

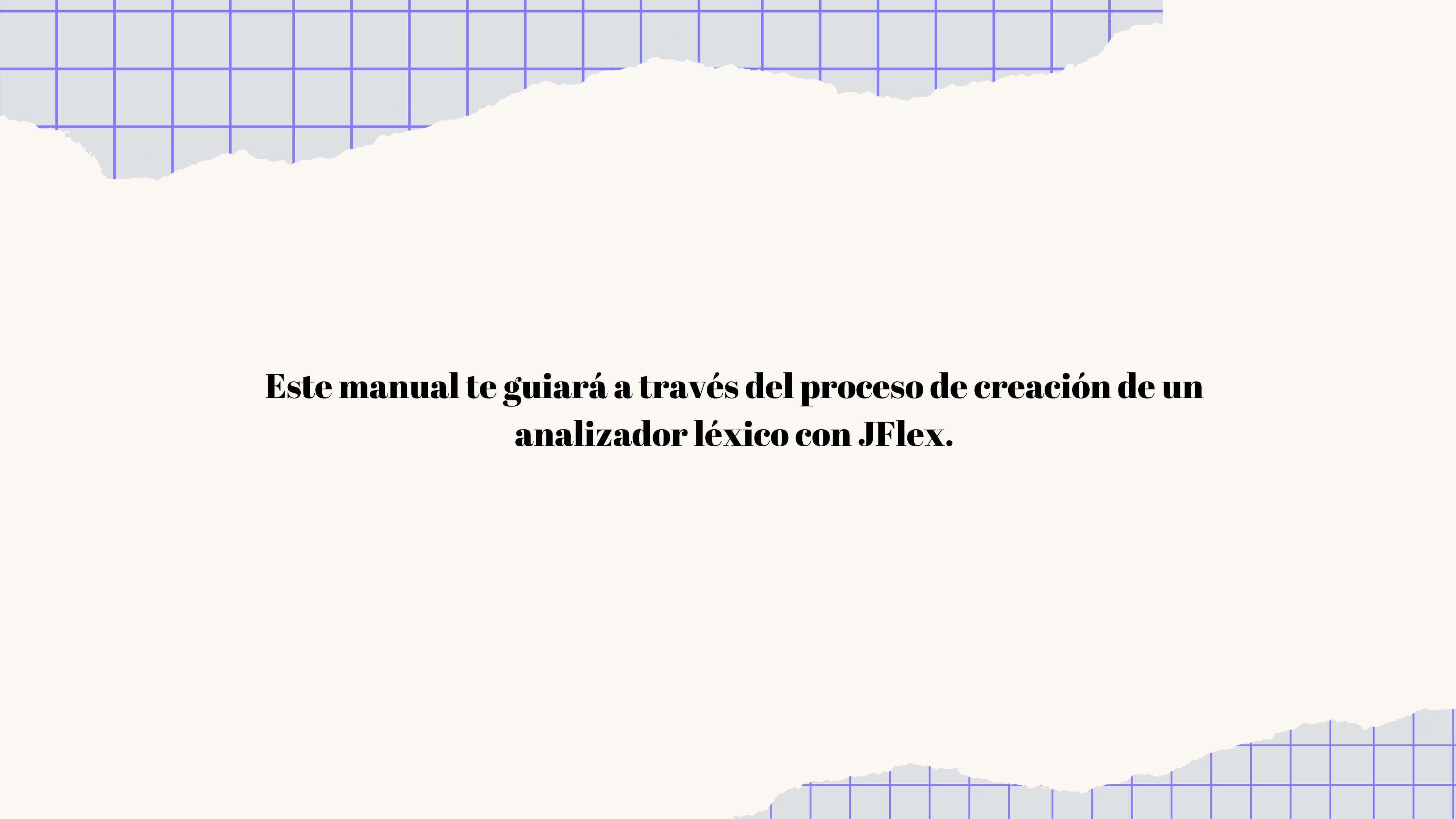
Universidad San Carlos De Guatemala
Centro Universitario De Occidente
Ingeniería en Ciencias y Sistemas
Lenguajes Formales
Auxiliar Julio Xicoy



Manual Tecnico

**Enmer Oswaldo Sandoval
Mazariegos
202031773**

15 de Octubre 2023



**Este manual te guiará a través del proceso de creación de un
analizador léxico con JFlex.**

Requisitos

Antes de comenzar, asegúrate de tener instalado JFlex y Java en tu sistema.

