONOCIMIENTO DE LOS TOKENS

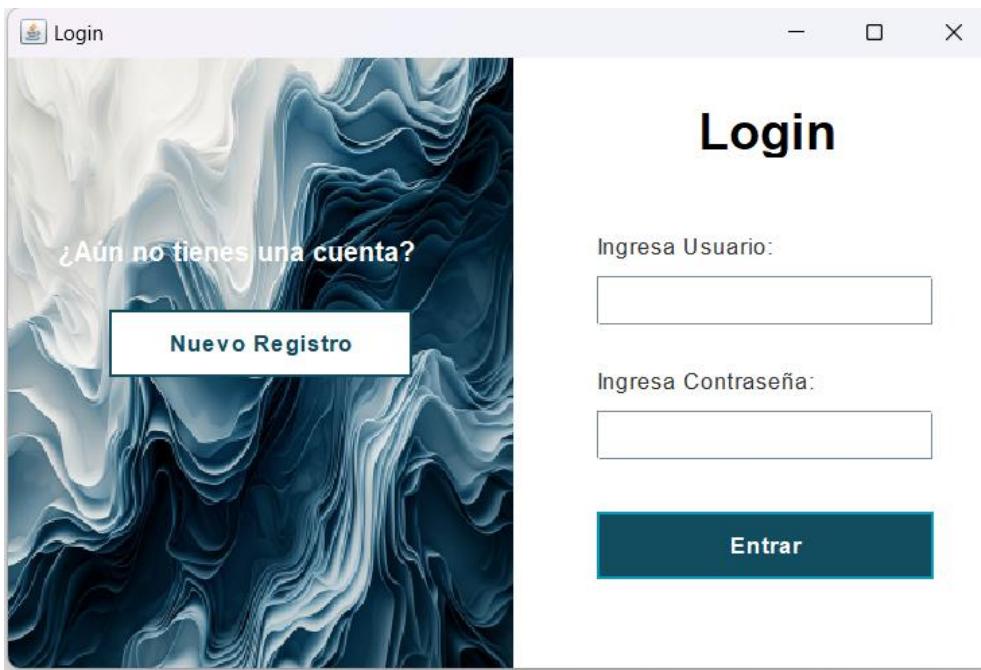
Pantallas de ejecución

Splash.java

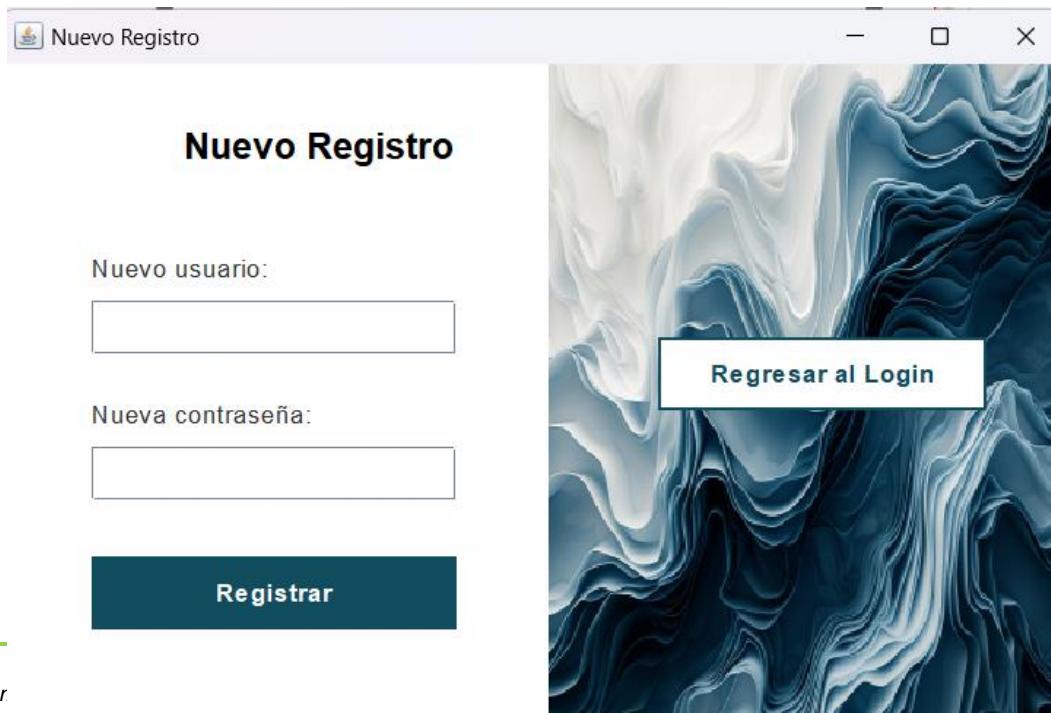




LoginForm.java



RegisterForm.java

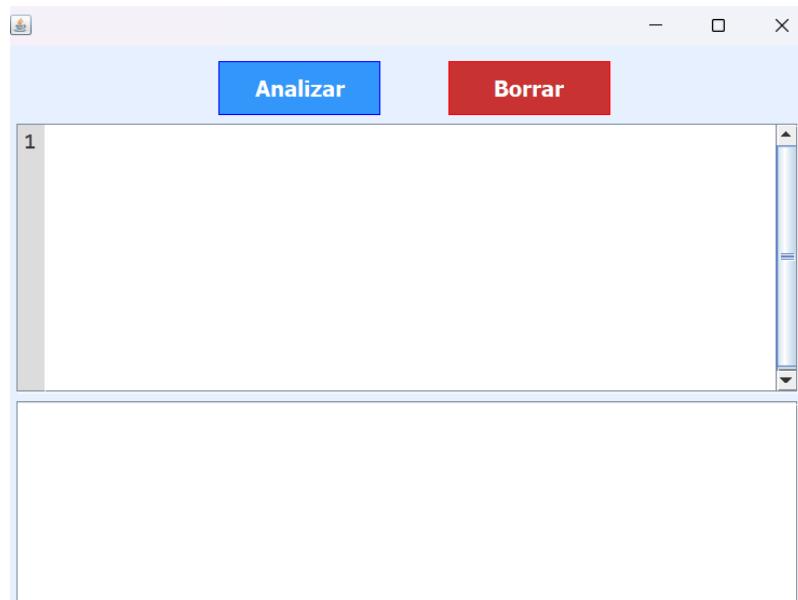




FrmBienvenida.java



FrmPrincipal.java





Ejemplo del análisis de un código:

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

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

ANALIZADOR SINTACTICO

....

Descripción del problema

El **anificador sintáctico** desempeña un papel esencial en el procesamiento de un lenguaje de programación al validar no solo la correcta formación de los elementos básicos, tarea realizada por el analizador léxico, sino también la coherencia estructural y lógica de las instrucciones según las reglas gramaticales definidas. Este proceso es fundamental para garantizar que el código sea interpretado o ejecutado correctamente, evitando errores durante la compilación o la interpretación.

El analizador sintáctico funciona tomando como entrada los tokens generados por el analizador léxico. Estos tokens representan las unidades mínimas del lenguaje, como palabras reservadas, identificadores, operadores y delimitadores, y suelen incluir su tipo y posición en el código para facilitar la localización de errores. Con esta entrada, el analizador organiza los tokens en estructuras jerárquicas conocidas como árboles de derivación o árboles sintácticos abstractos (AST). Estas estructuras reflejan cómo los tokens se agrupan para formar expresiones, sentencias y bloques de código más complejos. Utilizando una gramática formal predefinida, el analizador verifica que la secuencia de tokens cumpla con las reglas establecidas. Si, por ejemplo, una gramática exige que una función comience con un tipo de dato seguido de un identificador y unos paréntesis, el analizador detectará y reportará errores como la falta de paréntesis, el uso de un identificador inválido o la ausencia del tipo de dato.

Además, el analizador sintáctico es capaz de manejar errores de manera eficiente. Si encuentra una estructura incorrecta, no solo lo reporta, sino que intenta continuar el análisis para identificar otros problemas, un proceso conocido como recuperación de errores. Esto permite ofrecer una retroalimentación más completa al programador, ayudándolo a corregir múltiples errores en una sola iteración.

La salida del analizador sintáctico, el árbol sintáctico, se utiliza como base para etapas posteriores del procesamiento del lenguaje, como el análisis semántico, que valida aspectos relacionados con el significado del código, y la generación de código, que traduce estas estructuras a un formato ejecutable.

En el contexto de mi proyecto, la integración del analizador léxico y sintáctico es esencial para validar completamente el código escrito. Mientras que el analizador léxico asegura que los elementos básicos del lenguaje, como palabras reservadas y delimitadores, sean correctos, el analizador sintáctico valida que estas unidades formen estructuras coherentes según la lógica del lenguaje diseñado. Esto garantiza

que las instrucciones del código no solo sean válidas en términos de sus componentes básicos, sino que también sigan una estructura lógica y fluida, acorde a las expectativas del lenguaje.

¿Qué es un analizador sintáctico?

El **analizador sintáctico**, también conocido como parser, es una etapa crucial en el proceso de compilación o interpretación de lenguajes de programación. Su función principal es tomar los tokens generados por el analizador léxico y organizar estos elementos en una estructura jerárquica, típicamente un árbol de derivación o un árbol sintáctico abstracto (AST). Este proceso permite validar que las secuencias de tokens cumplan con las reglas gramaticales del lenguaje.

Funcionamiento del Analizador Sintáctico:

- Entrada de tokens:** Recibe los tokens generados por el analizador léxico, que son las unidades mínimas reconocibles del lenguaje.
- Reconocimiento de estructuras:** Utilizando una gramática formal (como gramáticas libres de contexto), el analizador verifica si los tokens siguen una secuencia válida. Esto se logra mediante algoritmos como el análisis descendente (top-down) o ascendente (bottom-up).
- Producción de salida:** Si los tokens forman una estructura válida, se genera un árbol sintáctico que representa la jerarquía de las construcciones del lenguaje. En caso contrario, se detectan y reportan errores sintácticos al usuario.

Importancia del Analizador Sintáctico:

- Detección de errores estructurales:** Identifica problemas como declaraciones incompletas, uso incorrecto de operadores o sintaxis no permitida.
- Preparación para etapas posteriores:** Proporciona una base estructurada para la generación de código intermedio o la interpretación directa del código fuente.
- Eficiencia y claridad:** Facilita el diseño de compiladores modulares al separar la validación de la estructura del código de su análisis léxico y semántico.

Relación con el Analizador Léxico:

Mientras el analizador léxico valida los elementos básicos del código (tokens), el analizador sintáctico se enfoca en la lógica y la organización jerárquica de estos elementos. Por ejemplo, en el código `ent principal() {}`, el analizador léxico identifica los tokens `ent`, `principal`, `(`, `)`, `{`, `y`}; el analizador sintáctico, por su parte, asegura que esta secuencia siga la gramática de declaración de funciones del lenguaje.

La integración de ambos analizadores garantiza un procesamiento robusto y estructurado del código fuente, minimizando errores y facilitando la correcta interpretación o compilación de programas en cualquier lenguaje.

Estructura del proyecto

- `LexerCup.flex`
- `LexerCup.java`
- `Sintaxis.cup`
- `Sintaxis.java`
- `sym.java`
- `Principal.java`
- `FrmPrincipla.java`
- `Ayuda.java`
- `Creditos.java`



Projects X

- Analizador
 - Source Packages
 - codigo
 - Ayuda.java
 - Creditos.java
 - FrmBienvenida.java
 - FrmPrincipal.java
 - ImageDecore.java
 - Lexer.flex
 - Lexer.java
 - LexerCup.flex
 - LexerCup.java
 - LoginForm.java
 - Main.java
 - Principal.java
 - RegisterForm.java
 - Sintax.cup
 - Sintax.java
 - Splash.java
 - Tokens.java
 - sym.java
 - icons
 - Test Packages
 - Libraries
 - java-cup-11a.jar
 - JFlex (1).jar
 - java_cup.jar
 - JDK 21 (Default)
 - Test Libraries

LexerCup.flex

El archivo LexerCup.flex es una parte fundamental en la creación del analizador léxico para mi proyecto. Este archivo define cómo se identifican y clasifican los elementos básicos de mi lenguaje de programación, conocidos como tokens. Su propósito principal es leer el código de entrada y dividirlo en unidades significativas como palabras reservadas, operadores, identificadores y literales, mientras ignora elementos irrelevantes como espacios en blanco o comentarios. Esto permite que el código ingresado sea limpio y esté listo para ser procesado en las etapas siguientes.

El funcionamiento del archivo se basa en patrones definidos mediante expresiones regulares, que describen cómo identificar cada tipo de token. Cuando una parte del código coincide con un patrón, el analizador léxico genera un token que encapsula información como el tipo, valor y posición del elemento en el texto. Por ejemplo, al detectar un operador de suma (+), se genera un token correspondiente al tipo sym.Sum. De esta manera, el analizador léxico organiza y valida los componentes básicos del lenguaje, asegurándose de que solo se consideren elementos válidos para el análisis posterior.

En mi proyecto, este analizador léxico es crucial, ya que actúa como el primer filtro para garantizar que el código escrito cumpla con las reglas básicas de formación. Al dividir el código en tokens estructurados y libres de elementos innecesarios, se mejora la precisión del análisis sintáctico y se asegura una base sólida para la interpretación o ejecución del lenguaje.



Código:

```
1 package codigo;
2 import java_cup.runtime.Symbol;
3 /**
4 %class LexerCup
5 %type java_cup.runtime.Symbol
6 %cup
7 %full
8 %line
9 %char
10 L=[a-zA-Z_]+
11 D=[0-9]+
12
13 espacio=[ ,\t,\r,\n]+
14 %}
15     private Symbol symbol(int type, Object value){
16         return new Symbol(type, yyline, yycolumn, value);
17     }
18     private Symbol symbol(int type){
19         return new Symbol(type, yyline, yycolumn);
20     }
21 }
22 /**
23 /* Espacios en blanco */
24 (espacio) {/*Ignore*/}
25
26 /* Comentarios */
27 (("//".*) ) {/*Ignore*/}
28
29
30 /* OPERADORES ARITMÉTICOS */
31
32 /* Operador Suma */
33 ( "+" ) {return new Symbol(sym.Suma, yychar, yyline, yytext());}
34
```

```
36 /* Operador Resta */
37 ( "-" ) {return new Symbol(sym.Resta, yychar, yyline, yytext());}
38
39 /* Operador Multiplicacion */
40 ( "*" ) {return new Symbol(sym.Multiplicacion, yychar, yyline, yytext());}
41
42 /* Operador Division */
43 ( "/" ) {return new Symbol(sym.Division, yychar, yyline, yytext());}
44
45 /* OPERADORES RELACIONALES */
46
47 /* Operador Mayor */
48 ( ">" ) {return new Symbol(sym.Mayor, yychar, yyline, yytext());}
49
50 /* Operador Menor */
51 ( "<" ) {return new Symbol(sym.Menor, yychar, yyline, yytext());}
52
53 /* Operador Mayor o Igual */
54 ( ">=" ) {return new Symbol(sym.MayorIgual, yychar, yyline, yytext());}
55
56 /* Operador Menor o Igual */
57 ( "<=" ) {return new Symbol(sym.MenorIgual, yychar, yyline, yytext());}
58
59 /* Operador Comparación */
60 ( "==" ) {return new Symbol(sym.Comparacion, yychar, yyline, yytext());}
61
62 /* Operador Distinto */
63 ( "!=" ) {return new Symbol(sym.Distinto, yychar, yyline, yytext());}
64
65 /* OPERADORES LÓGICOS */
66
67 /* Operador Y */
68 ( "&" ) {return new Symbol(sym.Y, yychar, yyline, yytext());}
69
70 /* Operador O */
71 ( "|" ) {return new Symbol(sym.O, yychar, yyline, yytext());}
```



```
73  /* Operador No */
74  ( "!" ) {return new Symbol(sym.No, yychar, yyline, yytext());}
75
76  /* OPERADORES */
77
78  /* Operador Módulo */
79  ( "%" ) {return new Symbol(sym.Modulo, yychar, yyline, yytext());}
80
81  /* Operador Potencia */
82  ( "\^" ) {return new Symbol(sym.Potencia, yychar, yyline, yytext());}
83
84  /* Función Raíz */
85  ( "raiz" ) {return new Symbol(sym.Raiz, yychar, yyline, yytext());}
86
87  /* Punto */
88  ( "\\" ) {return new Symbol(sym.Punto, yychar, yyline, yytext());}
89
90  /* Operador Asignación */
91  ( ">=" ) {return new Symbol(sym.Asignacion, yychar, yyline, yytext());}
92
93  /* Operador Igual */
94  ( "=" ) {return new Symbol(sym.Igual, yychar, yyline, yytext());}
95
96  /* Operador Concatenación */
97  ( "con" ) {return new Symbol(sym.Concatenacion, yychar, yyline, yytext());}
98
99  /* SIGNOS */
100
101 /* Operador Positivo */
102 ( "pos" ) {return new Symbol(sym.Positivo, yychar, yyline, yytext());}
103
104 /* Operador Negativo */
105 ( "neg" ) {return new Symbol(sym.Negativo, yychar, yyline, yytext());}
```

```
109 /* Incremento */
110 ( "+" ) {return new Symbol(sym.Incremento, yychar, yyline, yytext());}
111
112 /* Decremento */
113 ( "--" ) {return new Symbol(sym.Decremento, yychar, yyline, yytext());}
114
115 /* CONSTANTES MATEMÁTICAS */
116
117 /* Constante PI */
118 ( "PI" ) {return new Symbol(sym.Pi, yychar, yyline, yytext());}
119
120 /* Constante Euler */
121 ( "E" ) {return new Symbol(sym.Euler, yychar, yyline, yytext());}
122
123 /* DELIMITADORES */
124
125 /* Paréntesis Apertura */
126 ( "(" ) {return new Symbol(sym.ParentesisApertura, yychar, yyline, yytext());}
127
128 /* Paréntesis Cierre */
129 ( ")" ) {return new Symbol(sym.ParentesisCierre, yychar, yyline, yytext());}
130
131 /* Corchete Apertura */
132 ( "[" ) {return new Symbol(sym.CorcheteApertura, yychar, yyline, yytext());}
133
134 /* Corchete Cierre */
135 ( "]" ) {return new Symbol(sym.CorcheteCierre, yychar, yyline, yytext());}
136
137 /* Llave Apertura */
138 ( "(" ) {return new Symbol(sym.LlaveApertura, yychar, yyline, yytext());}
139
140 /* Llave Cierre */
141 ( ")" ) {return new Symbol(sym.LlaveCierre, yychar, yyline, yytext());}
```



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

```
179 /* Valor Booleano Falso */  
180 ( "f" ) {return new Symbol(sym.Falso, yychar, yyline, yytext());}  
181  
182 /* Tipo de Dato Cadena */  
183 ( "cadena" ) {return new Symbol(sym.Cadena, yychar, yyline, yytext());}  
184  
185 /* ESTRUCTURAS DE CONTROL */  
186  
187 /* Condicional Si */  
188 ( "si" ) {return new Symbol(sym.Si, yychar, yyline, yytext());}  
189  
190 /* Condicional SiNo */  
191 ( "siNo" ) {return new Symbol(sym.SiNo, yychar, yyline, yytext());}  
192  
193 /* Condicional SiNoSi */  
194 ( "siNoSi" ) {return new Symbol(sym.SiNoSi, yychar, yyline, yytext());}  
195  
196 /* Ciclo Repite */  
197 ( "repite" ) {return new Symbol(sym.CicloRepite, yychar, yyline, yytext());}  
198  
199 /* Ciclo Mientras */  
200 ( "mientras" ) {return new Symbol(sym.Mientras, yychar, yyline, yytext());}  
201  
202 /* Palabra Clave Hacer */  
203 ( "hacer" ) {return new Symbol(sym.Hacer, yychar, yyline, yytext());}  
204  
205 /* Seleccionador */  
206 ( "seleccionar" ) {return new Symbol(sym.Seleccionador, yychar, yyline, yytext());}  
207  
208 /* Caso */  
209 ( "caso" ) {return new Symbol(sym.Caso, yychar, yyline, yytext());}  
210  
211 /* Predeterminado */  
212 ( "predeterminado" ) {return new Symbol(sym.Predeterminado, yychar, yyline, yytext());}  
213
```



```
214 /* Detener */
215 ( "detener" ) {return new Symbol(sym.Detener, yychar, yyline, yytext());}
216
217 /* DECLARACIONES */
218
219 /* Clase */
220 ( "Clase" ) {return new Symbol(sym.Clase, yychar, yyline, yytext());}
221
222 /* Arreglo */
223 ( "arreglo" ) {return new Symbol(sym.Arreglo, yychar, yyline, yytext());}
224
225
226 /* Función */
227 ( "fun" ) {return new Symbol(sym.Funcion, yychar, yyline, yytext());}
228
229 /* Procedimiento */
230 ( "proc" ) {return new Symbol(sym.Procedimiento, yychar, yyline, yytext());}
231
232 /* Principal */
233 ( "principal" ) {return new Symbol(sym.Principal, yychar, yyline, yytext());}
234
235 /* Biblioteca */
236 ( "biblioteca" ) {return new Symbol(sym.Biblioteca, yychar, yyline, yytext());}
237
238 /* ENTRADA Y SALIDA DE DATOS */
239
240 /* Imprimir */
241 ( "imprime" ) {return new Symbol(sym.Imprimir, yychar, yyline, yytext());}
242
243 /* Leer */
244 ( "leer" ) {return new Symbol(sym.Leer, yychar, yyline, yytext());}
245
246 /* Importar */
247 ( "importa" ) {return new Symbol(sym.Importar, yychar, yyline, yytext());}
248
249 /* Retornar */
250 ( "retorna" ) {return new Symbol(sym.Retornar, yychar, yyline, yytext());}
```

```
252 /* Metodo*/
253 ( "metodo" ) {return new Symbol(sym.Metodo, yychar, yyline, yytext());}
254
255 /* Nulo */
256 ( "nulo" ) {return new Symbol(sym.Nulo, yychar, yyline, yytext());}
257
258 /* COMPONENTES DE UNA CADENA */
259
260 /* Caracteres Especiales */
261 ( ":" | "$" ) {return new Symbol(sym.CaracteresEspeciales, yychar, yyline, yytext());}
262
263 /* Fin de Linea */
264 ( ";" ) {return new Symbol(sym.P_coma, yychar, yyline, yytext());}
265
266 /* Marcador de inicio de algoritmo */
267 ( "main" ) {return new Symbol(sym.Main, yychar, yyline, yytext());}
268
269 /* Constantes */
270 [A-Z_]+ {return new Symbol(sym.Constante, yychar, yyline, yytext());}
271
272 /* Nombres de clases */
273 [A-Z][a-zA-Z0-9_]* {return new Symbol(sym.NombreClase, yychar, yyline, yytext());}
274
275
276 /* Identificador */
277 ({L}|{D})* {return new Symbol(sym.Identificador, yychar, yyline, yytext());}
278
279 /* Numero */
280 ("-{D}+")|{D}+ {return new Symbol(sym.Numero, yychar, yyline, yytext());}
281
282 /* Clase */
283 ( "Clase" ) {return new Symbol(sym.Clase, yychar, yyline, yytext());}
284
285 /* Texto entre comillas dobles */
286 ("[^\"\\\\\\\\]*\\\\\\\\.["^\"\\\\\\\\]*")" { return new Symbol(sym.Texto, yychar, yyline, yytext());}
```



MANUAL DE PRÁCTICAS



```
288 /* Texto entre comillas simples */
289 /*'[^\\'\\\\\\\\]*\\\\\\\\.[^\\'\\\\\\\\]*')\\' { return new Symbol(sym.Texto, yychar, yyline, yytext());}
290
291 /* Constante: solo letras mayúsculas y guion bajo */
292 /*([A-Z_]+) {lexeme=yytext(); return Constante; }*/
293
294 /* Clase: comienza con mayúscula seguida de minúsculas o guion bajo */
295 /*[A-Z][a-z_]+ {lexeme=yytext(); return Clase; }*/
296
297 /* Variable: empieza con minúsculas y puede contener mayúsculas, minúsculas, guiones bajos y dígitos */
298 /*[a-z_][a-zA-Z_0-9]* {lexeme=yytext(); return Variable; }*/
299
300
301 /* Error de análisis */
302 . {return new Symbol(sym.ERROR, yychar, yyline, yytext());}
```

LexerCup.java

El archivo LexerCup.java es un componente generado automáticamente que implementa el analizador léxico definido en LexerCup.flex. Este código es crucial, ya que transforma las especificaciones declaradas en el archivo de configuración en una implementación funcional que el programa utiliza para analizar el código de entrada. Es decir, este archivo contiene el código necesario para leer los caracteres de la entrada y clasificarlos como tokens según las reglas previamente definidas.

El funcionamiento del archivo se basa en el uso de un autómata finito, que se genera automáticamente a partir de las expresiones regulares especificadas. Este autómata permite al analizador identificar patrones en el texto de entrada y determinar a qué tipo de token corresponden. Por ejemplo, si encuentra un +, el autómata reconoce este carácter como un operador de suma y genera un token asociado.

El archivo también incluye estructuras y métodos necesarios para gestionar la entrada y manejar el flujo del análisis. Por ejemplo, define constantes como YYEOF para denotar el final del archivo, buffers de entrada para manejar los datos y tablas de transición que dictan cómo moverse entre estados en el autómata en función de los caracteres leídos.

En mi proyecto, este archivo es fundamental porque proporciona la implementación que el analizador léxico necesita para transformar el texto en una serie de tokens estructurados. Estos tokens son posteriormente utilizados por el analizador sintáctico para validar la gramática del código. En conjunto, LexerCup.java asegura que las instrucciones sean identificadas correctamente y que el lenguaje funcione de manera precisa y eficiente, proporcionando una base sólida para el análisis y la ejecución del código.



Código:

```
1  /* The following code was generated by JFlex 1.4.3 on 05/01/25, 02:00 */
2
3  package codigo;
4  import java_cup.runtime.Symbol;
5
6  /**
7   * This class is a scanner generated by
8   * <a href="http://www.jflex.de/">JFlex</a> 1.4.3
9   * on 05/01/25, 02:00 from the specification file
10  * <tt>src/codigo/LexerCup.flex</tt>
11  */
12  class LexerCup implements java_cup.runtime.Scanner {
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_BUFSIZE = 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 char [] ZZ_CMAP = {
37      0, 0, 0, 0, 0, 0, 0, 0, 3, 5, 0, 0, 0, 3, 0, 0,
38      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
39      3, 12, 39, 41, 54, 15, 13, 40, 33, 34, 8, 6, 3, 7, 22, 4,
40      2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
41      0, 57, 57, 51, 57, 32, 57, 57, 31, 57, 57, 57, 57, 46, 57,
42      30, 57, 57, 47, 57, 57, 57, 57, 57, 35, 21, 36, 16, 56,
43      0, 18, 53, 23, 43, 28, 45, 29, 49, 19, 1, 1, 50, 46, 25, 24,
44      26, 1, 17, 27, 42, 52, 44, 1, 1, 20, 37, 14, 38, 0, 0,
45      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
46      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
47      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
48      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
49      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
50      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
51      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
52      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
53  };
54
55  /**
56   * Translates DFA states to action switch labels.
57   */
58  private static final int [] ZZ_ACTION = zzUnpackAction();
59
60  private static final String ZZ_ACTION_PACKED_0 =
61      "\\"1\0\1\\1\\2\1\3\1\4\\1\5\1\6\1\7"+
62      "\\"1\10\1\11\1\12\1\13\1\4\1\5\1\16\1\17"+
63      "\\"1\20\1\3\2\1\1\5\1\2\1\21\1\22\1\23\1\24"+
64      "\\"1\25\1\26\1\27\1\30\1\31\1\32\1\33\1\2"+
65      "\\"1\34\1\35\1\32\1\21\1\21\1\36\1\37\1\21"+
66      "\\"1\4\1\40\1\41\1\42\1\43\1\44\1\45\1\46"+
67      "\\"1\47\1\50\1\51\1\52\1\53\1\54\1\55"+
68      "\\"2\2\1\55\1\56\1\30\1\57\1\20\1\7\1\21\1\55"+
69      "\\"10\2\1\60\1\61\1\2\1\62\1\2\1\63\1\3\0"+
70      "\\"1\64\1\2\1\65\1\2\1\55\1\2\1\66\1\2\1\52"+
71      "\\"1\67\1\2\1\70\1\2\1\71\1\2\1\72\1\2\1\52"
```



```
72 "11\3\1\2\1\73\3\2\1\74\1\55\16\2\1\75"+  
73 "11\76\1\2\1\77\4\2\1\100\2\2\1\101\3\2"+  
74 "11\102\1\2\1\103\1\104\1\105\1\106\3\2\1\107"+  
75 "15\2\1\110\1\2\1\111\5\2\1\112\1\2\1\113"+  
76 "12\2\1\114";  
77  
78     private static int [] zzUnpackAction() {  
79         int [] result = new int[202];  
80         int offset = 0;  
81         offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);  
82         return result;  
83     }  
84  
85     private static int zzUnpackAction(String packed, int offset, int [] result) {  
86         int i = 0; /* index in packed string */  
87         int j = offset; /* index in unpacked array */  
88         int l = packed.length();  
89         while (i < l) {  
90             int count = packed.charAt(i++);  
91             int value = packed.charAt(i++);  
92             do result[j++] = value; while (--count > 0);  
93         }  
94         return j;  
95     }  
96  
97  
98     /**  
99      * Translates a state to a row index in the transition table  
100     */  
101    private static final int [] ZZ_ROWMAP = zzUnpackRowMap();  
102  
103    private static final String ZZ_ROWMAP_PACKED_0 =  
104        "\u0\u072\0\u164\0\256\0\350\0\u0122\0\015c\0\0196"+  
105        "\u0\u01d0\0\020a\0\u0244\0\u027e\0\u02b8\0\072\0\072\0\072"+  
106        "\u0\72\0\u02f2\0\032c\0\u0366\0\u03a0\0\u03da\0\u0414\0\u044e"+  
107        "\u0\u0488\0\u04c2\0\u04fc\0\u0536\0\u0536\0\u0570\0\072\0\072"+
```

```
108 "10\72\0\72\0\72\0\05aa\0\u05e4\0\72\0\u061e\0\164"+  
109 "10\0\0659\0\u0692\0\u06cc\0\u0706\0\u0740\0\u077a\0\072\0\072"+  
110 "10\0\07b4\0\u07ee\0\072\0\072\0\072\0\072\0\072"+  
111 "10\72\0\072\0\72\0\072\0\072\0\072\0\072\0\072"+  
112 "10\0\08d6\0\072\0\u0910\0\u094a\0\u0984\0\u09be\0\u09f8\0\u0a32"+  
113 "10\0\0a6e\0\u0aa6\0\u0ae0\0\u0b1a\0\u0536\0\u0b54\0\u05aa\0\u0b8e"+  
114 "10\72\0\u00e4\0\u0bc9\0\u0c02\0\u0c3c\0\u0c76\0\u0cb0\0\u0cea"+  
115 "10\0\0d24\0\u0d5e\0\u0dd98\0\u0dd2\0\u0e0c\0\u0e46\0\u0e80\0\u0eba"+  
116 "10\0\0ef4\0\u0fe2\0\u0ff6\0\164\0\164\0\u0fa2\0\u0fdcc\0\u0106"+  
117 "10\0\1050\0\164\0\u108a\0\u10c4\0\164\0\u10fe\0\u1138\0\u1172"+  
118 "10\0\164\0\u1aac\0\164\0\u1le6\0\u1220\0\u125a\0\u1294\0\u12ce"+  
119 "10\0\1308\0\u1342\0\164\0\u137c\0\u13b6\0\u13f0\0\u142a\0\u1464"+  
120 "10\0\164\0\u149e\0\164\0\u14d8\0\164\0\u1512\0\u154c\0\u1586"+  
121 "10\0\72\0\u15c0\0\164\0\u15fa\0\u1634\0\u166e\0\164\0\u16a8"+  
122 "10\0\16e2\0\u171c\0\u1756\0\u1790\0\u17ea\0\u1804\0\u183e\0\u1878"+  
123 "10\0\18b2\0\u18ec\0\u1926\0\u1960\0\u199a\0\u19d4\0\u164\0\u0b1a"+  
124 "10\0\1a0e\0\164\0\u1a48\0\u1a82\0\u1abc\0\u1afe\0\164\0\u1b30"+  
125 "10\0\ulb6a\0\164\0\u1ba4\0\u1bde\0\u1c18\0\164\0\u1c52\0\164"+  
126 "10\0\164\0\164\0\u1c8c\0\u1cc6\0\u1d00\0\164\0\u1d3a"+  
127 "10\0\1d74\0\u1dae\0\u1de8\0\u1e22\0\164\0\u1e5c\0\164\0\u1e96"+  
128 "10\0\ulf0a\0\u1f44\0\u1f7e\0\164\0\u1ffb8\0\164\0\u1ff2"+  
129 "10\0\202c\0\164";  
130  
131     private static int [] zzUnpackRowMap() {  
132         int [] result = new int[202];  
133         int offset = 0;  
134         offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);  
135         return result;  
136     }  
137  
138     private static int zzUnpackRowMap(String packed, int offset, int [] result) {  
139         int i = 0; /* index in packed string */  
140         int j = offset; /* index in unpacked array */  
141         int l = packed.length();  
142         while (i < l) {  
143             int high = packed.charAt(i++) << 16;
```



MANUAL DE PRÁCTICAS





MANUAL DE PRÁCTICAS



218 "2\0\2\3\1\0\2\3\16\0\4\3\2\0\12\3"+
219 "\1\0\1\15\13\2\0\2\3\1\0\2\114\16\0"+
220 "\4\114\2\0\12\114\11\0\14\114\2\0\2\114\2\0"+
221 "\1\156\114\0\1\157\71\0\1\160\45\0\2\3\16\0"+
222 "\4\3\2\0\1\161\11\3\1\0\1\162\13\3\2\0"+
223 "\2\3\1\0\2\3\16\0\4\3\2\0\2\3\1\163"+
224 "\7\3\1\0\14\3\2\0\2\3\1\0\2\3\1\6\0"+
225 "\2\3\1\164\1\3\2\0\12\3\1\1\0\14\3\2\0"+
226 "\2\3\1\0\2\3\16\0\4\3\2\0\1\3\1\165"+
227 "\4\3\1\0\14\3\2\0\2\3\1\0\2\3\1\6\0"+
228 "\4\3\2\0\12\3\1\1\0\1\166\13\3\2\0\2\3"+
229 "\1\0\2\3\16\0\4\3\2\0\1\167\11\3\1\1\0"+
230 "\1\4\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+
231 "\5\3\1\170\4\3\1\1\0\14\3\2\0\2\3\1\0"+
232 "\2\114\16\0\1\114\171\2\114\2\0\12\114\11\0"+
233 "\1\4\114\2\0\2\114\1\0\2\3\16\0\4\3\2\0"+
234 "\12\3\1\1\0\13\3\1\172\2\0\2\3\1\0\2\3"+
235 "\16\0\3\3\1\173\2\0\12\3\1\0\14\3\2\0"+
236 "\2\3\1\0\2\3\16\0\2\3\1\174\1\3\2\0"+
237 "\12\3\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0"+
238 "\4\3\2\0\1\3\1\175\10\3\1\1\0\14\3\2\0"+
239 "\2\3\1\0\2\3\16\0\4\3\2\0\5\3\1\176"+
240 "\4\3\1\1\0\14\3\2\0\2\3\1\0\2\3\1\6\0"+
241 "\1\177\3\3\2\0\1\3\1\1\0\100\10\3\1\1\0\14\3"+
242 "\2\0\2\3\1\0\2\3\16\0\4\3\2\0\1\3\1"+
243 "\1\201\10\3\1\1\0\14\3\2\0\2\3\1\0\2\3"+
244 "\16\0\4\3\2\0\5\3\1\1\0\102\4\3\1\1\0\14\3"+
245 "\2\0\2\3\1\0\2\3\16\0\4\3\2\0\0\1\3\1"+
246 "\1\203\10\3\1\1\0\14\3\2\0\1\2\3\1\0\2\3"+
247 "\16\0\4\3\2\0\2\3\1\1\0\104\7\3\1\1\0\14\3"+
248 "\2\0\2\3\1\0\2\3\16\0\4\3\2\0\1\205"+
249 "\11\3\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0"+
250 "\4\3\2\0\12\3\1\1\0\1\1\0\106\12\3\2\0"+
251 "\2\3\1\0\2\3\16\0\4\3\2\0\1\3\1\207"+
252 "\10\3\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0"+
253 "\4\3\2\0\5\3\1\1\0\10\4\3\2\0\1\4\3\2\0"+
254 "\2\3\2\0\1\156\3\7\0\1\211\27\0\5\117\1\0"+

255 "\6\117\5\122\1\0\6\122\1\0\2\3\16\0\4\3"+
256 "\2\0\5\3\1\212\4\3\1\1\0\14\3\2\0\2\3"+
257 "\1\0\2\3\16\0\4\3\2\0\2\3\1\213\7\3"+
258 "\11\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+
259 "\2\0\2\3\1\214\7\3\1\1\0\14\3\2\0\2\3"+
260 "\1\0\2\3\16\0\4\3\2\0\1\3\1\215\10\3"+
261 "\11\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+
262 "\2\0\5\3\1\216\4\3\1\1\0\14\3\2\0\2\3"+
263 "\1\0\2\3\16\0\1\217\3\3\2\0\1\2\3\1\1\0"+
264 "\14\3\2\0\2\3\1\0\2\114\16\0\4\114\2\0"+
265 "\4\114\1\220\5\114\11\0\14\114\2\0\2\114\1\0"+
266 "\2\3\16\0\4\3\2\0\1\2\3\1\1\0\10\3\1\221"+
267 "\3\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+
268 "\12\3\1\1\0\1\222\13\3\2\0\2\3\1\0\2\3"+
269 "\16\0\1\223\3\3\2\0\12\3\1\0\14\3\2\0"+
270 "\2\3\1\0\2\3\16\0\4\3\2\0\1\3\1\224"+
271 "\3\3\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0"+
272 "\2\3\1\225\1\3\2\0\12\3\1\1\0\14\3\2\0"+
273 "\2\3\1\0\2\3\16\0\1\226\3\3\2\0\12\3"+
274 "\11\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+
275 "\2\0\2\3\1\227\7\3\1\1\0\14\3\2\0\1\3"+
276 "\1\0\2\3\16\0\4\3\2\0\1\230\11\3\1\1\0"+
277 "\14\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+
278 "\5\3\1\231\4\3\1\1\0\14\3\2\0\2\3\1\0"+
279 "\2\3\16\0\4\3\2\0\1\2\3\1\1\0\5\3\1\232"+
280 "\6\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+
281 "\1\233\1\1\3\1\1\0\14\3\2\0\2\3\1\0\2\3"+
282 "\16\0\1\3\2\0\2\3\1\1\0\1234\7\3\1\1\0\14\3"+
283 "\2\0\2\3\1\0\2\3\16\0\4\3\2\0\1\2\3"+
284 "\1\0\1\235\1\3\3\2\0\2\3\1\0\2\3\16\0"+
285 "\1\3\2\0\12\3\1\1\0\1\236\12\3\2\0"+
286 "\2\3\1\0\2\3\16\0\1\237\3\3\2\0\12\3"+
287 "\1\0\14\3\2\0\2\3\1\0\2\114\16\0\4\114"+
288 "\2\0\5\114\1\240\4\114\11\0\14\114\2\0\2\114"+
289 "\1\0\2\3\16\0\2\3\1\241\1\3\2\0\12\3"+
290 "\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+



```
291     "\2\0\5\3\1\242\4\3\11\0\14\3\2\0\2\3"+  
292     "\1\0\2\3\16\0\4\3\2\0\2\3\1\2\43\7\3"+  
293     "\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
294     "\2\0\12\3\11\0\10\3\1\244\3\3\2\0\2\3"+  
295     "\1\0\2\3\16\0\4\3\2\0\2\3\1\0\6\3"+  
296     "\1\245\5\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
297     "\2\0\12\3\11\0\1\246\3\3\2\0\2\3\1\0"+  
298     "\2\3\16\0\1\3\1\2\47\2\3\2\0\1\2\3\1\1\0"+  
299     "\1\4\3\2\0\2\3\1\0\2\3\16\0\2\3\1\250"+  
300     "\1\3\2\0\12\3\11\0\14\3\2\0\2\3\1\0"+  
301     "\2\3\16\0\4\3\2\0\12\3\1\1\0\1\251\1\3"+  
302     "\2\0\2\3\1\0\2\3\16\0\2\3\1\252\1\3"+  
303     "\2\0\12\3\11\0\14\3\2\0\2\3\1\0\2\3"+  
304     "\1\6\0\1\3\2\0\1\253\1\1\3\1\0\14\3\2\0"+  
305     "\2\3\1\0\2\3\16\0\4\3\2\0\5\3\1\254"+  
306     "\1\3\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
307     "\1\255\3\3\2\0\12\3\1\0\14\3\2\0\2\3"+  
308     "\1\0\2\3\16\0\4\3\2\0\1\3\1\256\1\3"+  
309     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
310     "\2\0\1\3\1\2\57\10\3\11\0\14\3\2\0\2\3"+  
311     "\1\0\2\3\16\0\1\3\1\260\2\3\2\0\12\3"+  
312     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
313     "\2\0\1\3\1\2\61\10\3\11\0\14\3\2\0\2\3"+  
314     "\1\0\2\3\16\0\4\3\2\0\5\3\1\262\4\3"+  
315     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\1\3"+  
316     "\1\263\2\3\2\0\12\3\11\0\14\3\2\0\2\3"+  
317     "\1\0\2\3\16\0\4\3\2\0\3\1\264\6\3"+  
318     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
319     "\2\0\5\3\1\2\65\4\3\11\0\14\3\2\0\2\3"+  
320     "\1\0\2\3\16\0\2\3\1\266\1\3\2\0\12\3"+  
321     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\1\267"+  
322     "\1\3\2\0\12\3\11\0\14\3\2\0\2\3\1\0"+  
323     "\2\3\16\0\1\3\1\2\70\2\3\2\0\1\12\3\1\0"+  
324     "\1\4\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+  
325     "\1\2\3\1\1\0\1\271\13\3\2\0\2\3\1\0\2\3"+  
326     "\1\6\0\1\3\1\272\2\3\2\0\12\3\11\0\14\3"+  
327     "\2\0\2\3\1\0\2\3\16\0\1\273\3\3\2\0"+  
328     "\1\2\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
329     "\1\3\2\0\1\3\1\274\10\3\11\0\14\3\2\0"+  
330     "\2\3\1\0\2\3\16\0\4\3\2\0\4\3\1\275"+  
331     "\1\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
332     "\1\3\2\0\5\3\1\276\4\3\11\0\14\3\2\0\4\3"+  
333     "\2\3\1\0\2\3\16\0\4\3\2\0\12\3\1\1\0"+  
334     "\10\3\1\277\3\2\0\2\3\1\0\2\3\16\0"+  
335     "\1\3\2\0\12\3\11\0\6\3\1\300\5\3\2\0"+  
336     "\2\3\1\0\2\3\16\0\4\3\2\0\2\3\1\301"+  
337     "\1\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
338     "\1\3\2\0\1\302\11\3\11\0\14\3\2\0\2\3"+  
339     "\1\0\2\3\16\0\2\3\1\303\1\3\2\0\12\3\1"+  
340     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\1\3"+  
341     "\1\304\2\3\1\0\12\3\11\0\14\3\2\0\2\3"+  
342     "\1\0\2\3\16\0\1\3\1\305\2\3\2\0\12\3\1"+  
343     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
344     "\2\0\2\3\1\306\7\3\11\0\14\3\2\0\2\3"+  
345     "\1\0\2\3\16\0\1\307\3\3\2\0\12\3\1\1\0"+  
346     "\1\4\3\2\0\2\3\1\0\2\3\16\0\1\3\1\310"+  
347     "\2\3\2\0\12\3\11\0\14\3\2\0\2\3\1\0"+  
348     "\2\3\16\0\4\3\2\0\12\3\11\0\1\3\1\311"+  
349     "\1\2\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+  
350     "\1\3\1\312\10\3\11\0\14\3\2\0\2\3";  
351  
352     private static int [] zzUnpackTrans() {  
353         int [] result = new int[8294];  
354         int offset = 0;  
355         offset = zzUnpackTrans(ZZ_TRANS_PACKED_0, offset, result);  
356         return result;  
357     }  
358  
359     private static int zzUnpackTrans(String packed, int offset, int [] result) {  
360         int i = 0; /* index in packed string */
```

```
325     "\12\3\11\0\1\271\13\3\2\0\2\3\1\0\2\3"+  
326     "\1\6\0\1\3\1\272\2\3\2\0\12\3\11\0\14\3"+  
327     "\2\0\2\3\1\0\2\3\16\0\1\273\3\3\2\0"+  
328     "\1\2\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
329     "\1\3\2\0\1\3\1\274\10\3\11\0\14\3\2\0"+  
330     "\2\3\1\0\2\3\16\0\4\3\2\0\4\3\1\275"+  
331     "\1\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
332     "\1\3\2\0\5\3\1\276\4\3\11\0\14\3\2\0\4\3"+  
333     "\2\3\1\0\2\3\16\0\4\3\2\0\12\3\1\1\0"+  
334     "\10\3\1\277\3\2\0\2\3\1\0\2\3\16\0"+  
335     "\1\3\2\0\12\3\11\0\6\3\1\300\5\3\2\0"+  
336     "\2\3\1\0\2\3\16\0\4\3\2\0\2\3\1\301"+  
337     "\1\3\11\0\14\3\2\0\2\3\1\0\2\3\16\0"+  
338     "\1\3\2\0\1\302\11\3\11\0\14\3\2\0\2\3"+  
339     "\1\0\2\3\16\0\2\3\1\303\1\3\2\0\12\3\1"+  
340     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\1\3"+  
341     "\1\304\2\3\1\0\12\3\11\0\14\3\2\0\2\3"+  
342     "\1\0\2\3\16\0\1\3\1\305\2\3\2\0\12\3\1"+  
343     "\1\1\0\14\3\2\0\2\3\1\0\2\3\16\0\4\3"+  
344     "\2\0\2\3\1\306\7\3\11\0\14\3\2\0\2\3"+  
345     "\1\0\2\3\16\0\1\307\3\3\2\0\12\3\1\1\0"+  
346     "\1\4\3\2\0\2\3\1\0\2\3\16\0\1\3\1\310"+  
347     "\2\3\2\0\12\3\11\0\14\3\2\0\2\3\1\0"+  
348     "\2\3\16\0\4\3\2\0\12\3\11\0\1\3\1\311"+  
349     "\1\2\3\2\0\2\3\1\0\2\3\16\0\4\3\2\0"+  
350     "\1\3\1\312\10\3\11\0\14\3\2\0\2\3";  
351  
352     private static int [] zzUnpackTrans() {  
353         int [] result = new int[8294];  
354         int offset = 0;  
355         offset = zzUnpackTrans(ZZ_TRANS_PACKED_0, offset, result);  
356         return result;  
357     }  
358  
359     private static int zzUnpackTrans(String packed, int offset, int [] result) {  
360         int i = 0; /* index in packed string */
```



```
361     int j = offset; /* index in unpacked array */
362     int l = packed.length();
363     while (i < l) {
364         int count = packed.charAt(i++);
365         int value = packed.charAt(i++);
366         value--;
367         do result[j++] = value; while (--count > 0);
368     }
369     return j;
370 }
371
372
373 /* error codes */
374 private static final int ZZ_UNKNOWN_ERROR = 0;
375 private static final int ZZ_NO_MATCH = 1;
376 private static final int ZZ_PUSHBACK_TOO_BIG = 2;
377
378 /* error messages for the codes above */
379 private static final String ZZ_ERROR_MSG[] = {
380     "Unknown internal scanner error",
381     "Error: could not match input",
382     "Error: pushback value was too large"
383 };
384
385 /**
386 * ZZ_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
387 */
388 private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();
389
390 private static final String ZZ_ATTRIBUTE_PACKED_0 =
391     "\\\\0\\1\\1\\13\\1\\4\\11\\15\\1\\5\\1\\2\\1\\1\\11" +
392     "\\1\\0\\1\\2\\1\\1\\2\\1\\13\\1\\1\\4\\1\\1\\1\\1\\13\\1\\1\\3\\0" +
393     "\\1\\11\\2\\0\\0\\32\\1\\3\\0\\30\\1\\1\\101\\1";
394
395 private static int [] zzUnpackAttribute() {
396     int [] result = new int[202];
397     int offset = 0;
```

```
398     offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
399     return result;
400 }
401
402 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
403     int i = 0; /* index in packed string */
404     int j = offset; /* index in unpacked array */
405     int l = packed.length();
406     while (i < l) {
407         int count = packed.charAt(i++);
408         int value = packed.charAt(i++);
409         do result[j++] = value; while (--count > 0);
410     }
411     return j;
412 }
413
414 /**
415 * the input device
416 */
417 private java.io.Reader zzReader;
418
419 /**
420 * the current state of the DFA
421 */
422 private int zzState;
423
424 /**
425 * this buffer contains the current text to be matched and is
426 * the source of the yytext() string
427 */
428 private char zzBuffer[] = new char[ZZ_BUFSIZE];
429
430 /**
431 * the textposition at the last accepting state
432 */
433 private int zzMarkedPos;
434
435 /**
436 * the current text position in the buffer
437 */
438 private int zzCurrentPos;
439
440 /**
441 * startRead marks the beginning of the yytext() string in the buffer
442 */
```



```
434     private int zzStartRead;
435
436     /** endRead marks the last character in the buffer, that has been read
437      * from input */
438     private int zzEndRead;
439
440     /** number of newlines encountered up to the start of the matched text */
441     private int yyline;
442
443     /** the number of characters up to the start of the matched text */
444     private int yychar;
445
446     /**
447      * the number of characters from the last newline up to the start of the
448      * matched text
449      */
450     private int yycolumn;
451
452     /**
453      * zzAtBOL == true <=> the scanner is currently at the beginning of a line
454      */
455     private boolean zzAtBOL = true;
456
457     /** zzAtEOF == true <=> the scanner is at the EOF */
458     private boolean zzAtEOF;
459
460     /** denotes if the user-EOF-code has already been executed */
461     private boolean zzEOFDone;
462
463     /* user code: */
464     private Symbol symbol(int type, Object value){
465         return new Symbol(type, yyline, yycolumn, value);
466     }
467     private Symbol symbol(int type){
468         return new Symbol(type, yyline, yycolumn);
469     }
```

```
472     /**
473      * Creates a new scanner
474      * There is also a java.io.InputStream version of this constructor.
475      *
476      * @param in the java.io.Reader to read input from.
477      */
478     LexerCup(java.io.Reader in) {
479         this.zzReader = in;
480     }
481
482     /**
483      * Creates a new scanner.
484      * There is also java.io.Reader version of this constructor.
485      *
486      * @param in the java.io.InputStream to read input from.
487      */
488     LexerCup(java.io.InputStream in) {
489         this(new java.io.InputStreamReader(in));
490     }
491
492     /**
493      * Refills the input buffer.
494      *
495      * @return      <code>false</code>, iff there was new input.
496      *
497      * @exception  java.io.IOException if any I/O-Error occurs
498      */
499     private boolean zzRefill() throws java.io.IOException {
500
501         /* first: make room (if you can) */
502         if (zzStartRead > 0) {
503             System.arraycopy(zzBuffer, zzStartRead,
504                             zzBuffer, 0,
505                             zzEndRead-zzStartRead);
```



```
508     /* translate stored positions */
509     zzEndRead-= zzStartRead;
510     zzCurrentPos-= zzStartRead;
511     zzMarkedPos-= zzStartRead;
512     zzStartRead = 0;
513 }
514
515 /* is the buffer big enough? */
516 if (zzCurrentPos >= zzBuffer.length) {
517     /* if not: blow it up */
518     char newBuffer[] = new char[zzCurrentPos*2];
519     System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
520     zzBuffer = newBuffer;
521 }
522
523 /* finally: fill the buffer with new input */
524 int numRead = zzReader.read(zzBuffer, zzEndRead,
525                             zzBuffer.length-zzEndRead);
526
527 if (numRead > 0) {
528     zzEndRead+= numRead;
529     return false;
530 }
531 // unlikely but not impossible: read 0 characters, but not at end of stream
532 if (numRead == 0) {
533     int c = zzReader.read();
534     if (c == -1) {
535         return true;
536     } else {
537         zzBuffer[zzEndRead++] = (char) c;
538         return false;
539     }
540 }
541 // numRead < 0
542 return true;
543 }
```

```
544 }
545
546 /**
547 * Closes the input stream.
548 */
549 public final void yyclose() throws java.io.IOException {
550     zzAtEOF = true; /* indicate end of file */
551     zzEndRead = zzStartRead; /* invalidate buffer */
552
553     if (zzReader != null)
554         zzReader.close();
555 }
556
557 /**
558 * Resets the scanner to read from a new input stream.
559 * Does not close the old reader.
560 *
561 * All internal variables are reset, the old input stream
562 * <b>cannot</b> be reused (internal buffer is discarded and lost).
563 * Lexical state is set to <tt>YY_INITIAL</tt>.
564 *
565 * @param reader the new input stream
566 */
567 public final void yyreset(java.io.Reader reader) {
568     zzReader = reader;
569     zzAtBOL = true;
570     zzAtEOF = false;
571     zzEOFDone = false;
572     zzEndRead = zzStartRead = 0;
573     zzCurrentPos = zzMarkedPos = 0;
574     yyline = yychar = yycolumn = 0;
575     zzLexicalState = YY_INITIAL;
576 }
577
578 }
```



```
581 /**
582 * Returns the current lexical state.
583 */
584 public final int yystate() {
585     return zzLexicalState;
586 }
587
588 /**
589 * Enters a new lexical state
590 *
591 * @param newState the new lexical state
592 */
593 public final void yybegin(int newState) {
594     zzLexicalState = newState;
595 }
596
597
598 /**
599 * Returns the text matched by the current regular expression.
600 */
601 public final String yytext() {
602     return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
603 }
604
605
606 /**
607 * Returns the character at position <tt>pos</tt> from the
608 * matched text.
609 *
610 * It is equivalent to yytext().charAt(pos), but faster
611 *
612 * @param pos the position of the character to fetch.
613 *           A value from 0 to yylength()-1.
614 *
615 * @return the character at position pos
616 */
```

```
618 public final char yycharat(int pos) {
619     return zzBuffer[zzStartRead+pos];
620 }
621
622
623 /**
624 * Returns the length of the matched text region.
625 */
626 public final int yylength() {
627     return zzMarkedPos-zzStartRead;
628 }
629
630
631 /**
632 * Reports an error that occurred while scanning.
633 *
634 * In a wellformed scanner (no or only correct usage of
635 * yypushback(int) and a match-all fallback rule) this method
636 * will only be called with things that "Can't Possibly Happen".
637 * If this method is called, something is seriously wrong
638 * (e.g. a JFlex bug producing a faulty scanner etc.).
639 *
640 * Usual syntax/scanner level error handling should be done
641 * in error fallback rules.
642 *
643 * @param errorCode the code of the error message to display
644 */
645 private void zzScanError(int errorCode) {
646     String message;
647     try {
648         message = ZZ_ERROR_MSG[errorCode];
649     }
650     catch (ArrayIndexOutOfBoundsException e) {
651         message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
652     }
653 }
654 throw new Error(message);
```



```
658  /**
659   * Pushes the specified amount of characters back into the input stream.
660   *
661   * They will be read again by then next call of the scanning method
662   *
663   * @param number the number of characters to be read again.
664   *                 This number must not be greater than yylength()!
665   */
666  public void yypushback(int number) {
667      if ( number > yylength() )
668          zzScanError(ZZ_PUSHBACK_2BIG);
669
670      zzMarkedPos -= number;
671  }
672
673
674  /**
675   * Contains user EOF-code, which will be executed exactly once,
676   * when the end of file is reached
677   */
678  private void zzDoEOF() throws java.io.IOException {
679      if (!zzEOFDone) {
680          zzEOFDone = true;
681          yyclose();
682      }
683  }
684
685
686  /**
687   * Resumes scanning until the next regular expression is matched,
688   * the end of input is encountered or an I/O-Error occurs.
689   *
690   * @return      the next token
691   * @exception   java.io.IOException if any I/O-Error occurs
692   */
693  public java_cup.runtime.Symbol next_token() throws java.io.IOException {
```

```
694      public java_cup.runtime.Symbol next_token() throws java.io.IOException {
695          int zzInput;
696          int zzAction;
697
698          // cached fields:
699          int zzCurrentPosL;
700          int zzMarkedPosL;
701          int zzEndReadL = zzEndRead;
702          char [] zzBufferL = zzBuffer;
703          char [] zzCMapL = ZZ_CMAP;
704
705          int [] zzTransL = ZZ_TRANS;
706          int [] zzRowMapL = ZZ_ROWMAP;
707          int [] zzAttrL = ZZ_ATTRIBUTE;
708
709          while (true) {
710              zzMarkedPosL = zzMarkedPos;
711
712              yychar+= zzMarkedPosL-zzStartRead;
713
714              boolean zzR = false;
715              for (zzCurrentPosL = zzStartRead; zzCurrentPosL < zzMarkedPosL;
716                  zzCurrentPosL++ | zzCurrentPosL++) {
717                  switch (zzBufferL[zzCurrentPosL]) {
718                      case '\u0008':
719                      case '\u0000':
720                      case '\u0085':
721                      case '\u2028':
722                      case '\u2029':
723                          yyline++;
724                          zzR = false;
725                          break;
726                      case '\r':
727                          yyline++;
728                          zzR = true;
729                          break;
```



```
729         case '\n':
730             if (zzR)
731                 zzR = false;
732             else {
733                 yyline++;
734             }
735             break;
736         default:
737             zzR = false;
738         }
739     }
740
741     if (zzR) {
742         // peek one character ahead if it is \n (if we have counted one line too much)
743         boolean zzPeek;
744         if (zzMarkedPosL < zzEndReadL)
745             zzPeek = zzBufferL[zzMarkedPosL] == '\n';
746         else if (zzAtEOF)
747             zzPeek = false;
748         else {
749             boolean eof = zzRefill();
750             zzEndReadL = zzEndRead;
751             zzMarkedPosL = zzMarkedPos;
752             zzBufferL = zzBuffer;
753             if (eof)
754                 zzPeek = false;
755             else
756                 zzPeek = zzBufferL[zzMarkedPosL] == '\n';
757         }
758         if (zzPeek) yyline--;
759     }
760     zzAction = -1;
761
762     zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
763
764     zzState = ZZ_LEXSTATE[zzLexicalState];
```

```
767     zzForAction: {
768         while (true) {
769
770             if (zzCurrentPosL < zzEndReadL)
771                 zzInput = zzBufferL[zzCurrentPosL++];
772             else if (zzAtEOF) {
773                 zzInput = YYEOF;
774                 break zzForAction;
775             }
776             else {
777                 // store back cached positions
778                 zzCurrentPos = zzCurrentPosL;
779                 zzMarkedPos = zzMarkedPosL;
780                 boolean eof = zzRefill();
781                 // get translated positions and possibly new buffer
782                 zzCurrentPosL = zzCurrentPos;
783                 zzMarkedPosL = zzMarkedPos;
784                 zzBufferL = zzBuffer;
785                 zzEndReadL = zzEndRead;
786                 if (eof) {
787                     zzInput = YYEOF;
788                     break zzForAction;
789                 }
790                 else {
791                     zzInput = zzBufferL[zzCurrentPosL++];
792                 }
793             }
794             int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
795             if (zzNext == -1) break zzForAction;
796             zzState = zzNext;
797
798             int zzAttributes = zzAttrL[zzState];
799             if ( (zzAttributes & 1) == 1 ) {
800                 zzAction = zzState;
801                 zzMarkedPosL = zzCurrentPosL;
802                 if ( (zzAttributes & 8) == 8 ) break zzForAction;
```



```
803     }
804
805   }
806 }
807
808 // store back cached position
809 zzMarkedPos = zzMarkedPosL;
810
811 switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
812   case 76:
813     { return new Symbol(sym.Predeterminado, yychar, yyline, yytext()); }
814   }
815   case 77: break;
816   case 58:
817     { return new Symbol(sym.SiNo, yychar, yyline, yytext()); }
818   }
819   case 78: break;
820   case 21:
821     { return new Symbol(sym.CorcheteApertura, yychar, yyline, yytext()); }
822   }
823   case 79: break;
824   case 38:
825     { return new Symbol(sym.InicioTexto, yychar, yyline, yytext()); }
826   }
827   case 80: break;
828   case 20:
829     { return new Symbol(sym.ParentesisCierre, yychar, yyline, yytext()); }
830   }
831   case 81: break;
832   case 44:
833     { return new Symbol(sym.Si, yychar, yyline, yytext()); }
834   }
835   case 82: break;
836   case 48:
837     { return new Symbol(sym.Concatenacion, yychar, yyline, yytext()); }
838 }
```

```
840   case 56:
841     { return new Symbol(sym.Nulo, yychar, yyline, yytext()); }
842   }
843   case 84: break;
844   case 75:
845     { return new Symbol(sym.Seleccionador, yychar, yyline, yytext()); }
846   }
847   case 85: break;
848   case 57:
849     { return new Symbol(sym.Procedimiento, yychar, yyline, yytext()); }
850   }
851   case 86: break;
852   case 55:
853     { return new Symbol(sym.Caso, yychar, yyline, yytext()); }
854   }
855   case 87: break;
856   case 14:
857     { return new Symbol(sym.O, yychar, yyline, yytext()); }
858   }
859   case 88: break;
860   case 59:
861     { return new Symbol(sym.Main, yychar, yyline, yytext()); }
862   }
863   case 89: break;
864   case 4:
865     { /*Ignore*/
866   }
867   case 90: break;
868   case 27:
869     { return new Symbol(sym.Comentario, yychar, yyline, yytext()); }
870   }
871   case 91: break;
872   case 34:
873     { return new Symbol(sym.Decremento, yychar, yyline, yytext()); }
874   }
875   case 92: break;
876 }
```



```
878     )
879     case 93: break;
880     case 47:
881     { return new Symbol(sym.Texto, yychar, yyline, yytext());
882     }
883     case 94: break;
884     case 64:
885     { return new Symbol(sym.Cadena, yychar, yyline, yytext());
886     }
887     case 95: break;
888     case 31:
889     { return new Symbol(sym.P_coma, yychar, yyline, yytext());
890     }
891     case 96: break;
892     case 46:
893     { return new Symbol(sym.Pi, yychar, yyline, yytext());
894     }
895     case 97: break;
896     case 53:
897     { return new Symbol(sym.Funcion, yychar, yyline, yytext());
898     }
899     case 98: break;
900     case 8:
901     { return new Symbol(sym.Multiplicacion, yychar, yyline, yytext());
902     }
903     case 99: break;
904     case 63:
905     { return new Symbol(sym.CicloRepite, yychar, yyline, yytext());
906     }
907     case 100: break;
908     case 1:
909     { return new Symbol(sym.ERROR, yychar, yyline, yytext());
910     }
911     case 101: break;
912     case 54:
913     { return new Symbol(sym.Raiz, yychar, yyline, yytext());
914     }
```

```
916     case 29:
917     { return new Symbol(sym.Falso, yychar, yyline, yytext());
918     }
919     case 103: break;
920     case 12:
921     { return new Symbol(sym.No, yychar, yyline, yytext());
922     }
923     case 104: break;
924     case 25:
925     { return new Symbol(sym.ComillaDoble, yychar, yyline, yytext());
926     }
927     case 105: break;
928     case 15:
929     { return new Symbol(sym.Modulo, yychar, yyline, yytext());
930     }
931     case 106: break;
932     case 32:
933     { return new Symbol(sym.InicioComentario, yychar, yyline, yytext());
934     }
935     case 107: break;
936     case 72:
937     { return new Symbol(sym.Mientras, yychar, yyline, yytext());
938     }
939     case 108: break;
940     case 11:
941     { return new Symbol(sym.Igual, yychar, yyline, yytext());
942     }
943     case 109: break;
944     case 66:
945     { return new Symbol(sym.Metodo, yychar, yyline, yytext());
946     }
947     case 110: break;
948     case 2:
949     { return new Symbol(sym.Identificador, yychar, yyline, yytext());
950     }
951     case 111: break;
952     case 17:
```



```
853     { return new Symbol(sym.Constante, yychar, yyline, yytext());
854 }
855 case 112: break;
856 case 26:
857     { return new Symbol(sym.ComillaSimple, yychar, yyline, yytext());
858 }
859 case 113: break;
860 case 49:
861     { return new Symbol(sym.Negativo, yychar, yyline, yytext());
862 }
863 case 114: break;
864 case 24:
865     { return new Symbol(sym.LlaveCierre, yychar, yyline, yytext());
866 }
867 case 115: break;
868 case 16:
869     { return new Symbol(sym.Potencia, yychar, yyline, yytext());
870 }
871 case 116: break;
872 case 52:
873     { return new Symbol(sym.Decimal, yychar, yyline, yytext());
874 }
875 case 117: break;
876 case 6:
877     { return new Symbol(sym.Sumas, yychar, yyline, yytext());
878 }
879 case 118: break;
880 case 70:
881     { return new Symbol(sym.Importar, yychar, yyline, yytext());
882 }
883 case 119: break;
884 case 10:
885     { return new Symbol(sym.Menor, yychar, yyline, yytext());
886 }
887 case 120: break;
888 case 41:
```

```
889     { return new Symbol(sym.Comparacion, yychar, yyline, yytext());
890 }
891 case 121: break;
892 case 60:
893     { return new Symbol(sym.Leer, yychar, yyline, yytext());
894 }
895 case 122: break;
896 case 67:
897     { return new Symbol(sym.Retornar, yychar, yyline, yytext());
898 }
899 case 123: break;
900 case 51:
901     { return new Symbol(sym.Entero, yychar, yyline, yytext());
902 }
903 case 124: break;
904 case 9:
905     { return new Symbol(sym.Mayor, yychar, yyline, yytext());
906 }
907 case 125: break;
908 case 62:
909     { return new Symbol(sym.Clase, yychar, yyline, yytext());
910 }
911 case 126: break;
912 case 71:
913     { return new Symbol(sym.Detener, yychar, yyline, yytext());
914 }
915 case 127: break;
916 case 18:
917     { return new Symbol(sym.Euler, yychar, yyline, yytext());
918 }
919 case 128: break;
920 case 45:
921     { return new Symbol(sym.NombreClase, yychar, yyline, yytext());
922 }
923 case 129: break;
924 case 33:
```



```
1025     { return new Symbol(sym.Increemento, yychar, yyline, yytext());
1026     }
1027     case 130: break;
1028     case 36:
1029     { return new Symbol(sym.FinalTexto, yychar, yyline, yytext());
1030     }
1031     case 131: break;
1032     case 43:
1033     { return new Symbol(sym.Punto, yychar, yyline, yytext());
1034     }
1035     case 132: break;
1036     case 65:
1037     { return new Symbol(sym.SiNoSi, yychar, yyline, yytext());
1038     }
1039     case 133: break;
1040     case 28:
1041     { return new Symbol(sym.Veradero, yychar, yyline, yytext());
1042     }
1043     case 134: break;
1044     case 23:
1045     { return new Symbol(sym.LlaveApertura, yychar, yyline, yytext());
1046     }
1047     case 135: break;
1048     case 30:
1049     { return new Symbol(sym.CaracteresEspeciales, yychar, yyline, yytext());
1050     }
1051     case 136: break;
1052     case 13:
1053     { return new Symbol(sym.Y, yychar, yyline, yytext());
1054     }
1055     case 137: break;
1056     case 68:
1057     { return new Symbol(sym.Arreglo, yychar, yyline, yytext());
1058     }
1059     case 138: break;
```

```
1060     case 50:
1061     { return new Symbol(sym.Positivo, yychar, yyline, yytext());
1062     }
1063     case 139: break;
1064     case 5:
1065     { return new Symbol(sym.Division, yychar, yyline, yytext());
1066     }
1067     case 140: break;
1068     case 7:
1069     { return new Symbol(sym.Resta, yychar, yyline, yytext());
1070     }
1071     case 141: break;
1072     case 19:
1073     { return new Symbol(sym.ParentesisApertura, yychar, yyline, yytext());
1074     }
1075     case 142: break;
1076     case 35:
1077     { return new Symbol(sym.FinComentario, yychar, yyline, yytext());
1078     }
1079     case 143: break;
1080     case 74:
1081     { return new Symbol(sym.Bibliotecas, yychar, yyline, yytext());
1082     }
1083     case 144: break;
1084     case 39:
1085     { return new Symbol(sym.MenorIgual, yychar, yyline, yytext());
1086     }
1087     case 145: break;
1088     case 61:
1089     { return new Symbol(sym.Hacer, yychar, yyline, yytext());
1090     }
1091     case 146: break;
1092     case 3:
1093     { return new Symbol(sym.Numero, yychar, yyline, yytext());
1094     }
1095     case 147: break;
```



```
1095     case 147: break;
1096     case 22:
1097         { return new Symbol(sym.CorcheteCierre, yychar, yyline, yytext());
1098     }
1099     case 148: break;
1100     case 40:
1101         { return new Symbol(sym.Asignacion, yychar, yyline, yytext());
1102     }
1103     case 149: break;
1104     case 69:
1105         { return new Symbol(sym.Imprimir, yychar, yyline, yytext());
1106     }
1107     case 150: break;
1108     case 42:
1109         { return new Symbol(sym.Distinto, yychar, yyline, yytext());
1110     }
1111     case 151: break;
1112     case 37:
1113         { return new Symbol(sym.MayorIgual, yychar, yyline, yytext());
1114     }
1115     case 152: break;
1116     default:
1117         if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
1118             zzAtEOF = true;
1119             zzDoEOF();
1120             { return new java_cup.runtime.Symbol(sym.EOF); }
1121         }
1122         else {
1123             zzScanError(ZZ_NO_MATCH);
1124         }
1125     }
1126 }
1127 }
1128 }
```

Sintaxis.cup

El archivo **Sintaxis.cup** define las reglas y la estructura sintáctica de un lenguaje de programación personalizado. Utiliza **Java CUP**, una herramienta que genera analizadores sintáticos basados en gramáticas específicas. Este archivo combina componentes para manejar errores, definir terminales y no terminales, y construir reglas de producción que guían cómo se interpretan las instrucciones del lenguaje.