Pasos para Crear un Analizador Léxico con JFlex

Pasos para Crear un Analizador Léxico con JFlex

Paso 1: Crear el Archivo de Especificación de Patrones

El primer paso es crear un archivo de especificación de patrones. Este archivo contendrá las reglas que el analizador léxico utilizará para reconocer tokens en el código fuente. Un ejemplo sería:

jflex

%%

%class Lexico

%type Token

%line

%column

Pasos para Crear un Analizador Léxico con JFlex

```
%{  
    public String lexema;  
}%}
```

```
%%
```

```
// Definición de patrones y acciones para reconocer tokens  
palabra_reservada1 | palabra_reservada2 | palabra_reservada3 { lexema = yytext();  
return new Token(Tipo.PALABRA_RESERVADA, yyline + 1, yycolumn + 1, lexema); }
```

```
// Otras reglas para tokens, como operadores, números, identificadores, etc.
```

```
// Reglas para ignorar espacios en blanco y comentarios
```

```
. { lexema = yytext(); return new Token(Tipo.ERROR, yyline + 1, yycolumn + 1,  
lexema); }
```

Pasos para Crear un Analizador Léxico con JFlex

Paso 2-Generar el analizador léxico

Una vez que hayas creado el archivo de especificación de patrones, debes generar el analizador léxico utilizando JFlex. Para hacerlo, ejecuta el siguiente comando en la terminal:

shell

`jflex nombre_del_archivo.flex`

Pasos para Crear un Analizador Léxico con JFlex

Paso 3: Implementar la Clase Token

Debes implementar una clase llamada Token que represente los tokens reconocidos por el analizador léxico. Esta clase debe contener información relevante, como el tipo de token, el lexema y la posición en el código fuente.

Pasos para Crear un Analizador Léxico con JFlex

Paso 4: Integrar el Analizador Léxico en tu Proyecto

Integra el analizador léxico en tu proyecto Java. Puedes usar el analizador léxico generado junto con la clase Token en tu aplicación para escanear y reconocer tokens en el código fuente.

Pasos para Crear un Analizador Léxico con JFlex

Paso 5: Utilizar el Analizador Léxico

Finalmente, puedes utilizar el analizador léxico en tu proyecto para procesar código fuente.

CADENAS DE PRODUCCIÓN

sentencia → sentencia_if
| sentencia_for
| sentencia_while
| declaracion_var
| asignacion
| sentencia_funcion

sentencia_if → IF expresion DOSPUNTOS bloque [sentencia_elif] [sentencia_else]

sentencia_elif → ELIF expresion DOSPUNTOS bloque [sentencia_elif]

sentencia_else → ELSE DOSPUNTOS bloque

sentencia_for → FOR IDENTIFICADOR IN expresion DOSPUNTOS bloque [ELSE DOSPUNTOS bloque]

CADENAS DE PRODUCCIÓN

declaracion_var -> IDENTIFICADOR ASIGNACION expresion

asignacion -> IDENTIFICADOR operador_asignacion expresion

asignacion_vars -> lista_identificadores operador_asignacion lista_expresiones

lista_identificadores -> IDENTIFICADOR ["," lista_identificadores]

lista_expresiones -> expresion ["," lista_expresiones]

expresion -> BOOLEANO

| comparacion

| lista

| diccionario

| llamada_funcion

| operador_terciario

CAENAS DE PRODUCCIÓN

comparacion -> expresion ARITMETICO expresion

lista -> IDENTIFICADOR operador_asignacion LISTA valores_lista

valores_lista -> "[" lista_elementos "]"

lista_elementos -> expresion ["," lista_elementos]

diccionario -> IDENTIFICADOR operador_asignacion DICCIONARIO
valores_diccionario

valores_diccionario -> "{" lista_pares "}"

lista_pares -> clave ":" valor ["," lista_pares]

clave -> STRING

CAENAS DE PRODUCCIÓN

valor -> expresion
| STRING

llamada_funcion -> IDENTIFICADOR "(" lista_argumentos ")"

operador_tenario -> IDENTIFICADOR operador_ternario expresion DOSPUNTOS
expresion

sentencia_funcion -> DEF IDENTIFICADOR "(" lista_parametros ")" DOSPUNTOS
bloque [RETURN expresion]

lista_parametros -> IDENTIFICADOR ["," lista_parametros]

lista_argumentos -> expresion ["," lista_argumentos]

sentencia_if:
IF expresion
DOSPUNTOS bloque
[sentencia_elif]
[sentencia_else]

sentencia_elif:
ELIF expresion
DOSPUNTOS bloque
[sentencia_elif]