El archivo CUP establece las bases para analizar y validar el código escrito en este lenguaje, asegurando que siga las reglas sintácticas definidas. Es esencial para construir compiladores o intérpretes, verificando que el programa sea estructuralmente correcto antes de ejecutarlo o traducirlo a un código ejecutable.

Código:

```
1 package codigo;
2
3 import java_cup.runtime.Symbol;
4 import java.util.ArrayList;
5 import java.util.List;
6
7
8 parser code
9 {:
10     private java.util.List<String> errores = new java.util.ArrayList<>();
11     private String textoCompleto; // Para calcular las columnas por linea
12
13     public void syntax_error(Symbol s) {
14         int linea = s.right + 1;
15         int columnaReal = calcularColumnaReal(linea, s.left + 1);
16
17         errores.add("Error de sintaxis. Línea: " + linea +
18                     ", Columna: " + columnaReal + ", Texto: '" + s.value + "'");
19     }
20
21     public java.util.List<String> getErrores() {
22         return errores;
23     }
24
25     // Configura el texto completo para cálculos de posición
26     public void setTextoCompleto(String textoCompleto) {
27         this.textoCompleto = textoCompleto;
28     }
29
30     // Calcula la columna real reiniciando en cada nueva línea
31     private int calcularColumnaReal(int linea, int posicionGlobal) {
32         int columna = 1;
33         int contadorLineas = 1;
34         int i = 0;
35
36         // Recorremos el texto hasta la posición global
```



MANUAL DE PRÁCTICAS



```
36 // Recorremos el texto hasta la posición global
37 while (i < posicionGlobal) {
38     char c = textoCompleto.charAt(i);
39
40     if (c == '\n' || (c == '\r' && i + 1 < textoCompleto.length() && textoCompleto.charAt(i + 1) == '\n')) {
41         // Si encontramos un salto de línea (considerando \r\n en Windows)
42         contadorLineas++;
43         if (contadorLineas == linea) {
44             columna = 1; // Reiniciamos la columna al principio de la nueva línea
45         }
46         if (c == '\r' && i + 1 < textoCompleto.length() && textoCompleto.charAt(i + 1) == '\n') {
47             i++; // Avanzamos el índice para saltar el '\n' que sigue al '\r'
48         }
49     } else if (contadorLineas == linea) {
50         columna++; // Si estamos en la línea correcta, aumentamos la columna
51     }
52     i++;
53 }
54 return columna;
55 }
56
57 :;
58
59
60
61 terminal Suma, Resta, Multiplicacion, Division, Mayor, Menor, MayorIgual, MenorIgual,
62 Comparacion, Distinto, Y, O, No, Modulo, Potencia, Raiz, Punto, Asignacion, Igual,
63 Concatenacion, Positivo, Negativo, Incremento, Decremento, Pi, Euler, ParentesisApertura,
64 ParentesisCierre, CorcheteApertura, CorcheteCierre, LlaveApertura, LlaveCierre, InicioTexto,
65 FinalTexto, ComillaDoble, ComillaSimple, Comentario, InicioComentario, FinComentario,
66 Entero, Decimal, Verdadero, Falso, Cadena, Si, SiNo, SiNoSi, CicloRepite, Mientras, Hacer,
67 Seleccionador, Caso, Predeterminado, Detener, Arreglo, Funcion, Procedimiento, Principal,
68 Biblioteca, Imprimir, Leer, Importar, Retornar, Metodo, Nulo, Saltolinea, CaracteresEspeciales,
69 P_coma, Main, Constante, NombreClase, Identificador, Numero, Clase, Texto, ERROR;
70
```

```
71 non terminal INICIO, SENTENCIA, DECLARACION, DECLARACION_FOR, IF, IF_ELSE,
72 WHILE, DO_WHILE, FOR, SENTENCIA_BOLEANA, SENTENCIA_FOR, METODO, PARAMETROS, CUERPO_METODO;
73
74 start with INICIO;
75
76 INICIO ::=*
77     Entero Principal ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre |
78     Clase ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre|
79     Clase NombreClase ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre |
80     error P_coma { System.out.println("Error en SENTENCIA"); :} |;
81
82 SENTENCIA ::=
83     SENTENCIA DECLARACION |
84     DECLARACION |
85     SENTENCIA IF |
86     IF |
87     SENTENCIA IF_ELSE |
88     IF_ELSE |
89     SENTENCIA WHILE |
90     WHILE |
91     SENTENCIA DO_WHILE |
92     DO_WHILE |
93     SENTENCIA FOR |
94     FOR |
95     METODO |
96     Imprimir ParentesisApertura Texto ParentesisCierre P_coma
97     | Leer ParentesisApertura Identificador ParentesisCierre P_coma
98     | Importar ParentesisApertura Texto ParentesisCierre P_coma |
99
100    error P_coma { System.out.println("Error en SENTENCIA"); :} |
101    error LlaveCierre { System.out.println("Error en SENTENCIA"); :}
102 ;
103
104 DECLARACION ::=
105     Identificador P_coma |
106     Identificador Asignacion Numero P_coma |
```



MANUAL DE PRÁCTICAS



```
106 Identificador Asignacion Numero P_coma |
107 Identificador Igual Numero P_coma |
108 Identificador Incremento P_coma |
109 Identificador Decremento P_coma |
110
111 Entero Identificador P_coma |
112 Entero Identificador Asignacion Numero P_coma |
113 Entero Identificador Igual Numero P_coma |
114 Entero Identificador Incremento P_coma |
115 Entero Identificador Decremento P_coma |
116
117 Decimal Identificador P_coma |
118 Decimal Identificador Asignacion Numero P_coma |
119 Decimal Identificador Igual Numero P_coma |
120 Decimal Identificador Incremento P_coma |
121 Decimal Identificador Decremento P_coma |
122
123 Entero Identificador Potencia Numero P_coma |
124 Entero Identificador Raiz Numero P_coma |
125 Entero Identificador Modulo Numero P_coma |
126 Entero Identificador Punto Numero P_coma |
127 Cadena Identificador Concatenacion Identificador P_coma|
128
129 Cadena Identificador Asignacion ComillaDoble ComillaDoble P_coma |
130 Cadena Identificador Igual ComillaDoble ComillaDoble P_coma |
131 Cadena Identificador Asignacion ComillaDoble Identificador ComillaDoble P_coma |
132 Cadena Identificador Igual ComillaDoble Identificador ComillaDoble P_coma |
133
134 Constante P_coma |
135 Constante Asignacion Numero P_coma |
136 Constante Igual Numero P_coma |
137 Constante Incremento P_coma |
138 Constante Decremento P_coma |
139
140 Entero Constante P_coma |
141 Entero Constante Asignacion Numero P_coma |
```

```
141 Entero Constante Asignacion Numero P_coma |
142 Entero Constante Igual Numero P_coma |
143 Entero Constante Incremento P_coma |
144 Entero Constante Decremento P_coma |
145
146 Decimal Constante P_coma |
147 Decimal Constante Asignacion Numero P_coma |
148 Decimal Constante Igual Numero P_coma |
149 Decimal Constante Incremento P_coma |
150 Decimal Constante Decremento P_coma |
151
152 Entero Constante Potencia Numero P_coma |
153 Entero Constante Raiz Numero P_coma |
154 Entero Constante Modulo Numero P_coma |
155 Entero Constante Punto Numero P_coma |
156 Cadena Constante Concatenacion Identificador P_coma |
157
158 Cadena Constante Asignacion ComillaDoble ComillaDoble P_coma |
159 Cadena Constante Igual ComillaDoble ComillaDoble P_coma |
160 Cadena Constante Asignacion ComillaDoble Identificador ComillaDoble P_coma |
161 Cadena Constante Igual ComillaDoble Identificador ComillaDoble P_coma ;
162
163 IF ::=
164     Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre |
165     Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre |
166     Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre |
167     Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
168
169 ;
170
171 SENTENCIA_BOLEANA ::=
172     Identificador Comparacion Numero |
173     Identificador Mayor Numero |
174     Identificador Menor Numero |
175     Identificador MayorIgual Numero |
176     Identificador MenorIgual Numero |
```



MANUAL DE PRÁCTICAS



```
176 Identificador MenorIgual Numero |
177 Identificador Distinto Numero |
178 Identificador Comparacion Identificador |
179 Identificador Comparacion ComillaDoble ComillaDoble |
180 Identificador Comparacion ComillaDoble Identificador ComillaDoble |
181 Identificador Igual Verdadero |
182 Identificador Igual Falso |
183 Identificador Comparacion Pi |
184 Identificador Comparacion Euler
185 ;
186
187 IF_ELSE ::=
188     Si ParentesisApertura SENTENCIA_BOOLEANA ParentesisCierre LlaveApertura SENTENCIA_LlaveCierre SiNo LlaveApertura SENTENCIA_LlaveCierre |
189     Si ParentesisApertura SENTENCIA_BOOLEANA ParentesisCierre LlaveApertura SENTENCIA_LlaveCierre SiNoSi ParentesisApertura SENTENCIA_BOOLEANA ParentesisCierre
190 ;
191
192 DO_WHILE ::=
193     Hacer LlaveApertura SENTENCIA_LlaveCierre Mientras ParentesisApertura SENTENCIA_BOOLEANA ParentesisCierre P_coma |
194     Hacer LlaveApertura SENTENCIA_LlaveCierre Mientras ParentesisApertura No SENTENCIA_BOOLEANA ParentesisCierre P_coma;
195
196
197 FOR ::=
198     CicloRepite ParentesisApertura SENTENCIA_FOR ParentesisCierre LlaveApertura SENTENCIA_LlaveCierre |
199     CicloRepite ParentesisApertura SENTENCIA_FOR ParentesisCierre LlaveApertura Imprimir ParentesisApertura Identificador ParentesisCierre P_coma LlaveCierre
200 ;
201
202 SENTENCIA_FOR ::=
203     Decimal Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma DECLARACION_FOR |
204     Identificador Igual Numero P_coma SENTENCIA_BOOLEANA P_coma DECLARACION_FOR;
205
206 DECLARACION_FOR ::=
207     Identificador Asignacion Numero |
208     Identificador Incremento |
209     Incremento Identificador;
```

```
211 PARAMETROS ::=
212     Identificador |
213     Identificador PARAMETROS ;
214
215 CUERPO_METODO ::=
216     SENTENCIA |
217     SENTENCIA Retornar Identificador P_coma |
218     SENTENCIA Retornar Numero P_coma |
219     SENTENCIA Retornar Texto P_coma |
220     SENTENCIA Retornar Nulo P_coma |
221     SENTENCIA Retornar Pi P_coma |
222     SENTENCIA Retornar Euler P_coma ;
223
224 METODO ::=
225     Metodo Identificador ParentesisApertura PARAMETROS ParentesisCierre LlaveApertura CUERPO_METODO LlaveCierre
226 ;
```

Sintaxis.java

El archivo Sintaxis.java es un componente generado automáticamente como parte del proceso de compilación de un analizador sintáctico utilizando Java CUP. Este archivo implementa un analizador LR (Left-to-Right, Rightmost derivation) que valida las estructuras del código fuente según las reglas definidas en el archivo de gramática (Sintaxis.cup). Su principal función es verificar si un programa cumple con la gramática del lenguaje y, en caso contrario, reportar errores.

El archivo Sintaxis.java valida la estructura de los programas escritos en el lenguaje definido. Es esencial para cualquier compilador o intérprete, ya que asegura que el código cumpla con las reglas sintácticas antes de continuar con otras fases, como el análisis semántico o la generación de código. Al ser generado automáticamente, es una pieza clave para transformar las definiciones de gramática en funcionalidad ejecutable.

Código:

```
5 //-----
6
7 package codigo;
8
9 import java_cup.runtime.Symbol;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 /** CUP v0.11a beta 20060608 generated parser.
14 * @version Mon Jan 06 05:13:27 CST 2025
15 */
16 public class Sintax extends java_cup.runtime.lr_parser {
17
18     /** Default constructor. */
19     public Sintax() {super();}
20
21     /** Constructor which sets the default scanner. */
22     public Sintax(java_cup.runtime.Scanner s) {super(s);}
23
24     /** Constructor which sets the default scanner. */
25     public Sintax(java_cup.runtime.Scanner s, java_cup.runtime.SymbolFactory sf) {super(s,sf);}
26
27     /** Production table. */
28     protected static final short production_table[][] =
29         unpackFromStrings(new String[] {
30             "\\"00\\245\\000\\002\\002\\004\\000\\002\\002\\011\\000\\002\\002",
31             "\\"01\\000\\002\\002\\011\\000\\002\\002\\004\\000\\002\\002\\002",
32             "\\"00\\002\\003\\003\\000\\002\\003\\003\\000\\002\\003\\004\\000",
33             "\\"00\\003\\003\\000\\002\\003\\004\\000\\002\\003\\003\\000\\002",
34             "\\"003\\004\\000\\002\\003\\003\\000\\002\\003\\004\\000\\002\\003",
35             "\\"003\\000\\002\\003\\004\\000\\002\\003\\003\\000\\002\\003\\004",
36             "\\"000\\002\\003\\003\\000\\002\\003\\003\\000\\002\\003\\004\\000",
37             "\\"002\\003\\003\\000\\002\\003\\007\\000\\002\\003\\007\\000\\002\\003",
38             "\\"003\\011\\000\\002\\003\\007\\000\\002\\003\\007\\000\\002\\003",
39             "\\"007\\000\\002\\003\\004\\000\\002\\003\\004\\000\\002\\004\\004",
40             "\\"000\\002\\004\\006\\000\\002\\004\\006\\000\\002\\004\\010\\000"
        });
41 }
```



```
57      "\002\000\000\000\002\003\007\000\002\003\007\000\002\003" +
58      "\003\011\000\002\003\007\000\002\003\007\000\002\003" +
59      "\007\000\002\003\004\000\002\003\004\000\002\004\004" +
60      "\000\002\004\006\000\002\004\006\000\002\004\006\000" +
61      "\002\004\010\000\002\004\005\000\002\004\005\000\002" +
62      "\004\005\000\002\004\004\007\000\002\004\007\000\002\004" +
63      "\006\000\002\004\006\000\002\004\005\000\002\004\011" +
64      "\000\002\004\007\001\000\002\004\006\000\002\004\006\000" +
65      "\002\004\007\000\002\004\007\000\002\004\010\000\002\004" +
66      "\010\000\002\004\011\000\002\004\011\000\002\004\007" +
67      "\000\002\004\007\000\002\004\010\000\002\004\010\000" +
68      "\002\004\004\000\002\004\006\000\002\004\006\000\002" +
69      "\004\005\000\002\004\005\000\002\004\005\000\002\004" +
70      "\007\000\002\004\007\000\002\004\006\000\002\004\006\000" +
71      "\000\002\004\007\001\000\002\004\006\000\002\004\006\000" +
72      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
73      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
74      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
75      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
76      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
77      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
78      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
79      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
80      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
81      "\000\002\004\007\000\002\004\006\000\002\004\006\000" +
82      /* Access to production table. */
83      public short[][] production_table() {return _production_table;}
84
85      /* Parse-action table. */
86      protected static final short[][] action_table =
87          unpackFromStrings(new String[] {
88              "\000\023e\000\012\002\ufffc\003\004\053\007\115\005\001" +
89              "\002\000\004\107\0u240\001\002\000\006\036\0u236\112\0u235" +
90              "\001\002\000\004\002\023\001\002\000\004\075\010\001" +
91              "\002\000\004\036\011\001\002\000\004\037\012\001\002" +
92              "\000\004\042\013\001\002\000\044\003\037\053\024\054" +
93              "\036\057\033\060\014\063\044\064\025\065\034\077\023" +
94              "\100\015\101\042\103\045\111\031\113\026\114\016\117" +
95              "\043\120\017\001\002\000\004\036\0u1a1\001\002\000\004" +
96              "\036\0u19d\001\002\000\020\004\0u130\005\0u0133\006\0u012c\007" +
97              "\u012e\021\0u134\022\0u0131\023\0u132\001\002\000\032\043\ufffa" +
98              "\053\ufffa\054\ufffa\057\ufffa\060\ufffa\063\ufffa\064\ufffa\065" +
99              "\ufffa\102\ufffa\111\ufffa\113\ufffa\114\ufffa\001\002\003" +
100             "\043\ufff2\053\ufff2\054\ufff2\057\ufff2\060\ufff2\063\ufff2\064" +
101             "\ufff2\065\ufff2\102\ufff2\111\ufff2\113\ufff2\114\ufff2\001\002" +
102             "\000\030\043\0u19e\053\024\054\036\057\033\060\014\063" +
103             "\044\064\025\065\034\111\031\113\026\114\016\001\002" +
104             "\000\032\043\ufff8\053\ufff8\054\ufff8\057\ufff8\060\ufff8\063" +
105             "\ufff8\064\ufff8\065\ufff8\102\ufff8\111\ufff8\113\ufff8\114\ufff8" +
106             "\001\002\000\004\036\0u19b\001\002\000\006\111\0u016a\113" +
107             "\u016b\001\002\000\004\036\0u0152\001\002\000\014\025\0u0120" +
108             "\026\0u121\032\0u011d\033\0u011f\107\0u011e\001\002\000\032\043" +
109             "\ufff6\053\ufff6\054\ufff6\057\ufff6\060\ufff6\063\ufff6\064\ufff6" +
110             "\065\ufff6\102\ufff6\111\ufff6\113\ufff6\114\ufff6\001\002\000" +
111             "\032\043\ufff4\053\ufff4\054\ufff4\057\ufff4\060\ufff4\063\ufff4" +
112             "\064\ufff4\065\ufff4\102\ufff4\111\ufff4\113\ufff4\114\ufff4\001" +
113             "\002\000\014\025\0u0115\026\0u0116\032\0u0113\033\0u0114\107\0u0112"
```



MANUAL DE PRÁCTICAS



```
115 "\uffed\063\uffed\064\uffed\065\uffed\102\uffed\111\uffed\113\uffed" +
116 "\114\uffed\001\002\000\006\111\354\113\355\001\002\000" +
117 "\004\042\330\001\002\000\032\043\ufff0\053\ufff0\054\ufff0" +
118 "\057\ufff0\060\ufff0\063\ufff0\064\ufff0\065\ufff0\012\ufff0\111" +
119 "\ufff0\113\ufff0\114\ufff0\001\002\000\006\111\270\113\271" +
120 "\001\002\000\006\043\267\107\266\001\002\000\032\043" +
121 "\uffeb\053\uffeb\054\uffeb\057\uffeb\060\uffeb\063\uffeb\064\uffeb" +
122 "\065\uffeb\102\uffeb\111\uffeb\113\uffeb\114\uffeb\001\002\000" +
123 "\032\043\ufree\053\ufree\054\ufree\057\ufree\060\ufree\063\ufree" +
124 "\064\ufree\065\ufree\102\ufree\111\ufree\113\ufree\114\ufree\001" +
125 "\002\000\004\036\257\001\002\000\032\043\ufffb\053\ufffb" +
126 "\054\ufffb\057\ufffb\060\ufffb\063\ufffb\064\ufffb\065\ufffb\102" +
127 "\ufffb\111\ufffb\113\ufffb\114\ufffb\001\002\000\004\036\112" +
128 "\001\002\000\004\113\046\001\002\000\004\036\047\001" +
129 "\002\000\012\053\052\054\053\057\054\113\051\001\002" +
130 "\000\004\037\061\001\002\000\014\037\uff70\053\052\054" +
131 "\053\057\054\113\051\001\002\000\004\113\057\001\002" +
132 "\000\004\113\056\001\002\000\004\113\055\001\002\000" +
133 "\004\037\uf6d\001\002\000\004\037\uf6e\001\002\000\004" +
134 "\037\uf6e\001\002\000\004\037\uf6e\001\002\000\004\042" +
135 "\062\001\002\000\004\003\037\053\024\054\036\057\033" +
136 "\060\014\063\044\064\025\065\034\077\023\100\015\101" +
137 "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
138 "\001\002\000\032\043\uff6b\053\024\054\036\057\033\060" +
139 "\014\063\044\064\025\065\034\102\073\111\031\113\026" +
140 "\114\016\001\002\000\004\043\065\001\002\000\032\043" +
141 "\uff64\053\uff64\054\uff64\057\uff64\060\uff64\063\uff64\064\uff64" +
142 "\065\uff64\102\uff64\111\uff64\113\uff64\114\uff64\001\002\000" +
143 "\032\043\ufef3\053\ufef3\054\ufef3\057\ufef3\060\ufef3\063\ufef3" +
144 "\064\ufef3\065\ufef3\102\ufef3\111\ufef3\113\ufef3\114\ufef3\001" +
145 "\002\000\032\043\ufef9\053\ufef9\054\ufef9\057\ufef9\060\ufef9" +
146 "\063\ufef9\064\ufef9\065\ufef9\102\ufef9\111\ufef9\113\ufef9\114" +
147 "\uffef9\001\002\000\032\043\uffef7\053\uffef7\054\uffef7\057\uffef7" +
148 "\060\ufef7\063\ufef7\064\ufef7\065\ufef7\102\ufef7\111\ufef7\113" +
149 "\uffef7\114\ufef7\001\002\000\032\043\uffef5\053\uffef5\054\uffef5" +
150 "\057\uffef5\060\uffef5\063\uffef5\064\uffef5\065\uffef5\102\uffef5\111" +
```

```
151 "\uffef5\113\uffef5\114\uffef5\001\002\000\032\043\ufff1\053\ufff1" +
152 "\054\ufff1\057\ufff1\060\ufff1\063\ufff1\064\ufff1\065\ufff1\102" +
153 "\ufff1\111\ufff1\113\ufff1\114\ufff1\001\002\000\016\034\103" +
154 "\035\077\104\101\103\102\114\076\116\100\001\002\000" +
155 "\032\043\uffee\053\uffee\054\uffee\057\uffee\060\uffee\063\uffee" +
156 "\064\uffee\065\uffee\102\uffee\111\uffee\113\uffee\114\uffee\001" +
157 "\002\000\032\043\uffee\053\uffee\054\uffee\057\uffee\060\uffee" +
158 "\063\uffee\064\uffee\065\uffee\102\uffee\111\uffee\113\uffee\114" +
159 "\uffee\001\002\000\004\107\111\001\002\000\004\107\110" +
160 "\001\002\000\004\107\107\001\002\000\004\107\106\001" +
161 "\002\000\004\107\105\001\002\000\004\107\104\001\002" +
162 "\000\004\043\ufef6\001\002\000\004\043\ufef6\001\002\000" +
163 "\004\043\ufef6\001\002\000\004\043\ufef6\001\002\000\004" +
164 "\043\ufef6\001\002\000\004\043\ufef6\001\002\000\006\053" +
165 "\115\113\114\001\002\000\004\037\233\001\002\000\004" +
166 "\026\225\001\002\000\004\113\116\001\002\000\004\010" +
167 "\120\011\117\012\121\013\122\026\123\001\002\000\004" +
168 "\114\220\001\002\000\004\114\231\001\002\000\004\114" +
169 "\206\001\002\000\004\114\201\001\002\000\004\114\124" +
170 "\001\002\000\004\107\125\001\002\000\004\113\127\001" +
171 "\002\000\004\107\116\001\002\000\022\010\132\011\131" +
172 "\012\135\013\136\014\134\015\130\021\133\026\137\001" +
173 "\002\000\004\114\165\001\002\000\004\114\164\001\002" +
174 "\000\004\114\163\001\002\000\004\114\160\001\002\000" +
175 "\116\034\152\035\147\046\146\113\151\114\145\116\150" +
176 "\001\002\000\004\114\144\001\002\000\004\114\143\001" +
177 "\002\000\010\055\140\056\141\116\012\001\002\000\014" +
178 "\016\uff98\017\uff98\037\uff98\077\uff98\113\uff98\001\002\000" +
179 "\014\016\007\017\uff97\037\uff97\070\uff97\113\uff97\001\002" +
180 "\000\014\016\007\017\uff98\037\uff98\0107\uff98\113\uff98\001" +
181 "\002\000\014\016\007\uffa0\017\uffa0\037\uffa0\0107\uffa0\113\uffa0" +
182 "\001\002\000\014\016\007\uffa1\017\uffa1\037\uffa1\0107\uffa1\113" +
183 "\uffa1\001\002\000\014\016\007\uffa4\017\uffa4\037\uffa4\0107\uffa4" +
184 "\113\uffa4\001\002\000\010\046\153\113\155\116\154\001" +
185 "\002\000\014\016\007\uff95\037\uff95\0107\uff95\113\uff95" +
186 "\001\002\000\014\016\007\uff9d\017\uff9d\037\uff9d\0107\uff9d\113" +
```



MANUAL DE PRÁCTICAS



```

184 "\u113\uffa4\001\002\000\010\046\153\113\155\116\154\001" +
185 "\002\000\014\016\uff95\017\uff95\017\uff95\107\uff95\113\uff95" +
186 "\001\002\000\014\016\uff9d\017\uff9d\037\uff9d\107\uff9d\113" +
187 "\uff9d\001\002\000\014\016\uff9e\017\uff9e\037\uff9e\107\uff9e" +
188 "\113\uff9e\001\002\000\014\016\uff96\017\uff96\037\uff96\107" +
189 "\uff96\113\uff96\001\002\000\014\016\uff9b\017\uff9b\037\uff9b" +
190 "\107\uff9b\113\uff9b\001\002\000\046\157\001\002\000" +
191 "\004\046\156\001\002\000\014\016\uff9a\017\uff9a\037\uff9a" +
192 "\107\uff9a\113\uff9a\001\002\000\014\016\uff99\017\uff99\037" +
193 "\uff99\107\uff99\113\uff99\001\002\000\040\014\161\001\002" +
194 "\000\004\114\162\001\002\000\014\016\uff94\017\uff94\037" +
195 "\uff94\107\uff94\113\uff94\001\002\000\014\016\uffa3\017\uffa3" +
196 "\037\uffa3\107\uffa3\113\uffa3\001\002\000\014\016\uffa2\017" +
197 "\uffa2\037\uffa2\107\uffa2\113\uffa2\001\002\000\014\016\uff9f" +
198 "\017\uff9f\037\uff9f\107\uff9f\113\uff9f\001\002\000\010\032" +
199 "\167\033\171\113\172\001\002\000\004\113\200\001\002" +
200 "\000\004\037\uff7b\001\002\000\004\113\177\001\002\000" +
201 "\010\025\175\032\173\033\174\001\002\000\004\037\uff74" +
202 "\001\002\000\004\037\uff73\001\002\000\004\114\176\001" +
203 "\002\000\004\037\uff75\001\002\000\004\037\uff71\001\002" +
204 "\000\004\037\uff72\001\002\000\004\107\202\001\002\000" +
205 "\004\113\127\001\002\000\004\107\204\001\002\000\010" +
206 "\032\167\033\171\113\172\001\002\000\004\037\uff77\001" +
207 "\002\000\004\107\207\001\002\000\004\113\127\001\002" +
208 "\000\004\107\211\001\002\000\004\103\167\033\171\113" +
209 "\172\001\002\000\004\037\uff78\001\002\000\004\107\214" +
210 "\001\002\000\004\113\127\001\002\000\004\107\216\001" +
211 "\002\000\010\032\167\033\171\113\172\001\002\000\004" +
212 "\037\uff7a\001\002\000\004\107\221\001\002\000\004\113" +
213 "\127\001\002\000\004\107\223\001\002\000\004\10\032\167" +
214 "\033\171\113\172\001\002\000\004\037\uff79\001\002\000" +
215 "\004\114\226\001\002\000\004\107\227\001\002\000\004" +
216 "\113\127\001\002\000\004\107\231\001\002\000\010\032" +
217 "\167\033\171\113\172\001\002\000\004\037\uff76\001\002" +
218 "\000\004\042\234\001\002\000\044\003\037\053\024\054" +
219 "\036\057\033\060\014\063\044\064\025\065\034\034\077\236" +

```

```

220 "\100\015\101\042\103\045\111\037\113\026\114\016\117" +
221 "\043\120\017\001\002\000\030\043\256\053\024\054\036" +
222 "\057\033\060\014\063\044\064\025\065\034\111\031\113" +
223 "\026\114\016\001\002\000\004\036\237\001\002\000\006" +
224 "\113\241\116\240\001\002\000\006\027\245\037\246\001" +
225 "\002\000\004\037\242\001\002\000\004\107\243\001\002" +
226 "\000\004\043\244\001\002\000\032\043\uff7c\053\uff7c\054" +
227 "\uff7c\057\uff7c\060\uff7c\063\uff7c\064\uff7c\065\uff7c\102\uff7c" +
228 "\111\uff7c\113\uff7c\114\uff7c\001\002\000\004\113\253\001" +
229 "\002\000\006\027\247\107\250\001\002\000\004\113\251" +
230 "\001\002\000\032\043\uffea\053\uffea\054\uffea\057\uffea\060" +
231 "\uffea\063\uffea\064\uffea\065\uffea\102\uffea\111\uffea\113\uffea" +
232 "\114\uffea\001\002\000\004\107\252\001\002\000\032\043" +
233 "\uffe9\053\uffe9\054\uffe9\057\uffe9\060\uffe9\063\uffe9\064\uffe9" +
234 "\065\uffe9\102\uffe9\111\uffe9\113\uffe9\114\uffe9\001\002\000" +
235 "\004\037\254\001\002\000\004\107\255\001\002\000\032" +
236 "\043\uffe8\053\uffe8\054\uffe8\057\uffe8\060\uffe8\063\uffe8\064" +
237 "\uffe8\065\uffe8\102\uffe8\111\uffe8\113\uffe8\114\uffe8\001\002" +
238 "\000\032\043\uff7d\053\uff7d\054\uff7d\057\uff7d\060\uff7d\063" +
239 "\uff7d\064\uff7d\065\uff7d\102\uff7d\111\uff7d\113\uff7d\114\uff7d" +
240 "\001\002\000\006\121\116\260\001\002\000\004\037" +
241 "\264\001\002\000\004\037\262\001\002\000\004\107\263" +
242 "\001\002\000\032\043\uffe5\053\uffe5\054\uffe5\057\uffe5\060" +
243 "\uffe5\063\uffe5\064\uffe5\065\uffe5\102\uffe5\111\uffe5\113\uffe5" +
244 "\114\uffe5\001\002\000\004\107\265\001\002\000\032\043" +
245 "\uffe6\053\uffe6\054\uffe6\057\uffe6\060\uffe6\063\uffe6\064\uffe6" +
246 "\065\uffe6\102\uffe6\111\uffe6\113\uffe6\114\uffe6\001\002\000" +
247 "\032\043\uffe4\053\uffe4\054\uffe4\054\uffe4\057\uffe4\060\uffe4\063\uffe4" +
248 "\064\uffe4\065\uffe4\102\uffe4\111\uffe4\113\uffe4\114\uffe4\001" +
249 "\002\000\032\043\uffe3\053\uffe3\054\uffe3\057\uffe3\060\uffe3" +
250 "\063\uffe3\064\uffe3\065\uffe3\102\uffe3\111\uffe3\113\uffe3\114" +
251 "\uffe3\001\002\000\014\025\314\026\315\032\312\033\313" +
252 "\107\311\001\002\000\014\025\275\026\276\032\273\033" +
253 "\274\107\272\001\002\000\032\043\ufd6\053\ufd6\054\ufd6" +
254 "\057\ufd6\060\ufd6\063\ufd6\064\ufd6\065\ufd6\102\ufd6\111" +
255 "\ufd6\113\ufd6\114\ufd6\001\002\000\004\107\310\001\002" +

```



```
257    "\004\114\277\001\002\000\004\024\300\001\002\000\000" +
258    "\114\301\001\002\000\004\107\302\001\002\000\032\043" +
259    "\ufffd4\053\ufffd4\054\ufffd4\057\ufffd4\060\ufffd4\063\ufffd4\064\ufffd4" +
260    "\065\ufffd4\102\ufffd4\111\ufffd4\113\ufffd4\114\ufffd4\001\002\000" +
261    "\004\024\304\001\002\000\004\114\305\001\002\000\004" +
262    "\107\306\001\002\000\032\043\ufffd5\053\ufffd5\054\ufffd5\057" +
263    "\ufffd5\060\ufffd5\063\ufffd5\064\ufffd5\065\ufffd5\102\ufffd5\111\ufffd5" +
264    "\113\ufffd5\114\ufffd5\001\002\000\032\043\ufffd2\053\ufffd2\054" +
265    "\ufffd2\057\ufffd2\060\ufffd2\063\ufffd2\064\ufffd2\065\ufffd2\102\uffd2" +
266    "\111\ufffd2\113\ufffd2\114\ufffd2\001\002\000\032\043\ufffd3\053" +
267    "\ufffd3\054\ufffd3\057\ufffd3\060\ufffd3\063\ufffd3\064\ufffd3\065\ufffd3" +
268    "\102\ufffd3\111\ufffd3\113\ufffd3\114\ufffd3\001\002\000\032\043" +
269    "\ufffb8\053\ufffb8\054\ufffb8\057\ufffb8\060\ufffb8\063\ufffb8\064\ufffb8" +
270    "\065\ufffb8\102\ufffb8\111\ufffb8\113\ufffb8\001\002\000" +
271    "\004\107\327\001\002\000\004\107\326\001\002\000\004" +
272    "\114\322\001\002\000\004\114\316\001\002\000\004\024" +
273    "\317\001\002\000\004\114\320\001\002\000\004\107\321" +
274    "\001\002\000\032\043\ufffb8\053\ufffb8\054\ufffb8\057\ufffb8\060" +
275    "\ufffb8\063\ufffb8\064\ufffb8\065\ufffb8\102\ufffb8\111\ufffb8\113\ufffb8" +
276    "\114\ufffb8\001\002\000\004\124\323\001\002\000\004\114" +
277    "\324\001\002\000\004\107\325\001\002\000\032\043\ufffb9" +
278    "\053\ufffb9\054\ufffb9\057\ufffb9\060\ufffb9\063\ufffb9\064\ufffb9\065" +
279    "\ufffb9\102\ufffb9\111\ufffb9\113\ufffb9\114\ufffb9\001\002\000\032" +
280    "\043\ufffb6\053\ufffb6\054\ufffb6\057\ufffb6\060\ufffb6\063\ufffb6\064" +
281    "\ufffb6\065\ufffb6\102\ufffb6\111\ufffb6\113\ufffb6\114\ufffb6\001\002" +
282    "\000\032\043\ufffb7\053\ufffb7\054\ufffb7\057\ufffb7\060\ufffb7\063" +
283    "\ufffb7\064\ufffb7\065\ufffb7\102\ufffb7\111\ufffb7\113\ufffb7\114\ufffb7" +
284    "\001\002\000\004\003\037\053\024\054\036\057\033\060" +
285    "\014\063\044\064\025\065\034\077\023\100\015\101\042" +
286    "\103\045\111\031\113\026\114\016\117\043\120\017\001" +
287    "\002\000\030\043\332\053\024\054\036\057\033\060\014" +
288    "\063\044\064\025\065\034\111\031\113\026\114\016\001" +
289    "\002\000\004\064\033\001\002\000\004\036\034\001\002" +
290    "\000\006\020\035\113\127\001\002\000\004\113\127\001" +
291    "\002\000\010\016\341\017\340\037\337\001\002\000\004" +
292    "\107\350\001\002\000\004\113\127\001\002\000\004\113" +
```

```
293    "\127\001\002\000\004\037\343\001\002\000\004\107\344" +
294    "\001\002\000\032\043\uffe0\053\uffe0\054\uffe0\057\uffe0\060" +
295    "\uffe0\063\uffe0\064\uffe0\065\uffe0\102\uffe0\111\uffe0\113\uffe0" +
296    "\114\uffe0\001\002\000\004\037\346\001\002\000\004\107" +
297    "\347\001\002\000\032\043\uffe7\053\uffe7\054\uffe7\057\uffe7\056" +
298    "\060\uffe7\063\uffe7\064\uffe7\065\uffe7\102\uffe7\111\uffe7\113" +
299    "\uffe7\114\uffe7\001\002\000\032\043\uffe8\053\uffe8\054\uffe8\055" +
300    "\057\uffe8\060\uffe8\063\uffe8\064\uffe8\065\uffe8\102\uffe8\111" +
301    "\uffe8\113\uffe8\114\uffe8\001\002\000\032\043\uffe7\053\uffe7\054" +
302    "\000\004\107\353\001\002\000\032\043\uffe7\053\uffe7\054" +
303    "\uffe7\057\uffe7\060\uffe7\063\uffe7\064\uffe7\065\uffe7\102\uffe7\066" +
304    "\uffe7\113\uffe7\114\uffe7\114\uffe7\001\002\000\010\025\0102\026" +
305    "\u0103\027\001\001\002\000\010\025\357\026\360\027\356" +
306    "\001\002\000\004\003\037\001\002\000\004\046\371\001" +
307    "\002\000\006\046\361\116\362\001\002\000\006\046\364" +
308    "\113\365\001\002\000\004\107\363\001\002\000\032\043" +
309    "\ufffaa\053\ufffaa\054\ufffaa\057\ufffaa\060\ufffaa\063\ufffaa\064\ufffaa" +
310    "\065\ufffaa\102\ufffaa\111\ufffaa\113\ufffaa\114\ufffaa\001\002\000" +
311    "\004\107\370\001\002\000\004\046\366\001\002\000\004" +
312    "\107\367\001\002\000\032\043\uffc9\053\uffc9\054\uffc9\057" +
313    "\uffc9\060\uffc9\063\uffc9\064\uffc9\065\uffc9\102\uffc9\111\uffc9" +
314    "\uffc9\113\uffc9\114\uffc9\001\002\000\032\043\uffcb\053\uffcb\054" +
315    "\uffcb\057\uffcb\060\uffcb\063\uffcb\064\uffcb\065\uffcb\102\uffcb" +
316    "\uffcb\113\uffcb\114\uffcb\001\002\000\006\046\372\113" +
317    "\373\001\002\000\004\107\376\001\002\000\004\046\374" +
318    "\001\002\000\004\107\375\001\002\000\032\043\uffca\053" +
319    "\uffca\054\uffca\057\uffca\060\uffca\063\uffca\064\uffca\065\uffca" +
320    "\102\uffca\111\uffca\113\uffca\114\uffca\001\002\000\032\043" +
321    "\uffcc\053\uffcc\054\uffcc\057\uffcc\060\uffcc\063\uffcc\064\uffcc" +
322    "\065\uffcc\102\uffcc\111\uffcc\113\uffcc\114\uffcc\001\002\000" +
323    "\004\107\0100\001\002\000\032\043\uffcd\053\uffcd\054\uffcd" +
324    "\057\uffcd\060\uffcd\063\uffcd\064\uffcd\065\uffcd\102\uffcd\111" +
325    "\uffcd\113\uffcd\114\uffcd\001\002\000\004\113\011\001\002" +
326    "\000\004\046\010\001\002\000\004\046\0104\001\002\000" +
327    "\006\046\0105\113\0106\001\002\000\004\107\0109\001\002" +
328    "\000\004\046\0107\001\002\000\004\107\0108\001\002\000" +
```



MANUAL DE PRÁCTICAS



```
330 " \u064\uffad\065\uffad\102\uffad\111\uffad\113\uffad\114\uffad\001" +
331 " \u002\000\032\043\uffaf\053\uffaf\054\uffaf\057\uffaf\060\uffaf" +
332 " \u063\uffaf\064\uffaf\065\uffaf\067\uffaf\102\uffaf\111\uffaf\113\uffaf\114" +
333 " \uffaf\001\002\000\006\046\uf01b\113\uf01c\001\002\000\004" +
334 " \107\uf010\001\002\000\004\046\uf01d\001\002\000\004\107" +
335 " \u010e\001\002\000\032\043\uffae\053\uffae\054\uffae\057\uffae" +
336 " \u060\uffae\063\uffae\064\uffae\065\uffae\102\uffae\111\uffae\113" +
337 " \uffae\114\uffae\001\002\000\032\043\uffb\053\uffb\054\uffb\057\uffb\060" +
338 " \uffb\057\uffb\060\uffb\063\uffb\064\uffb\065\uffb\066\uffb\067\uffb\070" +
339 " \uffb\060\uffb\063\uffb\064\uffb\065\uffb\066\uffb\067\uffb\071" +
340 " \u000\032\043\uffb\053\uffb\054\uffb\055\uffb\056\uffb\057\uffb\058" +
341 " \uffb\059\uffb\060\uffb\061\uffb\062\uffb\063\uffb\064\uffb\065\uffb\066" +
342 " \u001\002\000\032\043\uffc\053\uffc\054\uffc\055\uffc\056\uffc\057" +
343 " \uffc\058\uffc\063\uffc\064\uffc\065\uffc\066\uffc\067\uffc\068" +
344 " \uffc\069\uffc\070\uffc\071\uffc\072\uffc\073\uffc\074\uffc\075\uffc\076" +
345 " \uffc\077\uffc\078\uffc\079\uffc\080\uffc\081\uffc\082\uffc\083\uffc\084" +
346 " \uffc\085\uffc\086\uffc\087\uffc\088\uffc\089\uffc\090\uffc\091\uffc\092" +
347 " \uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098\uffc\099\uffc\090" +
348 " \uffc\091\001\002\000\032\043\uffc\053\uffc\054\uffc\055\uffc\056" +
349 " \uffc\057\uffc\058\uffc\059\uffc\060\uffc\061\uffc\062\uffc\063\uffc\064" +
350 " \uffc\065\uffc\066\uffc\067\uffc\068\uffc\069\uffc\070\uffc\071\uffc\072" +
351 " \uffc\073\uffc\074\uffc\075\uffc\076\uffc\077\uffc\078\uffc\079\uffc\080" +
352 " \uffc\082\uffc\083\uffc\084\uffc\085\uffc\086\uffc\087\uffc\088\uffc\089" +
353 " \uffc\081\uffc\082\uffc\083\uffc\084\uffc\085\uffc\086\uffc\087\uffc\088" +
354 " \uffc\089\uffc\090\uffc\091\uffc\092\uffc\093\uffc\094\uffc\095\uffc\096" +
355 " \uffc\097\uffc\098\uffc\099\uffc\090\uffc\091\uffc\092\uffc\093\uffc\094" +
356 " \uffc\095\uffc\096\uffc\097\uffc\098\uffc\099\uffc\090\uffc\091\uffc\092" +
357 " \uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098\uffc\099\uffc\090" +
358 " \uffc\091\uffc\092\uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098" +
359 " \uffc\092\uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098\uffc\099" +
360 " \uffc\094\uffc\095\uffc\096\uffc\097\uffc\098\uffc\099\uffc\090\uffc\091" +
361 " \uffc\096\uffc\097\uffc\098\uffc\099\uffc\090\uffc\091\uffc\092\uffc\093" +
362 " \uffc\098\uffc\099\uffc\090\uffc\091\uffc\092\uffc\093\uffc\094\uffc\095" +
363 " \uffc\099\uffc\090\uffc\091\uffc\092\uffc\093\uffc\094\uffc\095\uffc\096" +
364 " \uffc\091\uffc\092\uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098" +
365 " \uffc\092\uffc\093\uffc\094\uffc\095\uffc\096\uffc\097\uffc\098\uffc\099" +
```

```
375 " \uffe0\060\uffe0\063\uffe0\064\uffe0\065\uffe0\102\uffe0\111\uffe0" +
376 " \113\uffe0\114\uffe0\001\002\000\004\114\uf01d\001\002\000" +
377 " \004\114\uf01b\001\002\000\004\114\uf013\001\002\000\004" +
378 " \114\uf013\001\002\000\004\114\uf013\001\002\000\004\107" +
379 " \u0136\001\002\000\032\043\ufff\053\ufff\054\ufff\055\ufff\057\ufff\058" +
380 " \ufff\063\ufff\064\ufff\065\ufff\066\ufff\067\ufff\068\ufff\069\ufff\070" +
381 " \ufff\071\ufff\072\ufff\073\ufff\074\ufff\075\ufff\076\ufff\077\ufff\078" +
382 " \ufff\079\ufff\080\ufff\081\ufff\082\ufff\083\ufff\084\ufff\085\ufff\086" +
383 " \ufff\087\ufff\088\ufff\089\ufff\080\ufff\081\ufff\082\ufff\083\ufff\084" +
384 " \ufff\085\ufff\086\ufff\087\ufff\088\ufff\089\ufff\080\ufff\081\ufff\082" +
385 " \ufff\083\ufff\084\ufff\085\ufff\086\ufff\087\ufff\088\ufff\089\ufff\080" +
386 " \ufff\081\ufff\082\ufff\083\ufff\084\ufff\085\ufff\086\ufff\087\ufff\088" +
387 " \ufff\089\ufff\090\ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096" +
388 " \ufff\097\ufff\098\ufff\099\ufff\090\ufff\091\ufff\092\ufff\093\ufff\094" +
389 " \ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090\ufff\091\ufff\092" +
390 " \ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090" +
391 " \ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098" +
392 " \ufff\099\ufff\090\ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096" +
393 " \ufff\098\ufff\099\ufff\090\ufff\091\ufff\092\ufff\093\ufff\094\ufff\095" +
394 " \ufff\097\ufff\098\ufff\099\ufff\090\ufff\091\ufff\092\ufff\093\ufff\094" +
395 " \ufff\096\ufff\097\ufff\098\ufff\099\ufff\090\ufff\091\ufff\092\ufff\093" +
396 " \ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090\ufff\091\ufff\092" +
397 " \ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090\ufff\091" +
398 " \ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090" +
399 " \ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099" +
400 " \ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098" +
401 " \ufff\090\ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097" +
402 " \ufff\099\ufff\098\ufff\097\ufff\096\ufff\095\ufff\094\ufff\093\ufff\092" +
403 " \ufff\097\ufff\096\ufff\095\ufff\094\ufff\093\ufff\092\ufff\091\ufff\090" +
404 " \ufff\098\ufff\097\ufff\096\ufff\095\ufff\094\ufff\093\ufff\092\ufff\091" +
405 " \ufff\099\ufff\098\ufff\097\ufff\096\ufff\095\ufff\094\ufff\093\ufff\092" +
406 " \ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098" +
407 " \ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099\ufff\090" +
408 " \ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098\ufff\099" +
409 " \ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097\ufff\098" +
410 " \ufff\090\ufff\091\ufff\092\ufff\093\ufff\094\ufff\095\ufff\096\ufff\097" +
```



MANUAL DE PRÁCTICAS



```

411   "\ufffdf\102\ufffd\111\ufffdf\113\ufffdf\114\ufffd\001\002\000\032" +
412   "\043\ufffd\053\ufffd\054\ufffd\057\ufffd\060\ufffd\063\ufffd\064" +
413   "\ufffd\065\ufffd\102\ufffd\111\ufffd\113\ufffd\114\ufffd\001\002" +
414   "\000\032\043\ufffd\053\ufffd\054\ufffd\057\ufffd\060\ufffd\063" +
415   "\ufffd\064\ufffd\065\ufffd\102\ufffd\111\ufffd\113\ufffd\114\ufffd" +
416   "\001\002\000\006\020\0153\127\001\002\000\004\113" +
417   "\127\001\002\000\010\016\001\0157\017\010\0155\037\0156\001\002" +
418   "\000\004\113\127\001\002\000\004\042\015d\001\002\000" +
419   "\004\113\127\001\002\000\004\037\010\0159\001\002\000\004" +
420   "\042\015a\001\002\000\004\033\037\053\024\054\036\057" +
421   "\033\060\014\063\044\064\025\065\034\077\023\100\015" +
422   "\101\042\103\045\111\031\113\026\114\016\117\043\120" +
423   "\017\001\002\000\030\043\015\053\024\054\036\057\033" +
424   "\060\014\063\044\064\025\065\034\111\031\113\026\114" +
425   "\016\001\002\000\032\043\ufffd\053\ufffd\054\ufffd\057\ufffd" +
426   "\060\ufb84\063\ufb84\064\ufb84\065\ufb84\102\ufb84\111\ufb84\113" +
427   "\ufb84\114\ufb84\001\002\000\004\003\037\053\024\054\036" +
428   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
429   "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
430   "\120\017\001\002\000\030\043\015\053\024\054\036\057" +
431   "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
432   "\114\016\001\002\000\032\043\ufb85\053\ufb85\054\uffb85\057" +
433   "\ufb85\060\ufb85\063\ufb85\064\ufb85\065\ufb85\102\ufb85\111\uffb85" +
434   "\113\ufb85\114\ufb85\001\002\000\004\037\010\0161\001\002\000" +
435   "\004\042\0162\001\002\000\004\003\037\053\024\054\036" +
436   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
437   "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
438   "\120\017\001\002\000\030\043\016\053\024\054\036\057" +
439   "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
440   "\114\016\001\002\000\032\043\ufb83\053\ufb83\054\uffb83\057" +
441   "\ufb83\060\ufb83\063\ufb83\064\ufb83\065\ufb83\102\ufb83\111\uffb83" +
442   "\113\ufb83\114\ufb83\001\002\000\004\037\010\0166\001\002\000" +
443   "\004\042\0167\001\002\000\004\003\037\053\024\054\036" +
444   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
445   "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
446   "\120\017\001\002\000\030\043\0169\053\024\054\036\057" +

```

```

444   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
445   "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
446   "\120\017\001\002\000\030\043\0169\053\024\054\036\057" +
447   "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
448   "\114\016\001\002\000\032\043\ufb82\053\ufb82\054\uffb82\057" +
449   "\uffb82\060\uffb82\063\uffb82\064\uffb82\065\uffb82\102\uffb82\111\uffb82" +
450   "\113\uffb82\114\uffb82\001\002\000\024\021\018c\022\018a\023" +
451   "\018b\024\0184\025\0185\026\0189\032\0187\033\0188\0107\0186" +
452   "\001\002\000\024\021\0174\022\0172\023\0173\024\016c\025" +
453   "\016d\026\0171\032\016e\033\0170\017\016f\001\002\000\004" +
454   "\114\0182\001\002\000\004\114\0180\001\002\000\004\0107" +
455   "\017\001\002\000\032\043\uffdb\053\uffdb\054\uffdb\057\ufdb" +
456   "\060\ufdb\063\ufdb\064\ufdb\065\ufdb\102\ufdb\111\uffdb\113" +
457   "\uffdb\114\ufdb\001\002\000\004\107\017\001\002\000\004" +
458   "\114\017b\001\002\000\004\114\0179\001\002\000\004\114" +
459   "\0177\001\002\000\004\114\0175\001\002\000\004\0107\0176" +
460   "\001\002\000\032\043\ufccf\053\ufccf\054\uffccf\057\ufccf\060" +
461   "\ufccf\063\ufccf\064\ufccf\065\ufccf\102\ufccf\111\uffccf\113\uffccf" +
462   "\114\uffccf\001\002\000\004\107\0178\001\002\000\032\043" +
463   "\uffd0\053\uffd0\054\ufffd\057\uffd0\060\uffd0\063\ufffd\064\ufffd0" +
464   "\065\uffd0\102\uffd0\111\ufffd0\113\ufffd0\114\ufffd0\001\002\000" +
465   "\004\107\017\001\002\000\032\043\uffd1\053\uffd1\054\uffd1" +
466   "\057\uffd1\060\uffd1\063\uffd1\064\uffd1\065\uffd1\102\uffd1\111" +
467   "\uffd1\113\uffd1\114\uffd1\001\002\000\022\004\0130\005\0133" +
468   "\006\012c\007\012e\001\002\001\0134\022\0131\023\0132\0107\017d\001" +
469   "\002\000\032\043\ufffac\053\ufffac\054\ufffac\057\ufffac\060\ufffac" +
470   "\063\ufffac\064\ufffac\065\ufffac\102\ufffac\111\ufffac\113\ufffac\114" +
471   "\ufffac\001\002\000\032\043\ufd9\053\ufd9\054\uffd9\057\ufd9" +
472   "\060\ufd9\063\ufd9\064\ufd9\065\ufd9\102\ufd9\111\uffd9\113" +
473   "\ufd9\114\ufd9\001\002\000\032\043\ufd7\053\uffd7\054\ufd7" +
474   "\057\uffd7\060\uffd7\063\uffd7\064\uffd7\065\uffd7\102\uffd7\111" +
475   "\uffd7\113\uffd7\114\uffd7\001\002\000\032\043\uffd8\053\uffd8" +
476   "\054\uffd8\057\uffd8\060\ufd8\063\ufd8\064\uffd8\065\uffd8\102" +
477   "\uffd8\111\uffd8\113\uffd8\114\uffd8\001\002\000\004\0107\0181" +
478   "\001\002\000\032\043\uffda\053\uffda\054\uffda\057\uffda\060" +
479   "\uffda\063\uffda\064\uffda\065\uffda\102\uffda\111\uffda\113\uffda" +

```



MANUAL DE PRÁCTICAS



```

476   "\u054\uuffd8\057\uffd8\060\uffd8\063\uffd8\064\uffd8\065\uffd8\102" +
477   "\uffd8\111\uffd8\113\uffd8\114\uffd8\001\002\000\004\107\u0181" +
478   "\001\002\000\032\043\uffda\053\uffda\054\uffda\057\uffda\060" +
479   "\uffda\063\uffda\064\uffda\065\uffda\102\uffda\111\uffda\113\uffda" +
480   "\114\uffda\001\002\000\004\107\u0183\001\002\000\032\043" +
481   "\uffce\053\uffce\054\uffce\057\uffce\060\uffce\063\uffce\064\uffce" +
482   "\065\uffce\102\uffce\111\uffce\114\uffce\001\002\000" +
483   "\004\114\u0199\001\002\000\004\114\u0197\001\002\000\032" +
484   "\043\uffb\053\uffb\054\uffb\057\uffb\060\uffb\063\uffb\064" +
485   "\uffb\065\uffb\102\uffb\111\uffb\113\uffb\114\uffb\001\002" +
486   "\000\004\107\u0196\001\002\000\004\107\u0195\001\002\000" +
487   "\004\114\u0193\001\002\000\004\114\u0191\001\002\000\004" +
488   "\114\u018f\001\002\000\004\114\u018d\001\002\000\004\107" +
489   "\u018e\001\002\000\032\043\uffb\053\uffb\054\uffb\057\uffb\063" +
490   "\060\uffb\063\uffb\064\uffb\065\uffb\062\uffb\062\uffb\061\uffb\063" +
491   "\uffb\063\114\uffb\01\002\000\004\107\u0190\001\002\000\032" +
492   "\043\uffb\053\uffb\054\uffb\057\uffb\060\uffb\063\uffb\064" +
493   "\uffb\065\uffb\102\uffb\111\uffb\113\uffb\114\uffb\001\002" +
494   "\000\004\107\u0192\001\002\000\032\043\uffb\053\uffb\054" +
495   "\uffb\057\uffb\063\uffb\060\uffb\063\uffb\064\uffb\065\uffb\062\uffb\062" +
496   "\111\uffb\051\uffb\051\uffb\051\uffb\001\002\000\004\107\u0194\001" +
497   "\002\000\032\043\uffb\053\uffb\054\uffb\057\uffb\060\uffb\064" +
498   "\063\uffb\064\uffb\065\uffb\062\uffb\111\uffb\113\uffb\114" +
499   "\uffb\001\002\000\032\043\uffb\053\uffb\054\uffb\057\uffb\063" +
500   "\060\uffb\063\uffb\064\uffb\065\uffb\062\uffb\062\uffb\061\uffb\063" +
501   "\uffb\063\114\uffb\01\002\000\032\043\uffb\053\uffb\054\uffb\064" +
502   "\057\uffb\060\uffb\063\uffb\064\uffb\065\uffb\062\uffb\062\uffb\061" +
503   "\uffb\063\113\uffb\01\002\000\004\107\u0198\001\002" +
504   "\000\032\043\uffb\053\uffb\054\uffb\057\uffb\060\uffb\063" +
505   "\uffb\064\065\uffb\065\uffb\062\uffb\061\uffb\062\uffb\061\uffb\063" +
506   "\001\002\000\004\107\u019a\001\002\000\032\043\uffb\053" +
507   "\uffb\054\uffb\057\uffb\060\uffb\063\uffb\062\uffb\064\uffb\065\uffb\062" +
508   "\102\uffb\02\111\uffb\02\113\uffb\02\114\uffb\02\001\002\000\004\116" +
509   "\240\001\002\000\004\002\000\001\002\000\004\113\u019e" +
510   "\001\002\000\004\037\u019e\001\002\000\004\107\u01a0\001" +
511   "\002\000\032\043\uffe7\053\uffe7\054\uffe7\057\uffe7\060\uffe7" +
512   "\063\uffe7\064\uffe7\065\uffe7\102\uffe7\111\uffe7\113\uffe7\114" +
513   "\uffe7\001\002\000\006\020\001\012\113\127\001\002\000\004" +
514   "\113\127\001\002\000\016\016\016\017\016\015\037\001a\001" +
515   "\002\000\004\042\01f9\001\002\000\004\113\127\001\002" +
516   "\000\004\113\127\001\002\000\004\037\01a\001\002\000" +
517   "\042\01a9\001\002\000\044\003\037\053\024\054\036" +
518   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
519   "\015\010\042\013\045\111\031\113\026\114\016\117\043" +
520   "\120\017\001\002\000\030\043\01a\053\024\054\036\057" +
521   "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
522   "\114\016\001\002\000\034\043\0fa\053\0fa\054\0fa\057" +
523   "\uffa\060\uffa\061\01a\063\0fa\064\0fa\065\uffa\061\02\uffa\06" +
524   "\113\0ffa\061\013\0ffa\061\014\0ffa\061\001\002\000\006\042\01a\060" +
525   "\u01ae\001\002\000\044\003\037\053\024\054\036\057\033" +
526   "\060\014\063\044\064\025\065\034\077\023\100\015\01" +
527   "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
528   "\001\002\000\036\01a\001\002\000\004\113\127\001" +
529   "\002\000\010\016\01b\03\017\01a\0b\2\113\127\001\002\000\004" +
530   "\037\01a\001\002\000\004\113\127\001\002\000\004\113" +
531   "\127\001\002\000\004\037\01b\005\001\002\000\004\042\01b\06" +
532   "\001\002\000\044\003\037\053\024\054\036\057\033\060" +
533   "\014\063\044\064\025\065\034\077\023\100\015\01\042" +
534   "\103\045\111\031\113\026\114\016\117\043\120\017\001" +
535   "\002\000\030\043\01b\053\024\054\036\057\033\060\014" +
536   "\063\044\064\025\065\034\111\031\113\026\114\016\001" +
537   "\002\000\004\061\01b\09\001\002\000\004\042\01b\001\002" +
538   "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
539   "\044\064\025\065\034\077\023\100\015\01\042\103\045" +
540   "\111\031\113\026\114\016\117\043\120\017\001" +
541   "\030\043\01b\053\024\054\036\057\033\060\014\063\044" +
542   "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
543   "\032\043\0ff8\053\0ff8\054\0ff8\057\0ff8\060\0ff8\063\0ff8" +
544   "\001\002\000\004\113\121\001\002\000\004\114\121\001\002" +

```

```

509   "\240\001\002\000\004\002\000\001\002\000\004\113\019e" +
510   "\001\002\000\004\037\019e\001\002\000\004\107\01a0\001" +
511   "\002\000\032\043\uffe7\053\uffe7\054\uffe7\057\uffe7\060\uffe7" +
512   "\063\uffe7\064\uffe7\065\uffe7\102\uffe7\111\uffe7\113\uffe7\114" +
513   "\uffe7\001\002\000\006\020\001\012\113\127\001\002\000\004" +
514   "\113\127\001\002\000\016\016\016\017\016\015\037\001a\001" +
515   "\002\000\004\042\01f9\001\002\000\004\113\127\001\002" +
516   "\000\004\113\127\001\002\000\004\037\01a\001\002\000" +
517   "\042\01a9\001\002\000\044\003\037\053\024\054\036" +
518   "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
519   "\015\010\042\013\045\111\031\113\026\114\016\117\043" +
520   "\120\017\001\002\000\030\043\01a\053\024\054\036\057" +
521   "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
522   "\114\016\001\002\000\034\043\0fa\053\0fa\054\0fa\057" +
523   "\uffa\060\uffa\061\01a\063\0fa\064\0fa\065\uffa\061\02\uffa\06" +
524   "\113\0ffa\061\013\0ffa\061\014\0ffa\061\001\002\000\006\042\01a\060" +
525   "\u01ae\001\002\000\044\003\037\053\024\054\036\057\033" +
526   "\060\014\063\044\064\025\065\034\077\023\100\015\01" +
527   "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
528   "\001\002\000\036\01a\001\002\000\004\113\127\001" +
529   "\002\000\010\016\01b\03\017\01a\0b\2\113\127\001\002\000\004" +
530   "\037\01a\001\002\000\004\113\127\001\002\000\004\113" +
531   "\127\001\002\000\004\037\01b\005\001\002\000\004\042\01b\06" +
532   "\001\002\000\044\003\037\053\024\054\036\057\033\060" +
533   "\014\063\044\064\025\065\034\077\023\100\015\01\042" +
534   "\103\045\111\031\113\026\114\016\117\043\120\017\001" +
535   "\002\000\030\043\01b\053\024\054\036\057\033\060\014" +
536   "\063\044\064\025\065\034\111\031\113\026\114\016\001" +
537   "\002\000\004\061\01b\09\001\002\000\004\042\01b\001\002" +
538   "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
539   "\044\064\025\065\034\077\023\100\015\01\042\103\045" +
540   "\111\031\113\026\114\016\117\043\120\017\001" +
541   "\030\043\01b\053\024\054\036\057\033\060\014\063\044" +
542   "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
543   "\032\043\0ff8\053\0ff8\054\0ff8\057\0ff8\060\0ff8\063\0ff8" +
544   "\001\002\000\004\113\121\001\002\000\004\114\121\001\002" +

```



MANUAL DE PRÁCTICAS



545 "002\000\004\037\u01be\001\002\000\004\042\u01bf\001\002" +
546 "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
547 "\044\064\025\065\034\077\023\100\015\101\042\103\045" +
548 "\11\031\113\026\114\016\117\043\120\017\001\002\000" +
549 "\030\043\01e1\053\024\054\036\057\033\060\014\063\044" +
550 "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
551 "\004\061\01e2\001\002\000\004\042\01e3\001\002\000\044" +
552 "\003\037\053\024\054\036\057\033\060\014\063\044\064" +
553 "\025\065\034\077\023\100\015\101\042\103\045\111\031" +
554 "\113\026\114\016\117\043\120\017\001\002\000\030\043" +
555 "\01e1\053\024\054\036\057\033\060\014\063\044\064\025" +
556 "\065\034\111\031\113\026\114\016\001\002\000\032\043" +
557 "\uf8c\053\024\054\036\057\033\060\014\063\044\064\064" +
558 "\065\uf8c\102\uf8c\111\uf8c\113\uf8c\114\uf8c\001\002\000" +
559 "\004\042\01e7\001\002\000\043\003\037\053\024\054\036" +
560 "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
561 "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
562 "\120\017\001\002\000\030\043\01e9\053\024\054\036\057" +
563 "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
564 "\114\016\001\002\000\004\061\01ea\001\002\000\004\042" +
565 "\01ch\001\002\000\004\037\053\024\054\036\057\033" +
566 "\060\014\063\044\064\025\065\034\077\023\100\015\101" +
567 "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
568 "\001\002\000\030\043\01cd\053\024\054\036\057\033\060" +
569 "\014\063\044\064\025\065\034\111\031\113\026\114\016" +
570 "\001\002\000\032\043\uf8a\053\uf8a\054\uf8a\057\uf8a\060" +
571 "\uff8a\063\uf8a\064\uf8a\065\uf8a\102\uf8a\111\uf8a\113\uf8a" +
572 "\114\uf8a\001\002\000\030\043\01e9\053\024\054\036\057" +
573 "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
574 "\114\016\001\002\000\032\043\uf92\053\uf92\054\uf92\057" +
575 "\uf92\060\uf92\063\uf92\064\uf92\065\uf92\102\uf92\111\uf92" +
576 "\113\uf92\114\uf92\001\002\000\004\037\053\024\054\036" +
577 "\004\042\01d2\001\002\000\044\003\037\053\024\054\036\057" +
578 "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
579 "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
580 "\120\017\001\002\000\030\043\01d4\053\024\054\036\057" +

578 "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
579 "\015\101\042\103\045\111\031\113\026\114\016\117\043" +
580 "\120\017\001\002\000\030\043\01d4\053\024\054\036\057" +
581 "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
582 "\114\016\001\002\000\034\043\ufa7\053\ufa7\054\ufa7\057" +
583 "\uffa7\060\uffa7\061\01d5\063\uffa7\064\uffa7\065\uffa7\102\uffa7" +
584 "\111\uffa7\113\uffa7\114\uffa7\001\002\000\006\042\01d6\060" +
585 "\u01d7\001\002\000\044\003\037\053\024\054\036\057\033" +
586 "\060\014\063\044\064\025\065\034\077\023\100\015\101" +
587 "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
588 "\001\002\000\004\036\01d8\001\002\000\004\013\127\001" +
589 "\002\000\010\016\01dce\017\01d8\113\127\001\002\000\004" +
590 "\037\001\001\002\000\004\113\127\001\002\000\004\113" +
591 "\127\001\002\000\004\037\01d8\001\002\000\004\042\01d6" +
592 "\001\002\000\004\044\003\037\053\024\054\036\057\033\060" +
593 "\014\063\044\064\025\065\034\077\023\100\015\101\042" +
594 "\103\045\111\031\113\026\114\016\117\043\120\017\001" +
595 "\002\000\030\043\01e1\053\024\054\036\057\033\060\014\063\044" +
596 "\063\044\064\025\065\034\111\031\113\026\114\016\001" +
597 "\002\000\004\061\01e2\001\002\000\004\042\01e3\001\002" +
598 "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
599 "\044\064\025\065\034\077\023\100\015\101\042\103\045" +
600 "\111\031\113\026\114\016\117\043\120\017\001\002\000" +
601 "\030\043\01e5\053\024\054\036\057\033\060\014\063\044" +
602 "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
603 "\032\043\uf8b\053\uf8b\054\uf8b\057\uf8b\060\uff8b\063\uff8b" +
604 "\064\uf8b\065\uf8b\102\uf8b\111\uf8b\113\uf8b\114\uf8b\001" +
605 "\002\000\004\037\01e1\001\002\000\004\042\01e8\001\002" +
606 "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
607 "\044\064\025\065\034\077\023\100\015\101\042\103\045" +
608 "\111\031\113\026\114\016\117\043\120\017\001\002\000" +
609 "\030\043\01e5\053\024\054\036\057\033\060\014\063\044" +
610 "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
611 "\004\061\01eb\001\002\000\004\042\01ec\001\002\000\044" +
612 "\003\037\053\024\054\036\057\033\060\014\063\044\064" +
613 "\025\065\034\077\023\100\015\101\042\103\045\111\031" +



MANUAL DE PRÁCTICAS



```

614      "\u113\026\u114\016\117\043\120\017\001\002\000\030\043" +
615      "\u01ee\053\024\054\036\057\033\060\014\063\044\064\025" +
616      "\065\034\111\031\113\026\114\016\001\002\000\032\043" +
617      "\uff8d\054\0ff8d\057\0ff8d\060\0ff8d\063\0ff8d\064\0ff8d" +
618      "\065\0ff8d\102\0ff8d\111\0ff8d\113\0ff8d\114\0ff8d\001\002\000" +
619      "\004\042\0u1e0\001\002\000\044\003\037\053\024\054\036" +
620      "\057\033\060\014\063\044\064\025\065\034\077\023\100" +
621      "\015\010\042\103\045\111\031\113\026\114\016\117\043" +
622      "\120\017\001\002\000\030\043\0u1f2\053\024\054\036\057" +
623      "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
624      "\114\016\001\002\000\004\061\0u1f3\001\002\000\004\042" +
625      "\001\001\002\000\044\003\037\053\024\054\036\057\033" +
626      "\060\014\063\044\064\025\065\034\077\023\100\015\101" +
627      "\042\103\045\111\031\113\026\114\016\117\043\120\017" +
628      "\001\002\000\030\043\0u1f6\053\024\054\036\057\030\060" +
629      "\014\063\044\064\025\065\034\111\031\113\026\114\016" +
630      "\001\002\000\032\043\0uf89\053\0uf89\054\0uf89\057\0uf89\060" +
631      "\0uf89\063\0uf89\064\0uf89\065\0uf89\102\0uf89\111\0uf89\113\0uf89" +
632      "\114\0uf89\001\002\000\030\043\0u01f8\053\024\054\036\057" +
633      "\033\060\014\063\044\064\025\065\034\111\031\113\026" +
634      "\114\016\001\002\000\032\043\0uff91\053\0uff91\054\0uff91\057" +
635      "\0uff91\060\0uff91\063\0uff91\064\0uff91\065\0uff91\102\0uff91\111\0uff91" +
636      "\113\0uff91\114\0uff91\001\002\000\044\003\037\053\024\054" +
637      "\036\057\033\060\014\063\044\064\025\065\034\077\023" +
638      "\100\015\101\042\103\045\111\031\113\026\114\016\117" +
639      "\043\120\017\001\002\000\030\043\0u01f8\053\024\054\036" +
640      "\057\033\060\014\063\044\064\025\065\034\111\031\113" +
641      "\026\114\016\001\002\000\034\043\0uffa8\053\0uffa8\054\0uffa8" +
642      "\057\0uffa8\060\0uffa8\061\0u01fc\063\0uffa8\064\0uffa8\065\0uffa8\102" +
643      "\0uffa8\111\0uffa8\113\0uffa8\114\0uffa8\001\002\000\006\042\0u01fd" +
644      "\060\0u01fe\001\002\000\044\003\037\053\024\054\036\057" +
645      "\033\060\014\063\044\064\025\065\034\077\023\100\015" +
646      "\101\042\103\045\111\031\113\026\114\016\117\043\120" +
647      "\017\001\002\000\004\036\0u01ff\001\002\000\004\113\127" +
648      "\001\002\000\010\016\0u203\017\0u0201\037\0u0201\001\002\000" +
649      "\004\113\127\001\002\000\004\042\0u020d\001\002\000\004" +

```

```

675      "\032\043\0uf8f\053\0uff8f\054\0uf8f\057\0uff8f\060\0uf8f\063\0uff8f" +
676      "\064\0uff8f\065\0uff8f\102\0uff8f\111\0uff8f\113\0uff8f\114\0uff8f\001" +
677      "\002\000\004\037\0u0215\001\002\000\004\042\0u0216\001\002" +
678      "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
679      "\044\064\025\065\034\077\023\100\015\101\042\103\045" +
680      "\111\031\113\026\114\016\117\043\120\017\001\002\000" +
681      "\030\043\0u0218\053\024\054\036\057\033\060\014\063\044" +
682      "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
683      "\004\061\0u0219\0001\002\000\004\042\0u021a\001\002\000\044" +
684      "\003\037\053\024\054\036\057\033\060\014\063\044\064" +
685      "\025\065\034\077\023\100\015\101\042\103\045\111\031" +
686      "\113\026\114\016\117\043\120\017\001\002\000\030\043" +
687      "\0u021c\053\024\054\036\057\033\060\014\063\044\064\025" +
688      "\065\034\111\031\113\026\114\016\001\002\000\032\043" +
689      "\0uff87\053\0uff87\054\0uff87\057\0uff87\060\0uff87\063\0uff87\064\0uff87" +
690      "\065\0uff87\102\0uff87\111\0uff87\113\0uff87\114\0uff87\001\002\000" +
691      "\030\043\0u021e\053\024\054\036\057\033\060\014\063\044" +
692      "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
693      "\032\043\0uf93\053\0uff93\054\0uf93\057\0uff93\060\0uf93\063\0uff93" +
694      "\064\0uff93\065\0uf93\102\0uff93\111\0uff93\113\0uff93\114\0uff93\001" +
695      "\002\000\004\037\0u0220\001\002\000\004\042\0u0221\001\002" +
696      "\000\044\003\037\053\024\054\036\057\033\060\014\063" +
697      "\044\064\025\065\034\077\023\100\015\101\042\103\045" +
698      "\111\031\113\026\114\016\117\043\120\017\001\002\000" +
699      "\030\043\0u0223\053\024\054\036\057\033\060\014\063\044" +
700      "\064\025\065\034\111\031\113\026\114\016\001\002\000" +
701      "\034\043\0uffa5\053\0uffa5\054\0uffa5\057\0uffa5\060\0uffa5\061\0u0224" +
702      "\063\0uffa5\064\0uffa5\065\0uffa5\102\0uffa5\111\0uffa5\113\0uffa5\114" +
703      "\0uffa5\001\002\000\006\042\0u0225\060\0u0226\001\002\000\044" +
704      "\003\037\053\024\054\036\057\033\060\014\063\044\064" +
705      "\025\065\034\077\023\100\015\101\042\103\045\111\031" +
706      "\113\026\114\016\117\043\120\017\001\002\000\004\036" +
707      "\0u0227\001\002\000\004\020\0u0228\001\002\000\004\113\127" +
708      "\001\002\000\004\037\0u022a\001\002\000\004\042\0u022b\001" +
709      "\002\000\004\003\037\053\024\054\036\057\033\060\014" +
710      "\063\044\064\025\065\034\077\023\100\015\101\042\103" +

```



```
708 " \001\002\000\004\037\u022a\001\002\000\004\042\u022b\001" +
709 " \002\000\044\003\037\053\024\054\036\057\033\060\014" +
710 " \063\044\064\025\065\034\077\023\100\015\010\042\0103" +
711 " \045\111\031\113\026\114\016\117\043\120\017\001\002" +
712 " \000\030\043\0022d\053\024\054\036\057\033\060\014\063" +
713 " \044\064\025\065\034\111\031\113\026\114\016\001\002" +
714 " \000\004\061\0022e\001\002\000\004\042\0022f\001\002\000" +
715 " \044\037\053\024\054\036\057\033\060\014\063\044" +
716 " \064\025\065\034\077\023\100\015\101\042\013\045\111" +
717 " \031\113\026\114\016\117\043\120\017\001\002\000\030" +
718 " \043\003\053\024\054\036\057\033\060\014\063\044\064" +
719 " \025\065\034\111\031\113\026\114\016\001\002\000\032" +
720 " \043\ufe86\053\ufe86\054\ufe86\057\ufe86\060\ufe86\063\ufe86\064" +
721 " \ufe86\065\ufe86\102\ufe86\111\ufe86\113\ufe86\114\ufe86\001\002" +
722 " \000\030\043\0023\053\024\054\036\057\033\060\014\063" +
723 " \044\064\025\065\034\111\031\113\026\114\016\001\002" +
724 " \000\032\043\uff90\053\ufe90\054\ufe90\057\ufe90\060\ufe90\063" +
725 " \ufe90\064\ufe90\065\ufe90\102\ufe90\111\ufe90\113\ufe90\114\ufe90" +
726 " \001\002\000\004\002\001\001\002\000\004\036\023b\001" +
727 " \002\000\004\037\023\001\002\000\004\042\0023b\001\002" +
728 " \000\044\003\037\053\024\054\036\057\033\060\014\063" +
729 " \044\064\025\065\034\077\023\100\015\101\042\013\045" +
730 " \111\031\113\026\114\016\117\043\120\017\001\002\000" +
731 " \030\043\0023a\053\024\054\036\057\033\060\014\063\044" +
732 " \064\025\065\034\111\031\113\026\114\016\001\002\000" +
733 " \004\002\uffff\001\002\000\004\037\023c\001\002\000\004" +
734 " \042\0023d\001\002\000\043\003\037\053\024\054\036\057" +
735 " \033\060\014\063\044\064\025\065\034\077\023\100\015" +
736 " \101\042\103\045\111\031\113\026\114\016\117\043\120" +
737 " \017\001\002\000\004\002\001\003\043\0023f\053\024\054\036\057\033" +
738 " \060\014\063\044\064\025\065\034\111\031\113\026\114" +
739 " \016\001\002\000\004\002\001\000\002\001\000\004\002\001" +
740 " \001\002" );
741
742  /* Access to parse-action table. */
743  public short[][] action_table() {return action_table;}
```

```
745  /** <code>reduce_goto</code> table. */
746 protected static final short[][] reduce_table =
747     unpackFromStrings(new String[] {
748         "\000\023e\000\004\002\005\001\001\000\002\001\001\000" +
749         "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
750         "\001\001\000\002\001\001\000\002\001\001\000\024\003" +
751         "\020\004\021\006\026\007\027\010\017\011\034\012\040" +
752         "\015\031\020\037\001\001\000\002\001\001\000\002\001" +
753         "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
754         "\000\020\004\066\006\067\007\070\010\065\011\071\012" +
755         "\074\020\073\001\001\000\002\001\001\000\002\001\001" +
756         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
757         "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
758         "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
759         "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
760         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
761         "\002\001\001\000\002\001\001\000\002\001\001\000\004" +
762         "\016\047\001\001\000\002\001\001\000\004\016\057\001" +
763         "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
764         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
765         "\002\001\001\000\002\001\001\000\026\003\062\004\021" +
766         "\006\026\007\027\010\017\011\034\012\040\015\031\017" +
767         "\063\020\037\001\001\000\020\004\066\066\067\007\070" +
768         "\010\065\011\071\012\074\020\073\001\001\000\002\001" +
769         "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
770         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
771         "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
772         "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
773         "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
774         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
775         "\002\001\001\000\002\001\001\000\004\014\112\001\001" +
776         "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
777         "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
778         "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
779         "\001\000\004\013\125\001\001\000\002\001\001\000\002"
```



```

793 "002\001\001\000\004\013\202\001\001\000\002\001\001" +
794 "\000\004\005\204\001\001\000\002\001\001\000\002\001" +
795 "\001\000\004\013\207\001\001\000\002\001\001\000\004" +
796 "\005\211\001\001\000\002\001\001\000\002\001\001\000" +
797 "\004\013\214\001\001\000\002\001\001\000\004\005\216" +
798 "\001\000\002\001\001\000\002\001\001\000\002\001\001\003" +
799 "\221\001\001\000\002\001\001\000\004\005\223\001\001" +
800 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
801 "\04\013\227\001\001\000\002\001\001\000\004\005\231" +
802 "\001\001\000\002\001\001\000\002\001\001\000\004\003" +
803 "\234\004\021\006\026\007\027\010\017\011\034\012\040" +
804 "\015\031\020\037\001\001\000\002\004\066\006\067\007" +
805 "\070\010\065\011\071\012\074\020\073\001\001\000\002" +
806 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
807 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
808 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
809 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
810 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
811 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
812 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
813 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
814 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
815 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
816 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
817 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
818 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
819 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
820 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
821 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
822 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
823 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
824 "\004\021\006\026\007\027\010\017\011\034\012\040\015" +
825 "\031\020\037\001\001\000\020\004\066\006\067\007\070" +
826 "\010\065\011\071\012\074\020\073\001\001\000\002\001" +
827 "\001\000\002\001\001\000\004\013\35\001\001\000\004" +
828 "\013\350\001\001\000\002\001\001\000\002\001\001\000" +

```

```

854 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
855 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
856 "\001\001\000\002\002\001\001\000\002\002\001\001\001" +
857 "\001\000\002\002\001\001\000\002\001\001\000\002\001" +
858 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
859 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
860 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
861 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
862 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
863 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
864 "\001\001\000\002\001\001\000\004\013\015\001\001\000" +
865 "\004\013\013\016\001\001\000\002\001\001\000\004\013\015\001" +
866 "\001\001\000\002\001\001\000\004\013\015\001\001\000" +
867 "\002\001\001\000\002\001\001\000\024\003\015\004\021" +
868 "\006\026\007\027\010\017\011\034\012\040\015\031\020" +
869 "\037\001\001\000\020\004\066\006\067\007\070\010\065" +
870 "\011\071\012\074\020\073\001\001\000\002\001\001\000" +
871 "\024\003\015\021\006\026\007\027\010\017\011\034" +
872 "\012\040\015\031\020\037\001\001\000\020\004\066\006" +
873 "\067\007\070\010\065\011\071\012\074\020\073\001\001" +
874 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
875 "\024\003\016\021\006\026\007\027\010\017\011\034" +
876 "\012\040\015\031\020\037\001\001\000\020\004\066\006" +
877 "\067\007\070\010\065\011\071\012\074\020\073\001\001" +
878 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
879 "\024\003\016\021\006\026\007\027\010\017\011\034" +
880 "\012\040\015\031\020\037\001\001\000\020\004\066\006" +
881 "\067\007\070\010\065\011\071\012\074\020\073\001\001" +
882 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
883 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
884 "\001\001\000\002\001\001\000\004\020\017\001\001\000" +
885 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
886 "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
887 "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
888 "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
889 "\002\001\001\000\002\001\001\000\002\001\001\000\002" +

```



MANUAL DE PRÁCTICAS



"\001\000\002\001\001\000\002\001\001\000\002\001\001" +
"\000\002\001\001\000\001\000\002\001\001\000\002\001\001\000" +
"\002\001\001\000\001\000\002\001\001\000\002\001\001\000\002" +
"\001\001\000\001\000\002\001\001\000\002\001\001\000\002\001" +
"\001\000\002\001\001\000\001\000\004\13\01a2\01\001\001\000\004" +
"\013\02a1\001\001\000\002\001\001\000\002\001\001\000\001" +
"\004\013\01c5\01f\001\001\000\004\13\01a6\01\001\001\000\002" +
"\001\001\000\002\001\001\000\004\24\003\03u1a9\004\021\001\006" +
"\026\007\027\010\017\011\034\012\040\015\031\020\037" +
"\001\001\000\002\000\020\004\066\006\067\007\070\010\065\011" +
"\071\012\024\070\020\073\001\001\000\002\001\001\000\002" +
"\001\001\000\002\004\033\001\003\01u0d\004\021\006\066\027\072\010" +
"\017\011\034\012\040\015\031\020\037\001\001\000\002" +
"\001\001\000\004\13\01a1\001\000\004\13\01a1b\001" +
"\001\000\002\001\001\000\004\13\01a1c\001\001\000\004" +
"\013\02a1\001\001\000\002\001\001\000\002\001\001\000\001" +
"\024\003\01b6\004\021\006\026\007\027\010\017\011\034" +
"\012\040\015\031\020\037\001\001\000\020\004\066\006" +
"\067\007\070\010\065\011\071\012\074\020\073\001\001" +
"\000\002\001\001\000\002\001\001\000\004\24\003\03u1a0\004" +
"\021\006\026\007\027\010\017\011\034\012\040\015\031" +
"\020\037\001\001\000\002\000\020\004\066\006\067\007\070\010" +
"\065\011\071\012\074\020\073\001\000\002\001\001" +
"\000\002\001\001\000\002\001\001\000\004\24\003\03u1b\004" +
"\021\006\026\007\027\010\017\011\034\012\040\015\031" +
"\020\037\001\001\000\002\000\020\004\066\006\067\007\070\010" +
"\065\011\071\012\074\020\073\001\000\002\001\001" +
"\000\002\001\001\000\002\001\001\000\004\24\003\03u1b\004" +
"\005\011\071\012\074\020\073\001\000\002\001\001" +
"\000\002\001\001\000\002\004\033\01c3\04\021\006\026\007" +
"\027\010\017\011\034\012\040\015\031\020\037\001\001" +
"\000\020\004\066\006\067\007\070\010\065\011\071\012" +
"\074\020\073\001\001\000\002\001\001\000\002\001\001" +
"\000\024\003\01c7\004\021\006\026\007\027\010\017\011" +
"\034\012\024\040\015\031\020\037\001\001\000\002\004\066" +
"\006\067\007\070\010\065\011\071\012\074\020\073\001" +
"\001\000\002\001\000\001\000\002\001\001\000\004\24\003\03u1b" +
"\004\021\006\026\007\027\010\017\011\034\012\040\015"

"\"001\000\002\001\001\000\024\03v\01d2\04\021\006\026" +
"\007\027\010\017\011\034\012\040\015\031\020\037\01" /"
"\001\000\020\004\066\006\067\007\070\010\065\011\071" /"
"\012\074\020\073\001\001\000\002\001\001\000\021\001" /"
"\001\004\024\003\01f6\004\021\006\026\027\027\010\017" +
"\011\034\012\040\015\021\020\037\001\001\000\022\001" /"
"\001\000\004\013\01b8\001\001\000\004\013\01a9\001\001" +
"\000\002\001\001\000\004\013\01e5\001\001\000\004\013" +
"\001\0d\001\001\000\002\001\001\000\002\001\001\000\024" +
"\003\001\0d\004\021\006\026\027\027\010\017\011\034\012" +
"\040\015\031\020\037\001\001\000\002\000\046\066\067" +
"\007\070\010\065\011\071\021\074\020\073\001\001\000" +
"\002\001\001\000\002\001\001\000\002\004\023\003\01e3\004\021" +
"\006\026\007\027\010\017\011\034\012\040\015\031\020" +
"\037\001\001\000\002\004\066\006\067\007\070\010\065" +
"\011\071\012\074\020\073\001\001\000\002\001\001\000" +
"\002\001\001\000\002\001\001\000\002\004\023\003\01e8\004\021" +
"\006\026\007\027\010\017\011\034\012\040\015\031\020" +
"\037\001\001\000\002\004\066\006\067\007\070\010\065" +
"\011\071\012\074\020\073\001\001\000\002\001\001\000" +
"\002\001\001\000\004\023\003\01e5\004\021\006\026\027\027" +
"\010\017\011\034\012\040\015\031\020\037\001\001\000" +
"\020\004\066\006\067\007\070\010\065\011\071\012\074" +
"\020\073\001\001\000\002\001\001\000\002\001\001\000" +
"\024\003\01f6\004\021\006\026\027\027\010\017\011\034" +
"\012\040\015\031\020\037\001\001\000\002\000\046\066" +
"\067\007\070\010\065\011\071\012\074\020\073\001\001" +
"\000\002\001\001\000\002\001\001\000\002\004\023\003\01f4\004" +
"\021\006\026\007\027\010\017\011\034\012\040\015\031" +
"\020\037\001\001\000\002\001\001\000\002\001\001\000" +
"\065\011\071\012\074\020\073\001\001\000\002\001\001" +
"\000\020\004\066\006\067\007\070\010\065\011\071\012" +
"\074\020\073\001\001\000\002\001\001\000\002\004\023\003\01f9" +
"\004\021\006\026\007\027\010\017\011\034\012\040\015" +
"\031\020\037\001\001\000\002\004\066\006\067\007\070" +
"\010\065\011\071\012\074\020\073\001\001\000\002\001\001" +



```
972 "007\027\010\017\011\034\012\040\015\031\020\037\001" +
973 "\001\000\002\001\001\000\004\013\w0213\001\001\000\002\001\001\000" +
974 "\001\001\000\004\013\w023\001\001\000\000\002\001\001\001" +
975 "\004\013\w023\001\001\000\000\002\001\001\000\002\001\001" +
976 "\000\024\003\w0206\004\021\006\026\007\027\010\017\011" +
977 "\034\012\040\015\031\020\037\001\001\000\020\004\066" +
978 "\006\067\007\070\010\065\011\071\012\074\020\073\001" +
979 "\001\000\002\001\001\000\002\001\001\000\024\003\w020a" +
980 "\004\021\006\026\007\027\010\017\011\034\012\040\015" +
981 "\031\w020\037\001\001\000\000\004\066\006\067\007\070" +
982 "\010\065\011\071\012\074\020\073\001\001\000\002\001" +
983 "\001\000\024\003\w020d\004\021\006\026\007\027\010\017" +
984 "\011\012\040\015\031\020\037\001\001\000\020\004\044" +
985 "\066\006\067\007\070\010\065\011\071\012\074\020\073" +
986 "\001\001\000\002\001\001\000\002\001\001\000\024\003" +
987 "\021\004\021\006\026\007\027\010\017\011\034\012\040" +
988 "\015\031\w020\037\001\001\000\000\004\066\006\067\007" +
989 "\070\010\065\011\071\012\074\020\073\001\001\000\002" +
990 "\001\001\000\002\001\001\000\000\002\001\001\000\024\003" +
991 "\0216\004\021\006\026\007\027\010\017\011\034\012\040" +
992 "\015\031\w020\037\001\001\000\020\004\066\006\067\007" +
993 "\070\010\065\011\071\012\074\020\073\001\001\000\002" +
994 "\001\001\000\002\001\001\000\000\024\003\w021\006" +
995 "\026\007\027\010\017\011\034\012\040\015\031\w020\037" +
996 "\001\001\000\000\020\004\066\006\067\007\070\010\065\011" +
997 "\071\012\074\020\073\001\001\000\020\001\001\000\020" +
998 "\004\066\006\067\007\070\010\065\011\071\012\074\020" +
999 "\073\001\001\000\002\001\001\000\002\001\001\000\002" +
1000 "\001\001\000\024\003\w0221\004\021\006\026\007\027\010" +
1001 "\017\011\034\012\040\015\031\w020\037\001\001\000\020" +
1002 "\004\066\006\067\007\070\010\065\011\071\012\074\020" +
1003 "\073\001\001\000\002\001\001\000\002\001\001\000\024" +
1004 "\003\w0231\004\021\006\026\007\027\010\017\011\034\012" +
1005 "\040\015\031\w020\037\001\001\000\002\001\001\000\002" +
1006 "\001\001\000\004\013\w0228\001\001\000\002\001\001\000" +
1007 "\002\001\001\000\024\003\w022b\004\021\006\026\007\027" +
```

```
1012 "012\009\013\020\037\001\000\000\000\000\000" +
1013 "\067\007\070\010\065\011\071\012\074\020\073\001\001" +
1014 "\000\002\001\001\000\020\004\066\006\067\007\070\010" +
1015 "\065\011\071\012\074\020\073\001\001\000\002\001\001" +
1016 "\000\002\001\001\000\002\001\001\000\000\002\001\000" +
1017 "\002\001\001\000\024\003\w0238\004\021\006\026\007\027" +
1018 "\010\017\011\034\012\040\015\031\w020\037\001\001\000" +
1019 "\020\004\066\006\067\007\070\010\065\011\071\012\074" +
1020 "\020\073\001\001\000\002\001\001\000\002\001\001\000" +
1021 "\002\001\001\000\024\003\w023d\004\021\006\026\007\027" +
1022 "\010\017\011\034\012\040\015\031\w020\037\001\001\000" +
1023 "\020\004\066\006\067\007\070\010\065\011\071\012\074" +
1024 "\020\073\001\001\000\002\001\001\000\002\001\001" );
1025
1026 /**
1027  * Access to <code>reduce_goto</code> table.
1028 */
1029 public short[][] reduce_table() {return reduce_table;}
1030
1031 /**
1032  * Instance of action encapsulation class.
1033 */
1034 protected CUP$Syntax$Actions action_obj;
1035
1036 /**
1037  * Action encapsulation object initializer.
1038 */
1039 protected void init_actions()
1040 {
1041     action_obj = new CUP$Syntax$Actions(this);
1042 }
1043
1044 /**
1045  * Invoke a user supplied parse action.
1046 */
1047 public java_cup.runtime.Symbol do_action(
1048     int act_num,
1049     java_cup.runtime.lr_parser parser,
1050     java.util.Stack stack,
1051     int top)
1052     throws java.lang.Exception
1053 {
1054     /* call code in generated class */
1055     return action_obj.CUP$Syntax$do_action(act_num, parser, stack, top);
1056 }
```



MANUAL DE PRÁCTICAS



```
1050    /** Indicates start state. */
1051    public int start_state() {return 0;}
1052    /** Indicates start production. */
1053    public int start_production() {return 0;}
1054
1055    /** <code>EOF</code> Symbol index. */
1056    public int EOF_sym() {return 0;}
1057
1058    /** <code>error</code> Symbol index. */
1059    public int error_sym() {return 1;}
1060
1061
1062
1063
1064    private java.util.List<String> errores = new java.util.ArrayList<>();
1065    private String textoCompleto; // Para calcular las columnas por linea
1066
1067    public void syntax_error(Symbol s) {
1068        int linea = s.right + 1;
1069        int columnaReal = calcularColumnaReal(linea, s.left + 1);
1070
1071        errores.add("Error de sintaxis. Línea: " + linea +
1072                    ", Columna: " + columnaReal + ", Texto: \\" + s.value + "\\\"");
1073    }
1074
1075    public java.util.List<String> getErrores() {
1076        return errores;
1077    }
1078
1079    // Configura el texto completo para cálculos de posición
1080    public void setTextoCompleto(String textoCompleto) {
1081        this.textoCompleto = textoCompleto;
1082    }
1083
1084    // Calcula la columna real reiniendo en cada nueva linea
1085    private int calcularColumnaReal(int linea, int posicionGlobal) {
```

```
1086     int contadorLineas = 1;
1087     int i = 0;
1088
1089     // Recorremos el texto hasta la posición global
1090     while (i < posicionGlobal) {
1091         char c = textoCompleto.charAt(i);
1092
1093         if (c == '\n' || (c == '\r' && i + 1 < textoCompleto.length() && textoCompleto.charAt(i + 1) == '\n')) {
1094             // Si encontramos un salto de linea (considerando \r\n en Windows)
1095             contadorLineas++;
1096             if (contadorLineas == linea) {
1097                 columna = 1; // Reiniciamos la columna al principio de la nueva linea
1098             }
1099             if (c == '\r' && i + 1 < textoCompleto.length() && textoCompleto.charAt(i + 1) == '\n') {
1100                 i++; // Avanzamos el indice para saltar el '\n' que sigue al '\r'
1101             }
1102         } else if (contadorLineas == linea) {
1103             columna++; // Si estamos en la linea correcta, aumentamos la columna
1104         }
1105         i++;
1106     }
1107     return columna;
1108 }
1109
1110
1111
1112 }
1113
1114 /**
1115  * Cup generated class to encapsulate user supplied action code.*/
1116 class CUP$Sintax$Actions {
1117     private final Sintax parser;
1118
1119     /** Constructor */
1120     CUP$Sintax$Actions(Sintax parser) {
1121         this.parser = parser;
1122     }
1123 }
```



MANUAL DE PRÁCTICAS



```
1119  CUP$Sintax$actions(Syntax parser) {
1120      this.parser = parser;
1121  }
1122
1123  /** Method with the actual generated action code. */
1124  public final java_cup.runtime.Symbol CUP$Sintax$do_action(
1125      int CUP$Sintax$act_num,
1126      java_cup.runtime.lr_parser CUP$Sintax$parser,
1127      java.util.Stack CUP$Sintax$stack,
1128      int CUP$Sintax$stop)
1129  throws java.lang.Exception
1130  {
1131      /* Symbol object for return from actions */
1132      java_cup.runtime.Symbol CUP$Sintax$result;
1133
1134      /* select the action based on the action number */
1135      switch (CUP$Sintax$act_num)
1136      {
1137          /* . . . . . */
1138          case 164: // OPERACION_ARITMETICA ::= Numero Raiz Numero P_coma
1139          {
1140              Object RESULT =null;
1141
1142              CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1143
1144              return CUP$Sintax$result;
1145
1146          /* . . . . . */
1147          case 163: // OPERACION_ARITMETICA ::= Numero Potencia Numero P_coma
1148          {
1149              Object RESULT =null;
1150
1151              CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1152
1153          return CUP$Sintax$result;
```

```
1155      /* . . . . . */
1156      case 162: // OPERACION_ARITMETICA ::= Numero Modulo Numero P_coma
1157      {
1158          Object RESULT =null;
1159
1160          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1161
1162          return CUP$Sintax$result;
1163
1164      /* . . . . . */
1165      case 161: // OPERACION_ARITMETICA ::= Numero Division Numero P_coma
1166      {
1167          Object RESULT =null;
1168
1169          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1170
1171          return CUP$Sintax$result;
1172
1173      /* . . . . . */
1174      case 160: // OPERACION_ARITMETICA ::= Numero Multiplicacion Numero P_coma
1175      {
1176          Object RESULT =null;
1177
1178          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1179
1180          return CUP$Sintax$result;
1181
1182      /* . . . . . */
1183      case 159: // OPERACION_ARITMETICA ::= Numero Resta Numero P_coma
1184      {
1185          Object RESULT =null;
1186
1187          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java
1188
1189          return CUP$Sintax$result;
```



```
1192     case 158: // OPERACION_ARITMETICA ::= Numero Suma Numero P_coma
1193     {
1194         Object RESULT =null;
1195
1196         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("OPERACION_ARITMETICA",14, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-3)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)));
1197     }
1198     return CUPSSyntax$Result;
1199
1200     /* . . . . . */
1201     case 157: // METODO ::= Metodo Identificador ParentesisApertura PARAMETROS ParentesisCierre LlaveApertura CUERPO_METODO LlaveCierre
1202     {
1203         Object RESULT =null;
1204
1205         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("METODO",11, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-7)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-6)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-5)));
1206     }
1207     return CUPSSyntax$Result;
1208
1209     /* . . . . . */
1210     case 156: // CUERPO_METODO ::= SENTENCIA Retornar Euler P_coma
1211     {
1212         Object RESULT =null;
1213
1214         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("CUERPO_METODO",13, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-3)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)));
1215     }
1216     return CUPSSyntax$Result;
1217
1218     /* . . . . . */
1219     case 155: // CUERPO_METODO ::= SENTENCIA Retornar Pi P_coma
1220     {
1221         Object RESULT =null;
1222
1223         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("CUERPO_METODO",13, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-3)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)));
1224     }
1225     return CUPSSyntax$Result;
1226
1227
1300     case 146: // PARAMETROS ::= Entero Identificador
1301     {
1302         Object RESULT =null;
1303
1304         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("PARAMETROS",12, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)));
1305     }
1306     return CUPSSyntax$Result;
1307
1308     /* . . . . . */
1309     case 145: // PARAMETROS ::= Identificador
1310     {
1311         Object RESULT =null;
1312
1313         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("PARAMETROS",12, ((java_cup.runtime.Symbol)CUPSSyntax$stack.peek()), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)));
1314     }
1315     return CUPSSyntax$Result;
1316
1317     /* . . . . . */
1318     case 144: // DECLARACION_FOR ::= Decremento Identificador
1319     {
1320         Object RESULT =null;
1321
1322         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION_FOR",3, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)));
1323     }
1324     return CUPSSyntax$Result;
1325
1326     /* . . . . . */
1327     case 143: // DECLARACION_FOR ::= Incremento Identificador
1328     {
1329         Object RESULT =null;
1330
1331         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION_FOR",3, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-1)), ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-2)));
1332     }
1333     return CUPSSyntax$Result;
1334
1335     /* . . . . . */
```



MANUAL DE PRÁCTICAS



```
1366     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.ru
1367     )
1368     return CUP$Syntax$result;
1369
1370     /* . . . . . */
1371     case 138: // SENTENCIA_FOR ::= Entero Identificador MenorIgual Numero P_coma SENTENCIA_BOLEANA P_coma DECLARACION_FOR
1372     {
1373         Object RESULT =null;
1374
1375         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.ru
1376         )
1377         return CUP$Syntax$result;
1378
1379     /* . . . . . */
1380     case 137: // SENTENCIA_FOR ::= Entero Identificador MayorIgual Numero P_coma SENTENCIA_BOLEANA P_coma DECLARACION_FOR
1381     {
1382         Object RESULT =null;
1383
1384         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.ru
1385         )
1386         return CUP$Syntax$result;
1387
1388     /* . . . . . */
1389     case 136: // SENTENCIA_FOR ::= Entero Identificador Menor Numero P_coma SENTENCIA_BOLEANA P_coma DECLARACION_FOR
1390     {
1391         Object RESULT =null;
1392
1393         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.ru
1394         )
1395         return CUP$Syntax$result;
1396
1397     /* . . . . . */
1398     case 135: // SENTENCIA_FOR ::= Entero Identificador Mayor Numero P_coma SENTENCIA_BOLEANA P_coma DECLARACION_FOR
1399     {
1400         Object RESULT =null;
```

```
1401
1402     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.run
1403     )
1404     return CUP$Syntax$result;
1405
1406     /* . . . . . */
1407     case 134: // SENTENCIA_FOR ::= Entero Identificador Igual Numero P_coma SENTENCIA_BOLEANA P_coma DECLARACION_FOR
1408     {
1409         Object RESULT =null;
1410
1411         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_FOR",10, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.run
1412         )
1413         return CUP$Syntax$result;
1414
1415     /* . . . . . */
1416     case 133: // FOR ::= CicloRepite ParentesisApertura SENTENCIA_FOR ParentesisCierre LlaveApertura Imprimir ParentesisApertura Identificador ParentesisCierre P_coma Llav
1417     {
1418         Object RESULT =null;
1419
1420         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FOR",8, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-10)), ((java_cup.runtime.Symbol
1421         )
1422         return CUP$Syntax$result;
1423
1424
1425     /* . . . . . */
1426     case 132: // FOR ::= CicloRepite ParentesisApertura SENTENCIA_FOR ParentesisCierre LlaveApertura SENTENCIA_LlaveApertura SENTENCIA_LlaveCierre
1427     {
1428         Object RESULT =null;
1429
1430         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("FOR",8, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runtime.Symbol
1431         )
1432         return CUP$Syntax$result;
1433
1434
1435     /* . . . . . */
1436     case 131: // DO WHILE ::= Hacer LlaveApertura SENTENCIA_LlaveCierre Mientras ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre P_coma
1437     {
1438         Object RESULT =null;
```



```
1448     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-10)), ((java_cup.runtime.
1449     )
1450     return CUP$Syntax$result;
1451
1452     /* . . . . . */
1453     case 129: // DO WHILE ::= Hacer LlaveApertura SENTENCIA LlaveCierre Mientras ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre P_coma
1454     {
1455         Object RESULT =null;
1456
1457         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-10)), ((java_cup.runtime.
1458         )
1459         return CUP$Syntax$result;
1460
1461     /* . . . . . */
1462     case 128: // DO WHILE ::= Hacer LlaveApertura SENTENCIA LlaveCierre Mientras ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre P_coma
1463     {
1464         Object RESULT =null;
1465
1466         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-8)), ((java_cup.runtime.Symb
1467         )
1468         return CUP$Syntax$result;
1469
1470     /* . . . . . */
1471     case 127: // WHILE ::= Mientras ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1472     {
1473         Object RESULT =null;
1474
1475         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.runtime.Symb
1476         )
1477         return CUP$Syntax$result;
1478
1479     /* . . . . . */
1480     case 126: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1481     {
1482         Object RESULT =null;
```

```
1447     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-10)), ((java_cup.runtime.
1448     )
1449     return CUP$Syntax$result;
1450
1451     /* . . . . . */
1452     case 129: // DO WHILE ::= Hacer LlaveApertura SENTENCIA LlaveCierre Mientras ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre P_coma
1453     {
1454         Object RESULT =null;
1455
1456         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-10)), ((java_cup.runtime.
1457         )
1458         return CUP$Syntax$result;
1459
1460     /* . . . . . */
1461     case 128: // DO WHILE ::= Hacer LlaveApertura SENTENCIA LlaveCierre Mientras ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre P_coma
1462     {
1463         Object RESULT =null;
1464
1465         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DO_WHILE",7, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-8)), ((java_cup.runtime.Symb
1466         )
1467         return CUP$Syntax$result;
1468
1469     /* . . . . . */
1470     case 127: // WHILE ::= Mientras ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1471     {
1472         Object RESULT =null;
1473
1474         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)), ((java_cup.runtime.Symb
1475         )
1476         return CUP$Syntax$result;
1477
1478     /* . . . . . */
1479     case 126: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1480     {
1481         Object RESULT =null;
```



```
1480      case 126: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1481      {
1482          Object RESULT =null;
1483
1484          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)));
1485      }
1486      return CUP$SyntaxResult;
1487
1488      /* . . . . . */
1489      case 125: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1490      {
1491          Object RESULT =null;
1492
1493          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-8)));
1494      }
1495      return CUP$SyntaxResult;
1496
1497      /* . . . . . */
1498      case 124: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1499      {
1500          Object RESULT =null;
1501
1502          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-6)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-6)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-6)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-6)));
1503      }
1504      return CUP$SyntaxResult;
1505
1506      /* . . . . . */
1507      case 123: // IF_ELSE ::= Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura No SENTENCIA_BOOLEA
1508      {
1509          Object RESULT =null;
1510
1511          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)));
1512      }
1513      return CUP$SyntaxResult;
```

```
1514
1515
1516      }
1517      return CUP$SyntaxResult;
1518
1519      /* . . . . . */
1520      case 122: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA O
1521      {
1522          Object RESULT =null;
1523
1524          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)));
1525      }
1526      return CUP$SyntaxResult;
1527
1528
1529      /* . . . . . */
1530      case 121: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA Y
1531      {
1532          Object RESULT =null;
1533
1534          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-20)));
1535      }
1536      return CUP$SyntaxResult;
1537
1538
1539      /* . . . . . */
1540      case 120: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1541      {
1542          Object RESULT =null;
1543
1544          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)));
1545      }
1546      return CUP$SyntaxResult;
1547
1548
1549      /* . . . . . */
1550      case 119: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1551      {
1552          Object RESULT =null;
1553
1554          CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)), ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-21)));
1555      }
1556      return CUP$SyntaxResult;
```



```
1488     /* . . . . . */
1489     case 125: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1490     {
1491         Object RESULT =null;
1492
1493         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-8)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-7)));
1494     }
1495     return CUP$SyntaxResult;
1496
1497     /* . . . . . */
1498     case 124: // WHILE ::= Mientras ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1499     {
1500         Object RESULT =null;
1501
1502         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("WHILE",6, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-5)));
1503     }
1504     return CUP$SyntaxResult;
1505
1506     /* . . . . . */
1507     case 123: // IF_ELSE ::= Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura No SENTENCIA_BOLEANA O
1508     {
1509         Object RESULT =null;
1510
1511         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-20)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-19)));
1512     }
1513     return CUP$SyntaxResult;
1514
1515     /* . . . . . */
1516     case 122: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA O
1517     {
1518         Object RESULT =null;
1519
1520         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-20)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-19)));
1521     }
1522     return CUP$SyntaxResult;
```

```
1521     }
1522     return CUP$SyntaxResult;
1523
1524     /* . . . . . */
1525     case 121: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA Y
1526     {
1527         Object RESULT =null;
1528
1529         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-20)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-19)));
1530     }
1531     return CUP$SyntaxResult;
1532
1533     /* . . . . . */
1534     case 120: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1535     {
1536         Object RESULT =null;
1537
1538         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-21)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-20)));
1539     }
1540     return CUP$SyntaxResult;
1541
1542     /* . . . . . */
1543     case 119: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1544     {
1545         Object RESULT =null;
1546
1547         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-21)), ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-20)));
1548     }
1549     return CUP$SyntaxResult;
1550
1551     /* . . . . . */
1552     case 118: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1553     {
1554         Object RESULT =null;
1555     }
```



```
1553     {
1554         Object RESULT =null;
1555
1556         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-22)), ((java_cup.runtime.S
1557     )
1558
1559     return CUP$Syntax$result;
1560
1561     /*. . . . . */
1562     case 117: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1563     {
1564         Object RESULT =null;
1565
1566         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-22)), ((java_cup.runtime.S
1567     )
1568
1569     return CUP$Syntax$result;
1570
1571     /*. . . . . */
1572     case 116: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1573     {
1574         Object RESULT =null;
1575
1576         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-22)), ((java_cup.runtime.S
1577     )
1578
1579     return CUP$Syntax$result;
1580
1581     /*. . . . . */
1582     case 115: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura
1583     {
1584         Object RESULT =null;
1585
1586         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-22)), ((java_cup.runtime.S
1587     )
1588
1589     /*. . . . . */
1590     case 114: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA Pa
```

```
1596
1597     /*. . . . . */
1598     case 114: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo Si ParentesisApertura SENTENCIA_BOLEANA Pa
1599     {
1600         Object RESULT =null;
1601
1602         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-18)), ((java_cup.runtime.S
1603     )
1604
1605     return CUP$Syntax$result;
1606
1607     /*. . . . . */
1608     case 113: // IF_ELSE ::= Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo LlaveApertura SENTENCIA LlaveCierre
1609     {
1610         Object RESULT =null;
1611
1612         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-11)), ((java_cup.runtime.S
1613     )
1614
1615     return CUP$Syntax$result;
1616
1617     /*. . . . . */
1618     case 111: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA Y SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo LlaveApertura SENTENCI
1619     {
1620         Object RESULT =null;
1621
1622         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-12)), ((java_cup.runtime.S
```



MANUAL DE PRÁCTICAS



```
1622
1623     /* . . . . . */
1624     case 110: // IF_ELSE ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre SiNo LlaveApertura SENTENCIA LlaveCierre
1625     {
1626         Object RESULT =null;
1627
1628         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("IF_ELSE",5, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-10)), ((java_cup.runtime.
1629         )
1630     return CUP$SyntaxResult;
1631
1632     /* . . . . . */
1633     case 109: // SENTENCIA_BOLEANA ::= Identificador Modulo Numero Comparacion Numero
1634     {
1635         Object RESULT =null;
1636
1637         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-4)), ((java_cu
1638         )
1639     return CUP$SyntaxResult;
1640
1641     /* . . . . . */
1642     case 108: // SENTENCIA_BOLEANA ::= Identificador Comparacion Euler
1643     {
1644         Object RESULT =null;
1645
1646         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-2)), ((java_cu
1647         )
1648     return CUP$SyntaxResult;
1649
1650     /* . . . . . */
1651     case 107: // SENTENCIA_BOLEANA ::= Identificador Comparacion Pi
1652     {
1653         Object RESULT =null;
1654
1655         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-2)), ((java_cu
1656         )
1657     return CUP$SyntaxResult;
```

```
1659
1660     /* . . . . . */
1661     case 106: // SENTENCIA_BOLEANA ::= Identificador Igual Falso
1662     {
1663         Object RESULT =null;
1664
1665         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-2)), ((java_cu
1666         )
1667     return CUP$SyntaxResult;
1668
1669     /* . . . . . */
1670     case 105: // SENTENCIA_BOLEANA ::= Identificador Igual Verdadero
1671     {
1672         Object RESULT =null;
1673
1674         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-2)), ((java_cu
1675         )
1676     return CUP$SyntaxResult;
1677
1678     /* . . . . . */
1679     case 104: // SENTENCIA_BOLEANA ::= Identificador Comparacion ComillaDoble Texto ComillaDoble
1680     {
1681         Object RESULT =null;
1682
1683         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-4)), ((java_cu
1684         )
1685     return CUP$SyntaxResult;
1686
1687     /* . . . . . */
1688     case 103: // SENTENCIA_BOLEANA ::= Identificador Comparacion ComillaDoble Identificador ComillaDoble
1689     {
1690         Object RESULT =null;
1691
1692         CUP$SyntaxResult = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$SyntaxStack.elementAt(CUP$SyntaxStop-4)), ((java_cu
1693         )
1694     return CUP$SyntaxResult;
```



```
1695      /*. . . . . */
1696      case 102: // SENTENCIA_BOLEANA ::= Identificador Comparacion ComillaDoble ComillaDoble
1697      {
1698          Object RESULT =null;
1699
1700          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-3)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1701
1702          return CUP$Sintax$result;
1703
1704      /*. . . . . */
1705      case 101: // SENTENCIA_BOLEANA ::= Identificador Igual Texto
1706      {
1707          Object RESULT =null;
1708
1709          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1710
1711          return CUP$Sintax$result;
1712
1713      /*. . . . . */
1714      case 100: // SENTENCIA_BOLEANA ::= Identificador Comparacion Texto
1715      {
1716          Object RESULT =null;
1717
1718          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1719
1720          return CUP$Sintax$result;
1721
1722      /*. . . . . */
1723      case 99: // SENTENCIA_BOLEANA ::= Identificador Comparacion Identificador
1724      {
1725          Object RESULT =null;
1726
1727          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1728
1729          return CUP$Sintax$result;
```

```
1760      {
1761          Object RESULT =null;
1762
1763          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1764
1765          return CUP$Sintax$result;
1766
1767      /*. . . . . */
1768      case 94: // SENTENCIA_BOLEANA ::= Identificador Mayor Numero
1769      {
1770          Object RESULT =null;
1771
1772          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1773
1774          return CUP$Sintax$result;
1775
1776      /*. . . . . */
1777      case 93: // SENTENCIA_BOLEANA ::= Identificador Comparacion Numero
1778      {
1779          Object RESULT =null;
1780
1781          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA_BOLEANA",9, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-1)));
1782
1783          return CUP$Sintax$result;
1784
1785      /*. . . . . */
1786      case 92: // IF ::= Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
1787      {
1788          Object RESULT =null;
1789
1790          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("IF",4, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-7)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-6)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-5)));
1791
1792          return CUP$Sintax$result;
1793
1794      /*. . . . . */
1795      case 91: // ELSE ::= Si ParentesisApertura No SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
```



MANUAL DE PRÁCTICAS



```
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835

    return CUP$Syntax$result;

    /* . . . . . */
case 90: // IF ::= Si ParentesisApertura SENTENCIA_BOLEANA O SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF",4, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-8)), ((java_cup.runtime.Symbol)
}
return CUP$Syntax$result;

    /* . . . . . */
case 89: // IF ::= Si ParentesisApertura SENTENCIA_BOLEANA ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("IF",4, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runtime.Symbol)
}
return CUP$Syntax$result;

    /* . . . . . */
case 88: // DECLARACION ::= Identificador Igual Texto P_coma
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runt
}
return CUP$Syntax$result;

    /* . . . . . */
case 87: // DECLARACION ::= Cadena Identificador Igual Texto P_coma
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runt
}
```

```
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882

    {
        Object RESULT =null;

        CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runt
    }
return CUP$Syntax$result;

    /* . . . . . */
case 84: // DECLARACION ::= Cadena Constante Igual ComillaDoble Identificador ComillaDoble P_coma
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runt
}
return CUP$Syntax$result;

    /* . . . . . */
case 83: // DECLARACION ::= Cadena Constante Asignacion CcomillaDoble Identificador ComillaDoble P_coma
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runt
}
return CUP$Syntax$result;

    /* . . . . . */
case 82: // DECLARACION ::= Cadena Constante Igual ComillaDoble ComillaDoble P_coma
{
    Object RESULT =null;

    CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-5)), ((java_cup.runt
}
return CUP$Syntax$result;
```



MANUAL DE PRÁCTICAS



```
1890     CUPSSyntaxResult = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-4)), ((java_cup.runti
1891     )
1892     return CUPSSyntaxResult;
1893
1894     /* . . . . . */
1895     case 79: // DECLARACION ::= Entero Constante Punto Número P_coma
1896     {
1897         Object RESULT =null;
1898
1899         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-4)), ((java_cup.runti
1900     }
1901     return CUPSSyntaxResult;
1902
1903     /* . . . . . */
1904     case 78: // DECLARACION ::= Entero Constante Modulo Número P_coma
1905     {
1906         Object RESULT =null;
1907
1908         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-4)), ((java_cup.runti
1909     }
1910     return CUPSSyntaxResult;
1911
1912     /* . . . . . */
1913     case 77: // DECLARACION ::= Entero Constante Raiz Número P_coma
1914     {
1915         Object RESULT =null;
1916
1917         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-4)), ((java_cup.runti
1918     }
1919     return CUPSSyntaxResult;
1920
1921     /* . . . . . */
1922     case 76: // DECLARACION ::= Entero Constante Potencia Número P_coma
1923     {
1924         Object RESULT =null;
1925
1926         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-4)), ((java_cup.runti
1927     }
1928     return CUPSSyntaxResult;
1929
1930     /* . . . . . */
1931     case 75: // DECLARACION ::= Cadena Constante Igual ComillaDoble Identificador ComillaDoble P_coma
1932     {
1933         Object RESULT =null;
```

```
1859     case 84: // DECLARACION ::= Cadena Constante Igual ComillaDoble Identificador ComillaDoble P_coma
1860     {
1861         Object RESULT =null;
1862
1863         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-6)), ((java_cup.runti
1864     }
1865     return CUPSSyntaxResult;
1866
1867     /* . . . . . */
1868     case 83: // DECLARACION ::= Cadena Constante Asignacion ComillaDoble Identificador ComillaDoble P_coma
1869     {
1870         Object RESULT =null;
1871
1872         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-6)), ((java_cup.runti
1873     }
1874     return CUPSSyntaxResult;
1875
1876     /* . . . . . */
1877     case 82: // DECLARACION ::= Cadena Constante Igual ComillaDoble ComillaDoble P_coma
1878     {
1879         Object RESULT =null;
1880
1881         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-5)), ((java_cup.runti
1882     }
1883     return CUPSSyntaxResult;
1884
1885     /* . . . . . */
1886     case 81: // DECLARACION ::= Cadena Constante Asignacion ComillaDoble ComillaDoble P_coma
1887     {
1888         Object RESULT =null;
1889
1890         CUPSSyntax$Result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUPSSyntax$stack.elementAt(CUPSSyntax$top-5)), ((java_cup.runti
1891     }
1892     return CUPSSyntaxResult;
1893
1894     /* . . . . . */
```



MANUAL DE PRÁCTICAS



```
1914:     Object RESULT =null;
1915: 
1916:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtim
1917: )
1918:     return CUP$Syntax$result;
1919: 
1920:     /* . . . . . */
1921: case 77: // DECLARACION ::= Entero Constante Raiz Numero P_coma
1922: {
1923:     Object RESULT =null;
1924: 
1925:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtim
1926: )
1927:     return CUP$Syntax$result;
1928: 
1929:     /* . . . . . */
1930: case 76: // DECLARACION ::= Entero Constante Potencia Numero P_coma
1931: {
1932:     Object RESULT =null;
1933: 
1934:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtim
1935: )
1936:     return CUP$Syntax$result;
1937: 
1938:     /* . . . . . */
1939: case 75: // DECLARACION ::= Decimal Constante Decremento P_coma
1940: {
1941:     Object RESULT =null;
1942: 
1943:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runtim
1944: )
1945:     return CUP$Syntax$result;
1946: 
1947:     /* . . . . . */
1948: case 74: // DECLARACION ::= Decimal Constante Incremento P_coma
1949: 
```

```
1950:     return CUP$Syntax$result;
1951: 
1952:     /* . . . . . */
1953: case 76: // DECLARACION ::= Entero Constante Potencia Numero P_coma
1954: {
1955:     Object RESULT =null;
1956: 
1957:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtim
1958: )
1959:     return CUP$Syntax$result;
1960: 
1961:     /* . . . . . */
1962: case 75: // DECLARACION ::= Decimal Constante Decremento P_coma
1963: {
1964:     Object RESULT =null;
1965: 
1966:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runtim
1967: )
1968:     return CUP$Syntax$result;
1969: 
1970:     /* . . . . . */
1971: case 74: // DECLARACION ::= Decimal Constante Incremento P_coma
1972: {
1973:     Object RESULT =null;
1974: 
1975:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runtim
1976: )
1977:     return CUP$Syntax$result;
1978: 
1979:     /* . . . . . */
1980: case 73: // DECLARACION ::= Decimal Constante Igual Numero Punto Numero P_coma
1981: {
1982:     Object RESULT =null;
1983: 
1984:     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-6)), ((java_cup.runtim
1985: )
1986:     return CUP$Syntax$result;
1987: 
```



MANUAL DE PRÁCTICAS



```
1959     Object RESULT =null;
1960
1961     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-6)), ((java_cup.runtime.
1962     )
1963     return CUP$Sintax$result;
1964
1965     /* . . . . . */
1966     case 72: // DECLARACION ::= Decimal Constante Asignacion Numero Punto Número P_coma
1967     {
1968         Object RESULT =null;
1969
1970         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-6)), ((java_cup.runtime.
1971         )
1972     return CUP$Sintax$result;
1973
1974     /* . . . . . */
1975     case 71: // DECLARACION ::= Decimal Constante P_coma
1976     {
1977         Object RESULT =null;
1978
1979         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.
1980         )
1981     return CUP$Sintax$result;
1982
1983     /* . . . . . */
1984     case 70: // DECLARACION ::= Entero Constante Decremento P_coma
1985     {
1986         Object RESULT =null;
1987
1988         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-3)), ((java_cup.runtime.
1989         )
1990     return CUP$Sintax$result;
1991
1992     /* . . . . . */
1993     case 69: // DECLARACION ::= Entero Constante Incremento P_coma
1994     {
```

```
1996     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-3)), ((java_cup.runtime.
1997     )
1998     return CUP$Sintax$result;
1999
2000
2001     /* . . . . . */
2002     case 68: // DECLARACION ::= Entero Constante Igual Número P_coma
2003     {
2004         Object RESULT =null;
2005
2006         CUP$SintaxResult = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-4)), ((java_cup.runtime.
2007         )
2008     return CUP$SintaxResult;
2009
2010
2011     /* . . . . . */
2012     case 67: // DECLARACION ::= Entero Constante Asignacion Número P_coma
2013     {
2014         Object RESULT =null;
2015
2016         CUP$SintaxResult = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-4)), ((java_cup.runtime.
2017         )
2018     return CUP$SintaxResult;
2019
2020
2021     /* . . . . . */
2022     case 66: // DECLARACION ::= Entero Constante P_coma
2023     {
2024         Object RESULT =null;
2025
2026         CUP$SintaxResult = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-2)), ((java_cup.runtime.
2027         )
2028     return CUP$SintaxResult;
2029
2030
2031     /* . . . . . */
2032     case 65: // DECLARACION ::= Constante Decremento P_coma
2033     {
```



```
2044     return CUP$Syntax$result;
2045
2046     /*. . . . . */
2047     case 63: // DECLARACION ::= Constante Igual Número P_coma
2048     {
2049         Object RESULT =null;
2050
2051         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runtime.
2052
2053         return CUP$Syntax$result;
2054
2055     /*. . . . . */
2056     case 62: // DECLARACION ::= Constante Asignación Número P_coma
2057     {
2058         Object RESULT =null;
2059
2060         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-3)), ((java_cup.runtime.
2061
2062         return CUP$Syntax$result;
2063
2064     /*. . . . . */
2065     case 61: // DECLARACION ::= Constante P_coma
2066     {
2067         Object RESULT =null;
2068
2069         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-1)), ((java_cup.runtime.
2070
2071         return CUP$Syntax$result;
2072
2073     /*. . . . . */
2074     case 60: // DECLARACION ::= Identificador Igual ComillaDoble Identificador ComillaDoble P_coma
2075     {
2076         Object RESULT =null;
2077
2078         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-5)), ((java_cup.runtime.
2079
2080         return CUP$Syntax$result;
```

```
2081
2082
2083     /*. . . . . */
2084     case 59: // DECLARACION ::= Identificador Asignación ComillaDoble Identificador ComillaDoble P_coma
2085     {
2086         Object RESULT =null;
2087
2088         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-5)), ((java_cup.runtime.
2089
2090         return CUP$Syntax$result;
2091
2092     /*. . . . . */
2093     case 58: // DECLARACION ::= Identificador Igual ComillaDoble ComillaDoble P_coma
2094     {
2095         Object RESULT =null;
2096
2097         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtime.
2098
2099         return CUP$Syntax$result;
2100
2101     /*. . . . . */
2102     case 57: // DECLARACION ::= Identificador Asignación ComillaDoble ComillaDoble P_coma
2103     {
2104         Object RESULT =null;
2105
2106         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$top-4)), ((java_cup.runtime.
2107
2108         return CUP$Syntax$result;
2109
2110     /*. . . . . */
2111     case 56: // DECLARACION ::= Cadena Identificador Igual ComillaDoble Identificador ComillaDoble P_coma
2112     {
2113         Object RESULT =null;
```



MANUAL DE PRÁCTICAS



```
2113     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-6)), ((java_cup.runti
2114   }
2115   return CUP$Sintax$result;
2116
2117 /**
2118 * DECLARACION ::= Cadena Identificador Asignacion ComillaDoble Identificador ComillaDoble P_coma
2119 */
2120   {
2121     Object RESULT =null;
2122
2123     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-6)), ((java_cup.runti
2124   }
2125   return CUP$Sintax$result;
2126
2127 /**
2128 * DECLARACION ::= Cadena Identificador Igual ComillaDoble ComillaDoble P_coma
2129 */
2130   {
2131     Object RESULT =null;
2132
2133     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-5)), ((java_cup.runti
2134   }
2135   return CUP$Sintax$result;
2136
2137 /**
2138 * DECLARACION ::= Cadena Identificador Asignacion ComillaDoble ComillaDoble P_coma
2139 */
2140   {
2141     Object RESULT =null;
2142
2143     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-5)), ((java_cup.runti
2144   }
2145   return CUP$Sintax$result;
2146
2147 /**
2148 * DECLARACION ::= Cadena Identificador Concatenacion Identificador P_coma
2149 */
2150   {
2151     Object RESULT =null;
```

```
2152
2153
2154 /**
2155 * DECLARACION ::= Cadena Identificador Asignacion ComillaDoble P_coma
2156 */
2157   {
2158     Object RESULT =null;
2159
2160     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-5)), ((java_cup.runti
2161   }
2162   return CUP$Sintax$result;
2163
2164 /**
2165 * DECLARACION ::= Entero Identificador Punto Numero P_coma
2166 */
2167   {
2168     Object RESULT =null;
2169
2170     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$top-4)), ((java_cup.runti
2171   }
2172   return CUP$Sintax$result;
2173
2174 /**
2175 * DECLARACION ::= Entero Identificador Modulo Numero P_coma
2176 */
2177   {
2178     Object RESULT =null;
```



MANUAL DE PRÁCTICAS



```
2289      /*. . . . . */
2290      case 36: // DECLARACION ::= Identificador Incremento P_coma
2291      {
2292          Object RESULT =null;
2293
2294          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)));
2295      }
2296      return CUP$Sintax$result;
2297
2298      /*. . . . . */
2299      case 35: // DECLARACION ::= Identificador Igual Numero Punto Numero P_coma
2300      {
2301          Object RESULT =null;
2302
2303          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-5)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-4)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)));
2304      }
2305      return CUP$Sintax$result;
2306
2307      /*. . . . . */
2308      case 34: // DECLARACION ::= Identificador Asignacion Numero Punto Numero P_coma
2309      {
2310          Object RESULT =null;
2311
2312          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-5)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-4)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)));
2313      }
2314      return CUP$Sintax$result;
2315
2316      /*. . . . . */
2317      case 33: // DECLARACION ::= Identificador Igual Numero P_coma
2318      {
2319          Object RESULT =null;
2320
2321          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)));
2322      }
2323      return CUP$Sintax$result;
```

```
2329      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-3)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)));
2330  }
2331  return CUP$Sintax$result;
2332
2333  /*. . . . . */
2334  case 31: // DECLARACION ::= Identificador P_coma
2335  {
2336      Object RESULT =null;
2337
2338      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("DECLARACION",2, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)));
2339  }
2340  return CUP$Sintax$result;
2341
2342  /*. . . . . */
2343  case 30: // SENTENCIA ::= error LlaveCierre
2344  {
2345      Object RESULT =null;
2346      System.err.println("Error en SENTENCIA");
2347      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)));
2348  }
2349  return CUP$Sintax$result;
2350
2351  /*. . . . . */
2352  case 29: // SENTENCIA ::= error P_coma
2353  {
2354      Object RESULT =null;
2355      System.err.println("Error en SENTENCIA");
2356      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)), ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-2)));
2357  }
2358  return CUP$Sintax$result;
2359
2360  /*. . . . . */
2361  case 28: // SENTENCIA ::= Importar ParentesisApertura Identificador ParentesisCierre P_coma
2362  {
2363      Object RESULT =null;
```



MANUAL DE PRÁCTICAS



```
2365      CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-4)), ((java_cup.runtime.
2366      )
2367      return CUP$Sintax$result;
2368
2369      /*. . . . . */
2370      case 27: // SENTENCIA ::= Importar ParentesisApertura Texto ParentesisCierre P_coma
2371      {
2372          Object RESULT =null;
2373
2374          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-4)), ((java_cup.runtime.
2375          )
2376          return CUP$Sintax$result;
2377
2378      /*. . . . . */
2379      case 26: // SENTENCIA ::= Leer ParentesisApertura Identificador ParentesisCierre P_coma
2380      {
2381          Object RESULT =null;
2382
2383          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-4)), ((java_cup.runtime.
2384          )
2385          return CUP$Sintax$result;
2386
2387      /*. . . . . */
2388      case 25: // SENTENCIA ::= Imprimir ParentesisApertura Texto Concatenacion Identificador ParentesisCierre P_coma
2389      {
2390          Object RESULT =null;
2391
2392          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-6)), ((java_cup.runtime.
2393          )
2394          return CUP$Sintax$result;
2395
2396      /*. . . . . */
2397      case 24: // SENTENCIA ::= Imprimir ParentesisApertura Texto ParentesisCierre Concatenacion Identificador P_coma
2398      {
2399          Object RESULT =null;
```

```
2415      /*. . . . . */
2416      case 22: // SENTENCIA ::= OPERACION_ARITMETICA
2417      {
2418          Object RESULT =null;
2419
2420          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2421          )
2422          return CUP$Sintax$result;
2423
2424      /*. . . . . */
2425      case 21: // SENTENCIA ::= SENTENCIA OPERACION_ARITMETICA
2426      {
2427          Object RESULT =null;
2428
2429          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)), ((java_cup.runtime.
2430          )
2431          return CUP$Sintax$result;
2432
2433      /*. . . . . */
2434      case 20: // SENTENCIA ::= METODO
2435      {
2436          Object RESULT =null;
2437
2438          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2439          )
2440          return CUP$Sintax$result;
2441
2442      /*. . . . . */
2443      case 19: // SENTENCIA ::= FOR
2444      {
2445          Object RESULT =null;
2446
2447          CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2448          )
2449          return CUP$Sintax$result;
```



```
2463     Object RESULT =null;
2464
2465     CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2466 )
2467     return CUP$Sintax$result;
2468
2469     /* . . . . . */
2470     case 16: // SENTENCIA ::= SENTENCIA DO WHILE
2471     {
2472         Object RESULT =null;
2473
2474         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$SintaxStop-1)), ((java_cup.runtime.
2475 )
2476         return CUP$Sintax$result;
2477
2478     /* . . . . . */
2479     case 15: // SENTENCIA ::= WHILE
2480     {
2481         Object RESULT =null;
2482
2483         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2484 )
2485         return CUP$Sintax$result;
2486
2487     /* . . . . . */
2488     case 14: // SENTENCIA ::= SENTENCIA WHILE
2489     {
2490         Object RESULT =null;
2491
2492         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$SintaxStop-1)), ((java_cup.runtime.
2493 )
2494         return CUP$Sintax$result;
2495
2496     /* . . . . . */
2497     case 13: // SENTENCIA ::= IF ELSE
2498     {
```

```
2491         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$SintaxStop-1)), ((java_cup.runtime.
2492 )
2493         return CUP$Sintax$result;
2494
2495     /* . . . . . */
2496     case 13: // SENTENCIA ::= IF ELSE
2497     {
2498         Object RESULT =null;
2499
2500         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2501 )
2502         return CUP$Sintax$result;
2503
2504     /* . . . . . */
2505     case 12: // SENTENCIA ::= SENTENCIA IF ELSE
2506     {
2507         Object RESULT =null;
2508
2509         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$SintaxStop-1)), ((java_cup.runtime.
2510 )
2511         return CUP$Sintax$result;
2512
2513     /* . . . . . */
2514     case 11: // SENTENCIA ::= IF
2515     {
2516         Object RESULT =null;
2517
2518         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Sintax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Sintax$sta
2519 )
2520         return CUP$Sintax$result;
2521
2522     /* . . . . . */
2523     case 10: // SENTENCIA ::= SENTENCIA IF
2524     {
2525         Object RESULT =null;
```



MANUAL DE PRÁCTICAS



```
2528     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$stop-1)), ((java_cup.runtime.
2529     )
2530     return CUP$Syntax$result;
2531
2532     /* . . . . . */
2533     case 9: // SENTENCIA ::= DECLARACION
2534     {
2535         Object RESULT =null;
2536
2537         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$sta
2538     }
2539     return CUP$Syntax$result;
2540
2541     /* . . . . . */
2542     case 8: // SENTENCIA ::= SENTENCIA DECLARACION
2543     {
2544         Object RESULT =null;
2545
2546         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$stop-1)), ((java_cup.runtime.
2547     }
2548     return CUP$Syntax$result;
2549
2550     /* . . . . . */
2551     case 7: // SENTENCIA ::= ComentarioMultiple
2552     {
2553         Object RESULT =null;
2554
2555         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$sta
2556     }
2557     return CUP$Syntax$result;
2558
2559     /* . . . . . */
2560     case 6: // SENTENCIA ::= LineaComentario
2561     {
2562         Object RESULT =null;
```

```
2564     CUP$Syntax$result = parser.getSymbolFactory().newSymbol("SENTENCIA",1, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$sta
2565     )
2566     return CUP$Syntax$result;
2567
2568     /* . . . . . */
2569     case 5: // INICIO ::=
2570     {
2571         Object RESULT =null;
2572
2573         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Syntax$stack.peek()), ((java_cup.runtime.Symbol)CUP$Syntax$stack.
2574     }
2575     return CUP$Syntax$result;
2576
2577     /* . . . . . */
2578     case 4: // INICIO ::= error P_coma
2579     {
2580         Object RESULT =null;
2581         System.err.println("Error en SENTENCIA");
2582         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$stop-1)), ((java_cup.runtime.Sy
2583     }
2584     return CUP$Syntax$result;
2585
2586     /* . . . . . */
2587     case 3: // INICIO ::= Clase NombreClase ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
2588     {
2589         Object RESULT =null;
2590
2591         CUP$Syntax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Syntax$stack.elementAt(CUP$Syntax$stop-6)), ((java_cup.runtime.Sy
2592     }
2593     return CUP$Syntax$result;
2594
2595     /* . . . . . */
2596     case 2: // INICIO ::= Clase ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
2597     {
2598         Object RESULT =null;
```



```
2601     }
2602     return CUP$Sintax$result;
2603 
2604     /* . . . . . */
2605     case 1: // INICIO ::= Entero Principal ParentesisApertura ParentesisCierre LlaveApertura SENTENCIA LlaveCierre
2606     {
2607         Object RESULT =null;
2608 
2609         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("INICIO",0, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-6)), ((java_cup.runtime.Sym
2610         )
2611         return CUP$Sintax$result;
2612 
2613     /* . . . . . */
2614     case 0: // $START ::= INICIO EOF
2615     {
2616         Object RESULT =null;
2617         int start_valleft = ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).left;
2618         int start_valright = ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).right;
2619         Object start_val = (Object)((java_cup.runtime.Symbol) CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)).value;
2620         RESULT = start_val;
2621         CUP$Sintax$result = parser.getSymbolFactory().newSymbol("$START",0, ((java_cup.runtime.Symbol)CUP$Sintax$stack.elementAt(CUP$Sintax$stop-1)), ((java_cup.runtime.Sym
2622         )
2623         /* ACCEPT */
2624         CUP$Sintax$parser.done_parsing();
2625         return CUP$Sintax$result;
2626 
2627     /* . . . . . */
2628     default:
2629         throw new Exception(
2630             "Invalid action number found in internal parse table");
2631     }
2632 }
2633 }
```

sym.java

El archivo sym.java es una clase generada automáticamente por la herramienta CUP que actúa como un conjunto de constantes simbólicas asociadas a los diferentes elementos léxicos o tokens que pueden ser reconocidos por el analizador léxico y sintáctico del proyecto. Cada constante se define como un valor entero y representa un símbolo terminal en el lenguaje definido. Por ejemplo, los símbolos Suma, Resta, Multiplicacion, y Division representan operadores aritméticos, mientras que ParentesisApertura y ParentesisCierre corresponden a los paréntesis utilizados en expresiones. También incluye palabras reservadas como Si, Mientras, Principal, y Imprimir, así como identificadores como Numero, Cadena, y Identificador, que representan datos específicos en el código fuente.

El archivo es esencial para que el parser pueda identificar correctamente cada elemento léxico y asociarlo con las reglas gramaticales correspondientes durante el análisis sintáctico, facilitando así la validación y la traducción del código fuente en el proyecto. En esencia, este archivo sirve como puente entre el análisis léxico y el análisis sintáctico, proporcionando una referencia uniforme para los tokens en el lenguaje.

Código:

```
1 //-----
2 // The following code was generated by CUP v0.11a beta 20060608
3 // Sun Jan 05 02:00:15 CST 2025
4 //-----
5
6
7 package codigo;
8
9 /**
10  * CUP generated class containing symbol constants. */
11 public class sym {
12     /* terminals */
13     public static final int ComillaSimple = 37;
14     public static final int Incremento = 24;
15     public static final int CicloRepite = 49;
16     public static final int Menor = 7;
17     public static final int Punto = 18;
18     public static final int Concatenacion = 21;
19     public static final int Negativo = 23;
20     public static final int Decremento = 25;
21     public static final int ParentesisApertura = 28;
22     public static final int Importar = 63;
23     public static final int InicioComentario = 39;
24     public static final int Resta = 3;
25     public static final int Nulo = 66;
26     public static final int LlaveApertura = 32;
27     public static final int Biblioteca = 60;
28     public static final int Suma = 2;
29     public static final int Mientras = 50;
30     public static final int ERROR = 77;
31     public static final int Imprimir = 61;
32     public static final int Cadena = 45;
33     public static final int LlaveCierre = 33;
34     public static final int Principal = 59;
35     public static final int MenorIgual = 9;
36     public static final int Main = 70;
37     public static final int Constante = 71;
```



```
public static final int Constante = 71;
public static final int Comparacion = 10;
public static final int Raiz = 17;
public static final int P_coma = 69;
public static final int Predeterminado = 54;
public static final int ParentesisCierre = 29;
public static final int Caso = 53;
public static final int Pi = 26;
public static final int FinComentario = 40;
public static final int Asignacion = 19;
public static final int SaltoLinea = 67;
public static final int Y = 12;
47 public static final int CorcheteCierre = 31;
49 public static final int O = 13;
public static final int Mayor = 6;
public static final int Texto = 76;
public static final int MayorIgual = 8;
public static final int SiNo = 47;
55 public static final int CaracteresEspeciales = 68;
public static final int EOF = 0;
public static final int Retornar = 64;
public static final int InicioTexto = 34;
public static final int Positivo = 22;
public static final int SiNoSi = 48;
public static final int Bular = 27;
public static final int Numero = 74;
public static final int CorcheteApertura = 30;
public static final int Metodo = 65;
public static final int FinalTexto = 35;
public static final int Division = 5;
public static final int Falso = 44;
public static final int Seleccionador = 52;
public static final int Funcion = 57;
public static final int Verdadero = 43;
public static final int NombreClase = 72;
public static final int Procedimiento = 58;
```

```
50 }
51 public static final int Procedimiento = 58;
52 public static final int error = 1;
53 public static final int No = 14;
54 public static final int Potencia = 16;
55 public static final int Leer = 62;
56 public static final int Identificador = 73;
57 public static final int Hacer = 51;
58 public static final int Entero = 41;
59 public static final int Modulo = 15;
60 public static final int Comentario = 38;
61 public static final int Distinto = 11;
62 public static final int ComillaDoble = 36;
63 public static final int Igual = 20;
64 public static final int Clase = 75;
65 public static final int Detener = 55;
66 public static final int Decimal = 42;
67 public static final int Si = 46;
68 public static final int Arreglo = 56;
69 public static final int Multiplicacion = 4;
```



Principal.java

El archivo Principal.java es la clase principal de un proyecto que automatiza la generación de analizadores léxicos y sintácticos utilizando JFlex y CUP. Configura rutas relativas dentro del proyecto para localizar archivos fuente como Lexer.flex, LexerCup.flex y Sintax.cup. A través del método generar, ejecuta JFlex para generar los analizadores léxicos y CUP para crear los archivos de análisis sintáctico (sym.java y Sintax.java). Posteriormente, gestiona estos archivos moviéndolos a su ubicación final en la estructura del proyecto, asegurando una integración adecuada y evitando conflictos con versiones anteriores. Este archivo simplifica y automatiza todo el flujo de creación del compilador dentro del entorno del proyecto.

Código:

```
1 package codigo;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.nio.file.Files;
6 import java.nio.file.Path;
7 import java.nio.file.Paths;
8
9 public class Principal {
10
11     public static void main(String[] args) throws Exception {
12         // Usar rutas relativas basadas en la estructura del proyecto
13         String baseDir = "src/codigo"; // Asumiendo que los archivos están dentro de esta carpeta
14         String ruta1 = Paths.get(baseDir, "Lexer.flex").toString();
15         String ruta2 = Paths.get(baseDir, "LexerCup.flex").toString();
16         String[] rutaS = {"-parser", "Sintax", Paths.get(baseDir, "Sintax.cup").toString()};
17
18         generar(ruta1, ruta2, rutaS);
19     }
20
21     public static void generar(String ruta1, String ruta2, String[] rutaS) throws IOException, Exception {
22         File archivo;
23
24         // Usar JFlex para generar los archivos
25         archivo = new File(ruta1);
26         JFlex.Main.generate(archivo);
27
28         archivo = new File(ruta2);
29         JFlex.Main.generate(archivo);
30
31         // Ejecutar Java CUP para generar el parser
32         java_cup.Main.main(rutaS);
33
34         // Rutas relativas para los archivos generados
35         Path rutaSym = Paths.get("src/codigo/sym.java");
36         if (Files.exists(rutaSym)) {
37             Files.delete(rutaSym);
38         }
39
40         // Mover el archivo sym.java generado
41         Files.move(
42             Paths.get("sym.java"),
43             rutaSym
44         );
45
46         // Rutas relativas para el archivo Sintax.java
47         Path rutaSin = Paths.get("src/codigo/Sintax.java");
48         if (Files.exists(rutaSin)) {
49             Files.delete(rutaSin);
50         }
51
52         // Mover el archivo Sintax.java generado
53         Files.move(
54             Paths.get("Sintax.java"),
55             rutaSin
56         );
57     }
58 }
```



FrmPrincipal.java

El archivo FrmPrincipal.java define una interfaz gráfica de usuario (GUI) para el análisis léxico de un lenguaje, utilizando Java Swing. La clase extiende javax.swing.JFrame y se encarga de gestionar una ventana con varios componentes, como botones y áreas de texto, para realizar las funciones de análisis y visualización.

Entre sus principales funcionalidades, se incluyen la configuración de numeración de líneas en un área de texto (txtResultado), personalización de botones y actualización dinámica de los números de línea a medida que el usuario edita el texto. Además, implementa el análisis léxico de un código de entrada, donde los tokens son clasificados y mostrados en un formato estructurado, indicando el tipo de operador (aritmético, relacional, lógico, etc.). Los resultados del análisis se muestran en un área de texto específica (txtAnalizarLex). Esta clase también maneja errores de entrada y organiza los resultados de forma clara para facilitar su interpretación.

Código:

```
 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 import java.awt.Color;  
 9 import java.io.File;  
10 import java.io.FileNotFoundException;  
11 import java.io.IOException;  
12 import java.io.StringReader;  
13 import java.nio.file.Files;  
14 import java.util.logging.Level;  
15 import java.util.logging.Logger;  
16 import java_cup.runtime.Symbol;  
17 import javax.swing.JFileChooser;  
18 import javax.swing.*;  
19 import javax.swing.event.DocumentEvent;  
20 import javax.swing.event.DocumentListener;  
21 import java.awt.*;  
22 import javax.swing.border.Border;  
23  
24  
25 public class FrmPrincipal extends javax.swing.JFrame {  
26  
27     // JTextArea ya existente (txtResultado)  
28     private JTextArea linea; // Área para los números de linea  
29  
30     public FrmPrincipal() {  
31         initComponents();  
32         this.setLocationRelativeTo(null);  
33         // Configurar la numeración de líneas  
34         configurarNumeracion();  
35  
36         // Crear una lista o arreglo de botones a personalizar  
37     }  
38  
39 }
```



MANUAL DE PRÁCTICAS



```
36     // Crear una lista o arreglo de botones a personalizar
37     JButton[] botones = {btnLimpiarLex,btnArchivo,btnLimpiarSin, btnGuardar,btnCreditos, btnAyudal,btnLimpiarLex, btnLimpiarSin,btnAnalizarLex, btnAnalizarSin};
38     // Personalizar todos los botones
39     personalizarBotones(botones);
40     uniformarTamañoBotones(botones, 100, 50); // Todos tendrán un tamaño de 150x50
41 }
42
43     private void configurarNumeracion() {
44         // Crear el área de texto para los números de línea
45         lineas = new JTextArea("1");
46         // Establecer colores iniciales
47         lineas.setBackground(Color.decode("#f2e6cf")); // Fondo del área de números (modificar aquí con hex o rgba)
48         lineas.setForeground(Color.decode("#184563")); // Color de los números (modificar aquí con hex o rgba)
49         lineas.setEditable(false);
50
51         // Sincronizar el scroll de ambas áreas de texto
52         JScrollPane scrollPane = (JScrollPane) txtResultado.getParent().getParent();
53         scrollPane.setRowHeaderView(lineas);
54
55         // Agregar un DocumentListener para actualizar los números de linea
56         txtResultado.getDocument().addDocumentListener(new DocumentListener() {
57             @Override
58             public void insertUpdate(DocumentEvent e) {
59                 actualizarNumerosDeLinea();
60             }
61
62             @Override
63             public void removeUpdate(DocumentEvent e) {
64                 actualizarNumerosDeLinea();
65             }
66
67             @Override
68             public void changedUpdate(DocumentEvent e) {
69                 actualizarNumerosDeLinea();
70             }
71         });
72     }
73
74     private void actualizarNumerosDeLinea() {
75         int totalLineas = txtResultado.getLineCount();
76         StringBuilder numeros = new StringBuilder();
77         for (int i = 1; i <= totalLineas; i++) {
78             numeros.append(i).append("\n");
79         }
80         lineas.setText(numeros.toString());
81     }
82
83
84     private void analizarLexico() throws IOException{
85         int cont = 1;
86
87         String expr = (String) txtResultado.getText();
88         Lexer lexer = new Lexer(new StringReader(expr));
89         String resultado = "LINEA " + cont + "\t\t\tSÍMBOLO\n";
90
91
92
93         while (true) {
94             Tokens token = lexer.yylex();
95             if (token == null) {
96                 txtAnalizarLex.setText(resultado);
97                 return;
98             }
99         }
100        switch (token) {
101            case Linea:
102                cont++;
103                resultado += "LINEA " + cont + "\n";
104                break;
105
106                /* Operadores Aritméticos */
```

```
71     });
72 }
73
74     private void actualizarNumerosDeLinea() {
75         int totalLineas = txtResultado.getLineCount();
76         StringBuilder numeros = new StringBuilder();
77         for (int i = 1; i <= totalLineas; i++) {
78             numeros.append(i).append("\n");
79         }
80         lineas.setText(numeros.toString());
81     }
82
83
84     private void analizarLexico() throws IOException{
85         int cont = 1;
86
87         String expr = (String) txtResultado.getText();
88         Lexer lexer = new Lexer(new StringReader(expr));
89         String resultado = "LINEA " + cont + "\t\t\tSÍMBOLO\n";
90
91
92
93         while (true) {
94             Tokens token = lexer.yylex();
95             if (token == null) {
96                 txtAnalizarLex.setText(resultado);
97                 return;
98             }
99         }
100        switch (token) {
101            case Linea:
102                cont++;
103                resultado += "LINEA " + cont + "\n";
104                break;
105
106                /* Operadores Aritméticos */
```



```
108         resultado += " <Operador suma>\t\t" + lexer.lexeme + "\n";
109         break;
110     case Resta:
111         resultado += " <Operador resta>\t\t" + lexer.lexeme + "\n";
112         break;
113     case Multiplicacion:
114         resultado += " <Operador multiplicación>\t\t" + lexer.lexeme + "\n";
115         break;
116     case Division:
117         resultado += " <Operador división>\t\t" + lexer.lexeme + "\n";
118         break;
119
120     /* Operadores Relacionales */
121     case Mayor:
122         resultado += " <Operador mayor>\t\t" + lexer.lexeme + "\n";
123         break;
124     case Menor:
125         resultado += " <Operador menor>\t\t" + lexer.lexeme + "\n";
126         break;
127     case MayorIgual:
128         resultado += " <Operador mayor o igual>\t\t" + lexer.lexeme + "\n";
129         break;
130     case MenorIgual:
131         resultado += " <Operador menor o igual>\t\t" + lexer.lexeme + "\n";
132         break;
133     case Comparacion:
134         resultado += " <Operador comparación>\t\t" + lexer.lexeme + "\n";
135         break;
136     case Distinto:
137         resultado += " <Operador distinto>\t\t" + lexer.lexeme + "\n";
138         break;
139
140     /* Operadores Lógicos */
141     case Y:
142         resultado += " <Operador lógico Y>\t\t" + lexer.lexeme + "\n";
143         break;
```

```
144
145     case O:
146         resultado += " <Operador lógico O>\t\t" + lexer.lexeme + "\n";
147         break;
148     case No:
149         resultado += " <Operador lógico NO>\t\t" + lexer.lexeme + "\n";
150         break;
151
152     /* Otros Operadores */
153     case Modulo:
154         resultado += " <Operador módulo>\t\t" + lexer.lexeme + "\n";
155         break;
156     case Potencia:
157         resultado += " <Operador potencia>\t\t" + lexer.lexeme + "\n";
158         break;
159     case Raiz:
160         resultado += " <Operador raíz>\t\t" + lexer.lexeme + "\n";
161         break;
162     case Punto:
163         resultado += " <Operador punto>\t\t" + lexer.lexeme + "\n";
164         break;
165     case Asignacion:
166         resultado += " <Operador asignación>\t\t" + lexer.lexeme + "\n";
167         break;
168     case Igual:
169         resultado += " <Operador igualdad>\t\t" + lexer.lexeme + "\n";
170         break;
171     case Concatenacion:
172         resultado += " <Operador concatenación>\t\t" + lexer.lexeme + "\n";
173         break;
174     case Positivo:
175         resultado += " <Operador positivo>\t\t" + lexer.lexeme + "\n";
176         break;
177     case Negativo:
178         resultado += " <Operador negativo>\t\t" + lexer.lexeme + "\n";
179         break;
180     case Incremento:
```



```
181         break;
182     case Decremento:
183         resultado += " <Operador decremento>\t\t" + lexer.lexeme + "\n";
184         break;
185
186     /* Constantes */
187     case PI:
188         resultado += " <Constante PI>\t\t" + lexer.lexeme + "\n";
189         break;
190     case Euler:
191         resultado += " <Constante Euler>\t\t" + lexer.lexeme + "\n";
192         break;
193
194     /* Agrupadores */
195     case ParentesisApertura:
196         resultado += " <Paréntesis apertura>\t\t" + lexer.lexeme + "\n";
197         break;
198     case ParentesisCierre:
199         resultado += " <Paréntesis cierre>\t\t" + lexer.lexeme + "\n";
200         break;
201     case CorcheteApertura:
202         resultado += " <Corchete apertura>\t\t" + lexer.lexeme + "\n";
203         break;
204     case CorcheteCierre:
205         resultado += " <Corchete cierre>\t\t" + lexer.lexeme + "\n";
206         break;
207     case LlaveApertura:
208         resultado += " <Llave apertura>\t\t" + lexer.lexeme + "\n";
209         break;
210     case LlaveCierre:
211         resultado += " <Llave cierre>\t\t" + lexer.lexeme + "\n";
212         break;
213
214     /* Otros Tokens */
215     case InicioTexto:
216         resultado += " <Inicio de texto>\t\t" + lexer.lexeme + "\n";
```

```
216         resultado += " <Inicio de texto>\t\t" + lexer.lexeme + "\n";
217         break;
218     case FinalTexto:
219         resultado += " <Final de texto>\t\t" + lexer.lexeme + "\n";
220         break;
221     case ComillaDoble:
222         resultado += " <Comilla doble>\t\t" + lexer.lexeme + "\n";
223         break;
224     case ComillaSimple:
225         resultado += " <Comilla simple>\t\t" + lexer.lexeme + "\n";
226         break;
227     case Comentario:
228         resultado += " <Comentario>\t\t" + lexer.lexeme + "\n";
229         break;
230     case InicioComentario:
231         resultado += " <Inicio de comentario>\t\t" + lexer.lexeme + "\n";
232         break;
233     case FinComentario:
234         resultado += " <Fin de comentario>\t\t" + lexer.lexeme + "\n";
235         break;
236     case Entero:
237         resultado += " <Tipo de dato Entero>\t\t" + lexer.lexeme + "\n";
238         break;
239     case Decimal:
240         resultado += " <Tipo de dato Decimal>\t\t" + lexer.lexeme + "\n";
241         break;
242     case Verdadero:
243         resultado += " <Booleano verdadero>\t\t" + lexer.lexeme + "\n";
244         break;
245     case Falso:
246         resultado += " <Booleano falso>\t\t" + lexer.lexeme + "\n";
247         break;
248     case Cadena:
249         resultado += " <Cadena>\t\t" + lexer.lexeme + "\n";
250         break;
```



```
252         case Si:
253             resultado += " <Palabra reservada 'Si'>\t\t" + lexer.lexeme + "\n";
254             break;
255     case SiNo:
256         resultado += " <Palabra reservada 'SiNo'>\t\t" + lexer.lexeme + "\n";
257         break;
258     case SiNoSi:
259         resultado += " <Palabra reservada 'SiNoSi'>\t\t" + lexer.lexeme + "\n";
260         break;
261     case CicloRepite:
262         resultado += " <Palabra reservada 'CicloRepite'>\t\t" + lexer.lexeme + "\n";
263         break;
264     case Mientras:
265         resultado += " <Palabra reservada 'Mientras'>\t\t" + lexer.lexeme + "\n";
266         break;
267     case Hacer:
268         resultado += " <Palabra reservada 'Hacer'>\t\t" + lexer.lexeme + "\n";
269         break;
270     case Seleccionador:
271         resultado += " <Palabra reservada 'Seleccionador'>\t\t" + lexer.lexeme + "\n";
272         break;
273     case Caso:
274         resultado += " <Palabra reservada 'Caso'>\t\t" + lexer.lexeme + "\n";
275         break;
276     case Predeterminado:
277         resultado += " <Palabra reservada 'Predeterminado'>\t\t" + lexer.lexeme + "\n";
278         break;
279     case Detener:
280         resultado += " <Palabra reservada 'Detener'>\t\t" + lexer.lexeme + "\n";
281         break;
282     case Arreglo:
283         resultado += " <Palabra reservada 'Arreglo'>\t\t" + lexer.lexeme + "\n";
284         break;
285     case Funcion:
286         resultado += " <Palabra reservada 'Funcion'>\t\t" + lexer.lexeme + "\n";
287         break;

288     case Procedimiento:
289         resultado += " <Palabra reservada 'Procedimiento'>\t\t" + lexer.lexeme + "\n";
290         break;
291     case Principal:
292         resultado += " <Palabra reservada 'Principal'>\t\t" + lexer.lexeme + "\n";
293         break;
294     case Biblioteca:
295         resultado += " <Palabra reservada 'Biblioteca'>\t\t" + lexer.lexeme + "\n";
296         break;
297     case Imprimir:
298         resultado += " <Palabra reservada 'Imprimir'>\t\t" + lexer.lexeme + "\n";
299         break;
300     case Leer:
301         resultado += " <Palabra reservada 'Leer'>\t\t" + lexer.lexeme + "\n";
302         break;
303     case Importar:
304         resultado += " <Palabra reservada 'Importar'>\t\t" + lexer.lexeme + "\n";
305         break;
306     case Retornar:
307         resultado += " <Palabra reservada 'Retornar'>\t\t" + lexer.lexeme + "\n";
308         break;
309     case Metodo:
310         resultado += " <Palabra reservada 'Metodo'>\t\t" + lexer.lexeme + "\n";
311         break;
312     case Nulo:
313         resultado += " <Palabra reservada 'Nulo'>\t\t" + lexer.lexeme + "\n";
314         break;
315
316     case CaracteresEspeciales:
317         resultado += " <Caracter especial>\t\t" + lexer.lexeme + "\n";
318         break;
319     case P_coma:
320         resultado += " <Punto y coma>\t\t" + lexer.lexeme + "\n";
321         break;
322     case Main:
323         resultado += " <Palabra reservada 'Main'>\t\t" + lexer.lexeme + "\n";
```



```
324         break;
325     case Constante:
326         resultado += " <Constante>\t\t\t" + lexer.lexeme + "\n";
327         break;
328     case NombreClase:
329         resultado += " <Nombre de la Clase>\t\t" + lexer.lexeme + "\n";
330         break;
331     case Identificador:
332         resultado += " <Identificador>\t\t" + lexer.lexeme + "\n";
333         break;
334     case Numero:
335         resultado += " <Número>\t\t" + lexer.lexeme + "\n";
336     case Clase:
337         resultado += " <Palabra reservada 'Clase'>\t\t" + lexer.lexeme + "\n";
338         break;
339     case Texto:
340         resultado += " <Texto>\t\t\t" + lexer.lexeme + "\n";
341         break;
342     /*case Constante:
343         resultado += " <Constante>\t\t\t" + lexer.lexeme + "\n";
344         break;
345     */
346
347     case ERROR:
348         resultado += " <Símbolo no definido>\n";
349         break;
350     default:
351         resultado += " <" + lexer.lexeme + ">\n";
352         break;
353     }
354   }
355 }
356
357 /**
358 * This method is called from within the constructor to initialize the form.
359 * WARNING: Do NOT modify this code. The content of this method is always
```

```
362     @SuppressWarnings("unchecked")
363     Generated Code
364
365     private void btnArchivoActionPerformed(java.awt.event.ActionEvent evt) {
366
367         // TODO add your handling code here:
368         JFileChooser chooser = new JFileChooser();
369         chooser.showOpenDialog(null);
370         File archivo = new File(chooser.getSelectedFile().getAbsolutePath());
371
372         try {
373             String ST = new String(Files.readAllBytes(archivo.toPath()));
374             txtResultado.setText(ST);
375         } catch (FileNotFoundException ex) {
376             Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
377         } catch (IOException ex) {
378             Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
379         }
380     }
381
382     private void btnLimpiarLexActionPerformed(java.awt.event.ActionEvent evt) {
383
384         // TODO add your handling code here:
385         txtAnalizarLex.setText(null);
386     }
387
388     private void btnLimpiarSinActionPerformed(java.awt.event.ActionEvent evt) {
389
390         // TODO add your handling code here:
391         txtAnalizarSin.setText(null);
392     }
393
394     private void btnAnalizarLexActionPerformed(java.awt.event.ActionEvent evt) {
395
396         try {
397             analizarLexico();
398         } catch (IOException ex) {
399             Logger.getLogger(FrmPrincipal.class.getName()).log(Level.SEVERE, null, ex);
400         }
401     }
```



```
617 }
618
619     private void btnAnalizarSinActionPerformed(java.awt.event.ActionEvent evt) {
620         String ST = txtResultado.getText();
621         Sintax s = new Sintax(new codigo.LexerCup(new StringReader(ST)));
622         s.setTextoCompleto(ST); // Pasa el texto completo al parser
623
624     try {
625         s.parse();
626         if (s.getErrores().isEmpty()) {
627             txtAnalizarSin.setText("Análisis realizado correctamente");
628             txtAnalizarSin.setForeground(new Color(25, 111, 61));
629         } else {
630             StringBuilder errores = new StringBuilder();
631             for (String error : s.getErrores()) {
632                 errores.append(error).append("\n");
633             }
634             txtAnalizarSin.setText(errores.toString());
635             txtAnalizarSin.setForeground(Color.red);
636         }
637     } catch (Exception ex) {
638         txtAnalizarSin.setText("Error inesperado durante el análisis.");
639         txtAnalizarSin.setForeground(Color.red);
640     }
641 }
642
643     private void btnLimpiarLex1ActionPerformed(java.awt.event.ActionEvent evt) {
644         txtResultado.setText(null);
645     }
646
647     private void btnCreditosActionPerformed(java.awt.event.ActionEvent evt) {
648         // Crear una instancia de la clase Ayuda y mostrarla
649         Creditos2 ventanaCreditos = new Creditos2();
650         ventanaCreditos.setVisible(true);
651 }
```

```
654     private void btnAyudalActionPerformed(java.awt.event.ActionEvent evt) {
655         Ayuda ventanaAyuda = new Ayuda();
656         ventanaAyuda.setVisible(true);
657     }
658
659     private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
660         // Crear un JFileChooser para seleccionar la ubicación donde guardar el archivo
661         JFileChooser fileChooser = new JFileChooser();
662         fileChooser.setDialogTitle("Guardar archivo");
663
664         // Configurar el filtro para que solo permita guardar archivos .txt
665         fileChooser.setFileFilter(new javax.swing.filechooser.FileNameExtensionFilter("Archivos de texto (*.txt)", "txt"));
666
667         // Abrir el cuadro de diálogo y verificar si el usuario seleccionó "Guardar"
668         int userSelection = fileChooser.showSaveDialog(this);
669
670         if (userSelection == JFileChooser.APPROVE_OPTION) {
671             // Obtener la ruta del archivo seleccionado
672             java.io.File fileToSave = fileChooser.getSelectedFile();
673
674             // Asegurarse de que el archivo tenga la extensión .txt
675             String filePath = fileToSave.getAbsolutePath();
676             if (!filePath.endsWith(".txt")) {
677                 filePath += ".txt";
678             }
679
680             // Guardar el contenido del JTextArea en el archivo
681             try (java.io.FileWriter writer = new java.io.FileWriter(filePath)) {
682                 writer.write(txtResultado.getText());
683                 JOptionPane.showMessageDialog(this, "Archivo guardado exitosamente en: " + filePath);
684             } catch (java.io.IOException e) {
685                 JOptionPane.showMessageDialog(this, "Error al guardar el archivo: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
686             }
687         }
688     }
```



```
691 */  
692 public static void main(String args[]) {  
693     /* Set the Nimbus look and feel */  
694     // Look and feel setting code (optional)  
695  
696     java.awt.EventQueue.invokeLater(new Runnable() {  
697         public void run() {  
698             FrmPrincipal frame = new FrmPrincipal();  
699             frame.setVisible(true);  
700  
701             // Configurar numeración de líneas  
702             frame.configurarNumeracion();  
703         }  
704     });  
705 }  
706  
707 // Variables declaration - do not modify  
708 private javax.swing.JButton btnAnalizarLex;  
709 private javax.swing.JButton btnAnalizarSin;  
710 private javax.swing.JButton btnArchivo;  
711 private javax.swing.JButton btnAyudal;  
712 private javax.swing.JButton btnCreditos;  
713 private javax.swing.JButton btnGuardar;  
714 private javax.swing.JButton btnLimpiarLex;  
715 private javax.swing.JButton btnLimpiarLex1;  
716 private javax.swing.JButton btnLimpiarSin;  
717 private javax.swing.JPanel jPanel1;  
718 private javax.swing.JPanel jPanel2;  
719 private javax.swing.JScrollPane jScrollPane1;  
720 private javax.swing.JScrollPane jScrollPane2;  
721 private javax.swing.JScrollPane jScrollPane3;  
722 private javax.swing.JTextArea txtAnalizarLex;  
723 private javax.swing.JTextArea txtAnalizarSin;  
724 private javax.swing.JTextArea txtResultado;  
725 // End of variables declaration  
726
```

```
760 private void personalizarBoton(JButton boton) {  
761     // Colores personalizados  
762     Color buttonColor = Color.decode("#f2e6cf"); // Fondo inicial  
763     Color hoverColor = Color.decode("#f2e6cf"); // Fondo al pasar el mouse  
764     Color textColor = Color.WHITE; // Color del texto  
765     Color borderColor = Color.WHITE; // Color del borde  
766  
767     // Personalizar fuente y estilo  
768     boton.setFont(new Font("Arial", Font.BOLD, 14));  
769     boton.setFocusPainted(false);  
770     boton.setContentAreaFilled(false); // Para permitir un fondo personalizado  
771     boton.setOpaque(false); // Fondo transparente para efectos de bordes  
772     boton.setForeground(textColor);  
773  
774     // Aplicar borde redondeado  
775     boton.setBorder(new RoundedBorder(20)); // Radio de las esquinas  
776     boton.setBackground(buttonColor);  
777  
778     // Cambiar color al pasar el mouse  
779     boton.addMouseListener(new java.awt.event.MouseAdapter() {  
780         @Override  
781         public void mouseEntered(java.awt.event.MouseEvent evt) {  
782             boton.setForeground(Color.BLACK); // Opcional: Cambiar el color del texto al pasar el mouse  
783             boton.setBackground(hoverColor);  
784         }  
785  
786         @Override  
787         public void mouseExited(java.awt.event.MouseEvent evt) {  
788             boton.setForeground(textColor);  
789             boton.setBackground(buttonColor);  
790         }  
791     });  
792 }
```



Creditos.java

Este es un archivo para mi aplicación que se encarga de mostrar los créditos en una ventana gráfica. Lo hice con Java Swing, y su propósito es simplemente darle un toque más personal y profesional a mi proyecto.

La ventana tiene un diseño sencillo, pero creo que bien organizado. Al abrirla, muestra un título grande que dice "Créditos" y, debajo, aparece mi nombre. También incluye un botón que dice "Regresar", el cual cierra la ventana cuando lo presionas. Todo está centrado para que se vea bien y no se sienta desordenado.

Además, le agregué un fondo con una imagen llamativa para que no se vea tan simple. Aunque es algo básico, creo que cumple bien con su función de dar un pequeño espacio en mi aplicación para reconocer a los que trabajaron en ella o, en este caso, mostrar mi nombre como creador.

Código:

```
5  package codigo;
6  import javax.swing.*;
7  import java.awt.*;
8
9  public class Creditos extends javax.swing.JFrame {
10
11    public Creditos() {
12      initComponents();
13    }
14
15    private void initComponents() {
16
17      // Crear un JLabel para mostrar el texto "Créditos"
18      JLabel lblCreditos = new JLabel("Créditos");
19      lblCreditos.setFont(new Font("Arial", Font.BOLD, 36)); // Cambiar tipo de letra y tamaño
20      lblCreditos.setForeground(Color.WHITE); // Cambiar color del texto
21      lblCreditos.setHorizontalAlignment(SwingConstants.CENTER); // Centrar el texto
22
23      // Crear un JLabel para mostrar tu nombre
24      JLabel lblNombre = new JLabel("Vanesa Hernández Martínez");
25      lblNombre.setFont(new Font("Arial", Font.PLAIN, 18)); // Cambiar tipo de letra y tamaño
26      lblNombre.setForeground(Color.WHITE); // Cambiar color
27      lblNombre.setHorizontalAlignment(SwingConstants.CENTER); // Centrar el texto
28
29      // Crear un JPanel con GridBagLayout para centrar los componentes
30      JPanel panel = new JPanel();
31      panel.setLayout(new GridBagLayout());
32      panel.setOpaque(false); // Hacer que el panel sea transparente para que se vea el fondo
33
34      GridBagConstraints gbc = new GridBagConstraints();
35      gbc.gridx = 0; // Establecer la posición horizontal
36      gbc.gridy = 0; // Establecer la posición vertical
37      gbc.insets = new Insets(10, 10, 10, 10); // Espaciado alrededor de los componentes
38      panel.add(lblCreditos, gbc); // Agregar el texto "Créditos"
39
40      gbc.gridy = 1; // Cambiar la posición vertical para el siguiente texto
```



MANUAL DE PRÁCTICAS



```
42      // Crear el botón "Regresar"
43      JButton btnRegresar = new JButton("Regresar");
44      btnRegresar.setFont(new Font("Arial", Font.PLAIN, 18)); // Cambiar tipo de letra y tamaño
45      btnRegresar.setForeground(Color.WHITE); // Cambiar color del texto
46      btnRegresar.setBackground(Color.DARK_GRAY); // Cambiar el color de fondo del botón
47      btnRegresar.setFocusPainted(false); // Eliminar el borde de enfoque
48      btnRegresar.setHorizontalTextPosition(SwingConstants.CENTER); // Centrar el texto del botón
49      // Acción del botón "Regresar"
50      btnRegresar.addActionListener(e -> {
51          // Cerrar solo esta ventana (Creditos2)
52          this.dispose();
53      });
54      // Colocar el botón en el panel
55      gbc.gridx = 2; // Colocamos el botón en la posición debajo del nombre
56      panel.add(btnRegresar, gbc);
57      // Crear un JLabel para la imagen de fondo
58      JLabel lblFondo = new JLabel(new ImageIcon(getClass().getResource("/icons/poligonos-abstractos_2560x1440_xtrafondos.com.jpg")));
59      lblFondo.setLayout(new BorderLayout());
60      // Agregar el JPanel con los textos al JLabel con el fondo
61      lblFondo.add(panel, BorderLayout.CENTER); // Centrar el panel dentro del fondo
62      // Configurar la ventana
63      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64      setSize(512, 500); // Tamaño de la ventana
65      setLocationRelativeTo(null); // Centrar la ventana
66      add(lblFondo); // Agregar el fondo con los textos y el botón a la ventana
67      setVisible(true); // Mostrar la ventana
68  }
69 }
70 public static void main(String args[]) {
71     java.awt.EventQueue.invokeLater(new Runnable() {
72         public void run() {
73             new Creditos().setVisible(true);
74         }
75     });
76 }
77 }
```



Ayuda.java

El código utiliza la biblioteca javax.swing para crear una interfaz gráfica de usuario (GUI). En esta interfaz se crean tres botones, los cuales tienen diferentes funcionalidades según el botón que se presione. Además, se incluye una imagen de fondo para la ventana principal. Los botones están alineados verticalmente, y cada uno de ellos abre una ventana separada cuando es clickeado.

Código:

```
package codigo;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Ayuda extends JPanel {

    public Ayuda() {
        initComponents();
    }

    private void initComponents() {
        // Crear un panel personalizado con imagen de fondo
        jPanell = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                // Cargar la imagen de fondo
                ImageIcon background = new ImageIcon(getClass().getResource("/icons/fondo.jpg"));
                g.drawImage(background.getImage(), 0, 0, getWidth(), getHeight(), this);
            }
        };

        // Configurar el layout del panel para que los botones se alineen verticalmente
        jPanell.setLayout(new BoxLayout(jPanell, BoxLayout.Y_AXIS)); // Diseño vertical
        jPanell.setAlignmentX(Component.CENTER_ALIGNMENT); // Alinear al centro

        // Crear los botones y configurarles el estilo
        jButtonManual = createStyledButton("Manual de Usuario");
        jButtonTokens = createStyledButton("Tokens");
        jButtonEstructuras = createStyledButton("Estructuras");

        // Añadir acción a los botones
        jButtonManual.addActionListener(e -> mostrarVentanaConImagen("/icons/manual.png"));
        jButtonTokens.addActionListener(e -> mostrarVentanaConImagen("/icons/tokens.png"));
    }
}
```



```
38     jButtonEstructuras.addActionListener(e -> mostrarVentanaConImagen("/icons/estructuras.png"));
39
40     // Añadir los botones al panel
41     jPanell.add(Box.createRigidArea(new Dimension(0, 50))); // Espacio antes del primer botón
42     jPanell.add(jButtonManual);
43     jPanell.add(Box.createRigidArea(new Dimension(0, 20))); // Espacio entre botones
44     jPanell.add(jButtonTokens);
45     jPanell.add(Box.createRigidArea(new Dimension(0, 20))); // Espacio entre botones
46     jPanell.add(jButtonEstructuras);
47
48     // Configurar el layout principal
49     setLayout(new BorderLayout());
50     add(jPanell, BorderLayout.CENTER); // Panel de botones
51 }
52
53     JButton createStyledButton(String text) {
54         JButton button = new JButton(text);
55         button.setFont(new Font("Arial", Font.BOLD, 16));
56         button.setBackground(new Color(7, 153, 182)); //
57         button.setForeground(Color.WHITE);
58         button.setFocusPainted(false);
59         button.setPreferredSize(new Dimension(200, 50));
60         button.setMaximumSize(new Dimension(200, 50)); // Tamaño máximo
61         button.setAlignmentX(Component.CENTER_ALIGNMENT); // Centrar los botones
62         button.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10)); // Espacio interno
63         button.setCursor(new Cursor(Cursor.HAND_CURSOR)); // Cambio de cursor
64         button.setOpaque(true);
65         button.setBorderPainted(false);
66         button.setToolTipText("Haz clic para " + text); // Tooltip
67         button.setRolloverEnabled(true); // Habilitar efecto hover
68         button.addActionListener(e -> button.setBackground(new Color(156, 210, 201))); // Efecto hover
69         return button;
70     }
71
72     private void mostrarVentanaConImagen(String rutaImagen) {
73         // Crear una nueva ventana (JFrame) para mostrar la imagen
```

```
74     JFrame ventanaImagen = new JFrame("Program-Art");
75     ventanaImagen.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
76
77     // Obtener el tamaño de la pantalla
78     Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
79
80     // Establecer el tamaño de la ventana para que ocupe el 90% de la pantalla
81     int width = (int) (screenSize.width * 0.9);
82     int height = (int) (screenSize.height * 0.9);
83     ventanaImagen.setSize(width, height); // Establecer el tamaño de la ventana
84     ventanaImagen.setLocationRelativeTo(null); // Centrar la ventana
85
86     // Crear un JLabel para mostrar la imagen
87     JLabel imagenLabel = new JLabel();
88     ImageIcon imageIcon = new ImageIcon(getClass().getResource(rutaImagen));
89     Image image = imageIcon.getImage(); // Obtener la imagen
90     Image scaledImage = image.getScaledInstance(width, height, Image.SCALE_SMOOTH); // Escalar imagen
91     imagenLabel.setIcon(new ImageIcon(scaledImage));
92
93     // Añadir el JLabel con la imagen a la nueva ventana
94     ventanaImagen.add(imagenLabel);
95     ventanaImagen.setVisible(true);
96 }
97
98     public static void main(String[] args) {
99         // Crear el frame principal para ejecutar la aplicación
100        JFrame frame = new JFrame("Ayuda");
101        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
102        frame.setSize(600, 400);
103        frame.setLocationRelativeTo(null); // Centrar la ventana
104        frame.setContentPane(new Ayuda()); // Añadir el panel principal
105        frame.setVisible(true);
106    }
107
108    // Variables de instancia
```

```
108    // Variables de instancia
109    private JPanel jPanell;
110    private JButton jButtonManual;
111    private JButton jButtonTokens;
112    private JButton jButtonEstructuras;
113 }
114
```



Pantallas de ejecución

FrmPrincipal.java





Ejemplo de un código con declaraciones:

Analizador Lexico

Botones: Abrir archivo, Analizar, Creditos, Salir.

```
1 ent principal(){}
2 n = "Vane";
3 }
```

LINEA	SIMBOLO
1	ent
1	principal
1	(
2)
2	{
3	n
3	=
3	"Vane"
3	:
3	}

Analizador Sintactico

Botones: Analizar, Salir.

Análisis realizado correctamente



Ejemplo de un código con if:

Analizador Lexico

Creditos

Abrir archivo Analizar

SIMBOL	LINEA 1	LINEA 2	LINEA 3
ent	<Tipo de dato Entero>	<Palabra reservada 'Si'>	<Palabra reservada 'Imprimir'>
principal	<Palabra reservada 'Principal'>	<Paréntesis apertura>	<Paréntesis apertura>
(<Paréntesis cierre>	<Identificador>	<Paréntesis cierre>
)	<Llave apertura>	<Operador módulo>	<Número>
{		<Operador comparación>	<Número>
		<Número>	<Operador comparación>
		<Paréntesis cierre>	<Número>
		<Llave apertura>	<Número>
			<Texto>

Analizador Sintactico

Anализar

Análisis realizado correctamente



Ejemplo de un código con ifElse:

Analizador Lexico

Analizar

Creditos

Abrir archivo

Analizador Sintactico

Análisis realizado correctamente

Entorno de desarrollo integrado (IDE) que muestra dos ventanas principales: "Analizador Lexico" y "Analizador Sintactico".

En la ventana "Analizador Lexico":

- Botones: "Abrir archivo", "Analizar", "Creditos".
- Área de texto:

```
1 ent principal(){  
2 si (edad < 12 & edad < 18) {  
3   imprime ("Eres un niño");  
4 }siNo{  
5   imprime ("Eres un adulto");  
6 }  
7 }
```
- Área de análisis:

LÍNEA	SÍMBOLO
1	ent
1	principal
1	{
2	si
2	(
2	edad
2	<
2	12
2	&
2	edad
2	<
2	18
2)
3	{
4	imprime

En la ventana "Analizador Sintactico":

- Botón: "Analizar".
- Área de texto: "Análisis realizado correctamente".



Ejemplo de un código con ifElse if:

Analizador Lexico

Analizar

Creditos

Abrir archivo

Analizador Sintactico

Análisis realizado correctamente

Entorno de desarrollo integrado (IDE) que muestra el análisis léxico y sintáctico de un código en pseudopseudo-lenguaje.

Analizador Lexico (Parte superior):

- Entrada:** 1 ent principal(){
2 si (edad < 12) {
3 imprime ("Eres un niño");
4 } siNo si (edad > 12){
5 imprime ("Eres un adolescente");
6 }siNo{
7 imprime ("Eres un adulto");
8 }
9 }
- Salida:** LINEA 1:
<Tipo de dato Entero>
<Palabra reservada 'Principal'>
<Paréntesis apertura>
<Paréntesis cierre>
<Llave apertura>
LINEA 2:
<Palabra reservada 'Si'>
<Paréntesis apertura>
<Identificador>
<Operador menor>
<Número>
<Paréntesis cierre>
<Llave apertura>
LINEA 3:
<Palabra reservada 'Imprimir'>
<Paréntesis apertura>
<Texto>
<Paréntesis cierre>
<Punto y coma>
- Símbolos:** ent, principal, (,), {, }, si, (,), edad, <, 12, {, }, imprime, (,), "Eres un",), ;

Analizador Sintactico (Parte inferior):

- Botones:** Analizar, Borrar.
- Mensaje:** Análisis realizado correctamente.



Ejemplo de un código con While:

Analizador Lexico

Abrir archivo Analizar

```
1 ent principal(){}
2 mientras (contador < 5) {
3     imprime ("Contador: ") con contador;
4     contador++;
5 }
6 }
```

SIMBOL	LINEA 1	LINEA 2	LINEA 3
ent	<Tipo de dato Entero>	<Palabra reservada 'Mientras'>	<Palabra reservada 'Imprimir'>
principal	<Paréntesis apertura>	<Paréntesis apertura>	<Paréntesis apertura>
(<Paréntesis cierre>	<Identificador>	<Paréntesis cierre>
)	<Llave apertura>	<Operador menor>	<Número>
{	LINEA 2	<Paréntesis cierre>	<Paréntesis cierre>
	<Palabra reservada 'Principal'>	<Llave apertura>	<Operador concatenación>
	<Paréntesis cierre>	<Identificador>	
	<Llave apertura>	<Operador menor>	
	LINEA 3	<Número>	
	<Palabra reservada 'Contador'>	<Paréntesis cierre>	
	<Paréntesis apertura>	<Operador concatenación>	
	<Texto>		
	<Paréntesis cierre>		
	<Operador concatenación>		

Analizador Sintactico

Analizar

Análisis realizado correctamente



Ejemplo de un código con do While:

Analizador Lexico

Abrir archivo Analizar

```
1 ent principal(){  
2   hacer {  
3     imprime ("Valor actual de i: ") con i;  
4     i++;  
5   } mientras (i < 5);  
6 }
```

LINEA 1
<Tipo de dato Entero>
<Palabra reservada 'Principal'>
<Paréntesis apertura>
<Paréntesis cierre>
<Llave apertura>
LINEA 2
<Palabra reservada 'Hacer'>
<Llave apertura>
LINEA 3
<Palabra reservada 'Imprimir'>
<Paréntesis apertura>
<Texto>
<Paréntesis cierre>
<Operador concatenación>
<Identificador>
<Punto y coma>
LINEA 4
<Identificador>
<Operador incremento>

Analizador Sintactico

Analizar

Análisis realizado correctamente



Ejemplo de un código con For:

The screenshot shows two windows side-by-side. The top window is titled "Analizador Lexico" and displays a code editor with the following C-like pseudocode:

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

To the right of the code editor is a detailed parse tree or symbol table mapping tokens to their corresponding symbols and types. The bottom window is titled "Analizador Sintactico" and contains a message: "Análisis realizado correctamente".

LÍNEA	SÍMBOLOS	
LINEA 1	ent principal () {	
LINEA 2	<Palabra reservada 'CicloRepite'> <Paréntesis apertura> <Tipo de dato Entero> <Identificador> <Operador menor o igual> <Número> <Punto y coma> <Identificador> <Operador menor> <Número> <Punto y coma> <Operador decremento> <Identificador>	(ent i = 0 ; i ^ 10 ; - i



Ejemplo de un código con métodos:

Analizador Lexico

Este panel muestra el código fuente y su análisis léxico:

```
1 ent principal() {  
2 metodo saludo(cadena nombre) {  
3     imprime("Hola, " con nombre);  
4 }  
5 }
```

LINEA 1	SIMBOLOS
<Tipo de dato Entero>	ent
<Palabra reservada 'Principal'>	principal
<Paréntesis apertura>	(
<Paréntesis cierre>)
<Llave apertura>	{
LINEA 2	
<Palabra reservada 'CicloRepite'>	
<Paréntesis apertura>	(
<Tipo de dato Entero>	ent
<Identificador>	i
<Operador menor o igual>	<=
<Número>	0
<Punto y coma>	;
<Identificador>	i
<Operador menor>	<
<Número>	10
<Punto y coma>	;
<Operador decremento>	--
<Identificador>	i

Analizador Sintactico

Este panel muestra el resultado del análisis sintáctico:

Análisis realizado correctamente



Ejemplo de un código con Operaciones Aritméticas:

Analizador Lexico

Abrir archivo Analizar

```
1 ent principal(){  
2   r = 3+5;  
3 }
```

LINEA	SIMBOLO
LINEA 1	ent
<Tipo de dato Entero>	principal
<Palabra reservada 'Principal'>	(
<Paréntesis apertura>)
<Paréntesis cierre>	{
<Llave apertura>	r
LINEA 2	=
<Identificador>	3
<Operador igualdad>	+
<Número>	5
<Operador suma>	;
<Número>	}
LINEA 3	
<Punto y coma>	
<Llave cierre>	

Analizador Sintactico

Analizar

Análisis realizado correctamente



Ejemplo de un código que imprime valores:

The screenshot shows two windows side-by-side. The top window is titled "Analizador Lexico" and displays a code editor with the following C-like pseudocode:

```
1 ent principal() {
2     imprime("Hola, " con nombre);
3 }
```

Below the code editor, the output is shown in three columns:

LÍNEA	SÍMBOLO
LINEA 1	ent
<Tipo de dato Entero>	principal
<Palabra reservada 'Principal'>	(
<Paréntesis apertura>)
<Paréntesis cierre>	{
<Llave apertura>	imprime
LINEA 2	(
<Palabra reservada 'Imprimir'>	"Hola, "
<Paréntesis apertura>	con
<Texto>	nombre
<Operador concatenación>)
<Identificador>	;
<Paréntesis cierre>	}
<Punto y coma>	
LINEA 3	
<Llave cierre>	

The bottom window is titled "Analizador Sintactico" and contains a button labeled "Analizar". Below it, a message box displays the text: "Análisis realizado correctamente".



Ejemplo de un código con comentarios:

Analizador Lexico

Abrir archivo Eliminar Analizar Creditos

```
1 ent principal(){}
2 # Esto es un comentario
3 ent edad => 10;
4 }
```

LINEA	SIMBOLO
1	ent
1	principal
1	(
1)
1	{
2	# Esto es u
3	ent
3	edad
3	=>
3	10
3	;
4	}

Analizador Sintactico

Analizar Eliminar

Análisis realizado correctamente



Ejemplo de un código con errores:

Analizador Lexico

Abrir archivo Analizar

```
1 ent principal(){  
2 # Esto es un comentario  
3 ent edad <= 10;  
4 }
```

LINEA	SIMBOLO
1	ent
1	principal
1	(
1)
1	{
2	# Esto es u
3	ent
3	edad
3	=>
3	10
3	,
4	}

Analizador Sintactico

Analizar

Error de sintaxis. Línea: 3, Columna: 11, Texto: "<"



Ejemplo de un código con varios errores:

Analizador Lexico

Este panel muestra el código fuente y las tokens correspondientes:

```
1 ent principal() {  
2     ent x;  
3     ent y = 5.0;  
4     x++;  
5     dec y = 5.6  
6     dec x++;  
7     dec x--;  
8     cadena x <= "hola";  
9     cadena y = "adios";  
10      
11    si (numero == "hola") {  
12        imprim ("El número es par");  
13    }  
14      
15    repite (ent i < 0 ; i < 10; i+) {  
16        imprime(i);  
17    }  
18}
```

Tokenización:

```
<Paréntesis cierre> )  
<Llave apertura> {  
LINEA 12  
<Identificador> imprim  
(  
<Paréntesis apertura> "  
<Texto> "El númer  
<Paréntesis cierre> )  
<Punto y coma> ;  
LINEA 13  
<Llave cierre> }  
LINEA 14  
<Comentario> # For  
LINEA 15  
<Palabra reservada 'CicloRepite'> (  
<Paréntesis apertura> ent  
<Tipo de dato Entero> i  
<Identificador> <= 0  
<Operador menor o igual> ;  
<Número> 0  
<Punto y coma> ;  
<Identificador> i
```

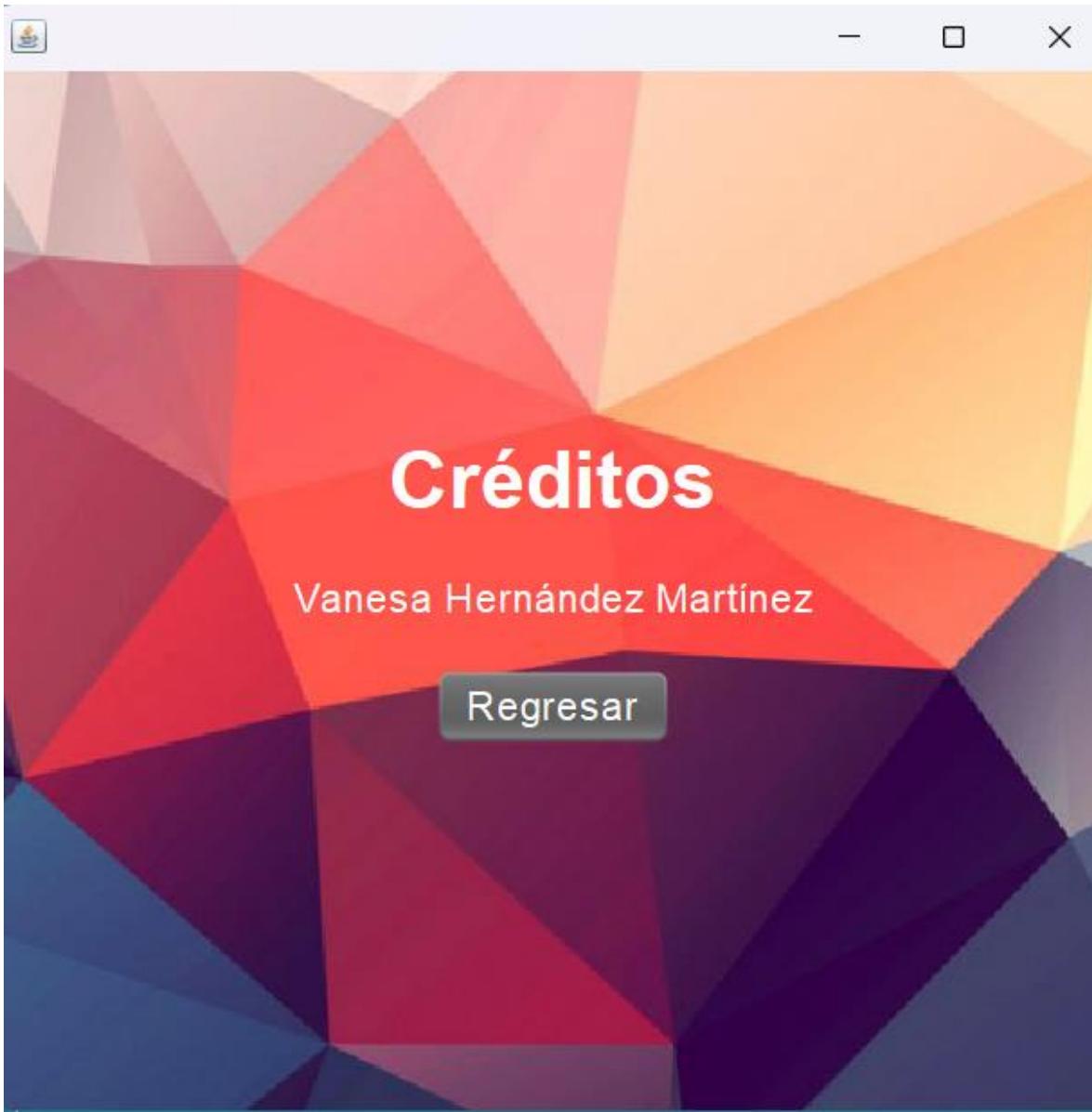
Analizador Sintactico

Este panel muestra los errores de sintaxis:

```
Error de sintaxis. Línea: 3, Columna: 15, Texto: "."
Error de sintaxis. Línea: 6, Columna: 6, Texto: "dec"
Error de sintaxis. Línea: 8, Columna: 16, Texto: "<"
Error de sintaxis. Línea: 12, Columna: 13, Texto: "("
Error de sintaxis. Línea: 15, Columna: 31, Texto: "+"
```

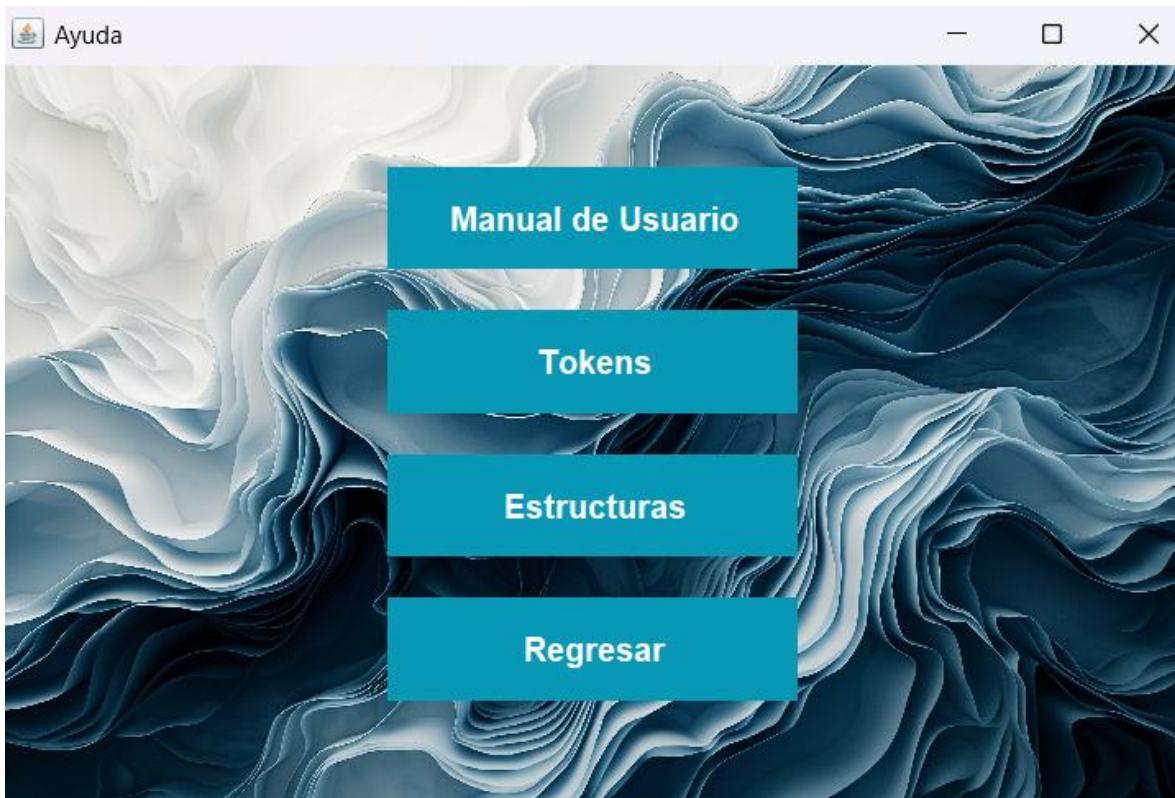


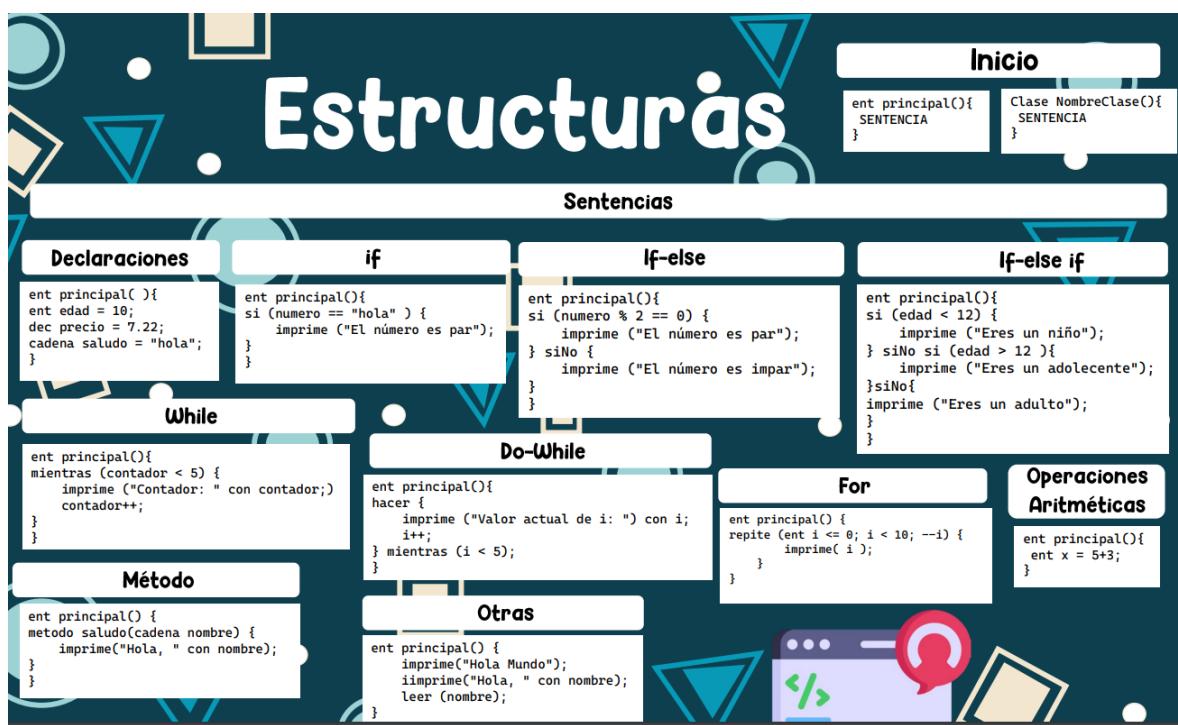
Creditos.java





Ayuda.java





Conclusiones

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

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

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

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

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

Considero que logre un buen resultado porque la final es un compilador funcional tiene una variada cantidad de tokens y así mismo de sentencias que ya se pueden identificar, creo que este proyecto me ayudo a comprender perfectamente que es lo que realiza el análisis léxico y que es lo que realiza el análisis sintáctico y la importancia de tener bien elaborado un analizador léxico antes del sintáctico.

Este proyecto implemento conocimientos de muchas otras unidades que vimos en la materia de Lenguajes y Autómatas I, por ejemplo, de la unidad 1 se relaciona porque utilizamos un alfabeto, cadenas y aplicamos la teoría de los lenguajes formales. De la unidad 2 de expresiones regulares esas nos fueron de gran ayuda para crear patrones más complejos con símbolos del alfabeto. De la unidad 3 aunque no diseñamos como tal un lenguaje autómata finito a mano, las computadoras y las librerías hicieron lo suyo para procesar estos diagramas y minimizar los estados si era necesario. Y de la unidad 6 de máquinas de Turing, aunque funcionan diferente también se relacionan en cuanto al concepto de lenguaje y a sus elementos.

La materia de Lenguajes y Autómatas es fundamental en Ingeniería en Sistemas porque proporciona los conocimientos necesarios para comprender cómo se diseñan y estructuran los lenguajes de programación. Estudia las reglas y principios que permiten a una computadora interpretar y procesar el código que escribimos, lo cual es clave en el desarrollo de software.

Además, introduce conceptos como los autómatas, que son modelos matemáticos esenciales para resolver problemas relacionados con el análisis y procesamiento de datos, como la validación de entradas, la búsqueda de patrones o el diseño de sistemas más eficientes.

En términos prácticos, esta materia te prepara para crear y mejorar lenguajes de programación, desarrollar compiladores e intérpretes, y optimizar procesos en la interacción entre los usuarios y las máquinas.