sentencia_else:
ELSE DOSPUNTOS
bloque

sentencia_if

sentencia_for

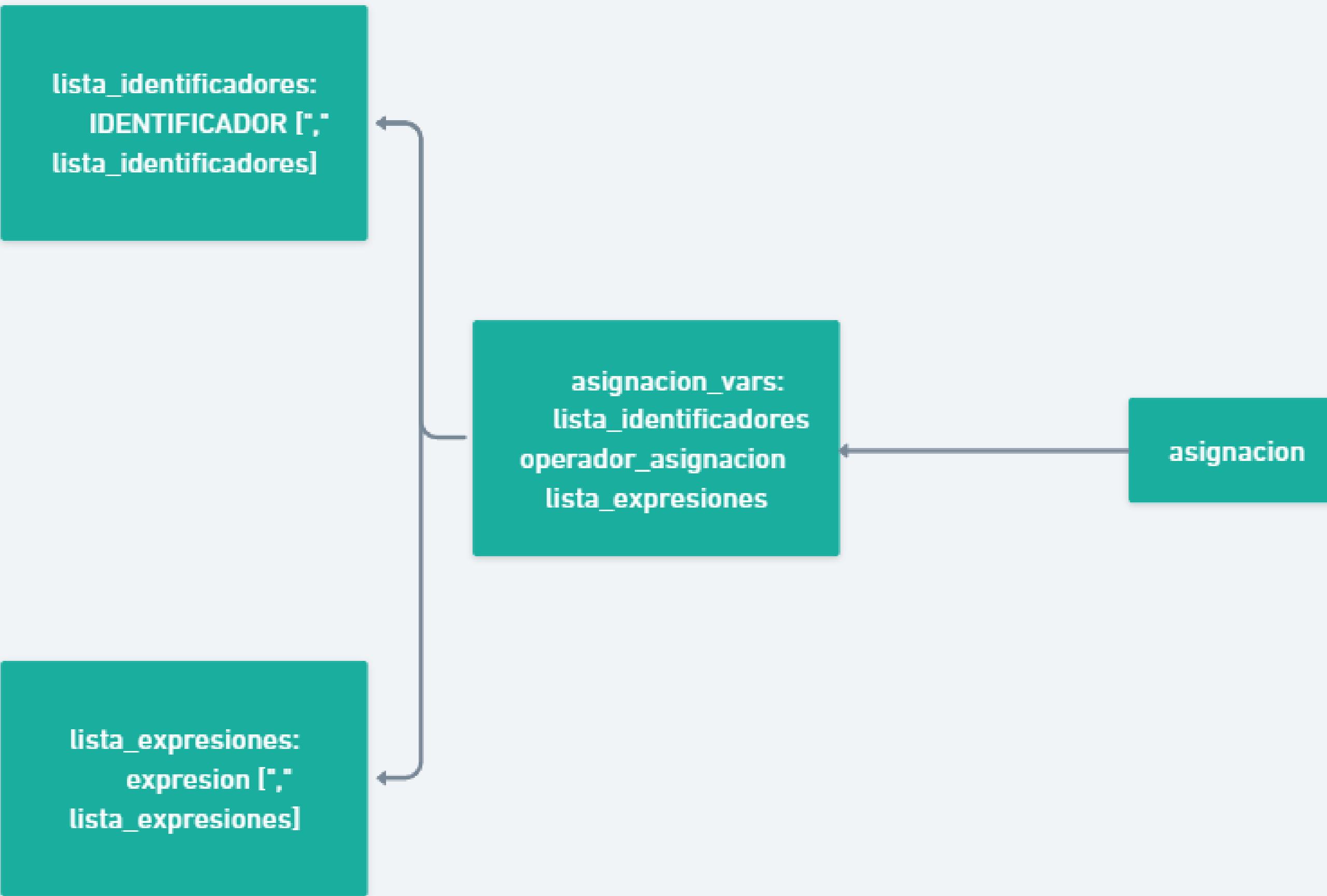
sentencia_for:
FOR IDENTIFICADOR IN
expresion DOSPUNTOS
bloque [ELSE DOSPUNTOS
bloque]

sentencia_while

sentencia_while:
WHILE expresion
DOSPUNTOS bloque [ELSE
DOSPUNTOS bloque]

sentencia_while:
WHILE expresion
DOSPUNTOS bloque [ELSE
DOSPUNTOS bloque]

declaracion_var



expresion:
BOOLEANO
comparacion
lista
diccionario
llamada_funcion
operador_terciario

comparacion:
expresion
ARITMETICO expresion

lista:
IDENTIFICADOR
operador_asignacion
LISTA valores_lista

Sentencia

valores_lista:
[" lista_elementos "]

lista_elementos:
expresion [;]
lista_elementos]

diccionario:
IDENTIFICADOR
operador_asignacion
DICCCIONARIO
valores_diccionario

Expresion

valores_diccionario:
{" lista_pares "}

lista_pares:
clave ":" valor [;
lista_pares]

clave:
STRING

valor:
expression
STRING

llamada_funcion:
IDENTIFICADOR "("
lista_argumentos ")"

operador_ternario:
IDENTIFICADOR
operador_ternario
expression DOSPUNTOS
expression

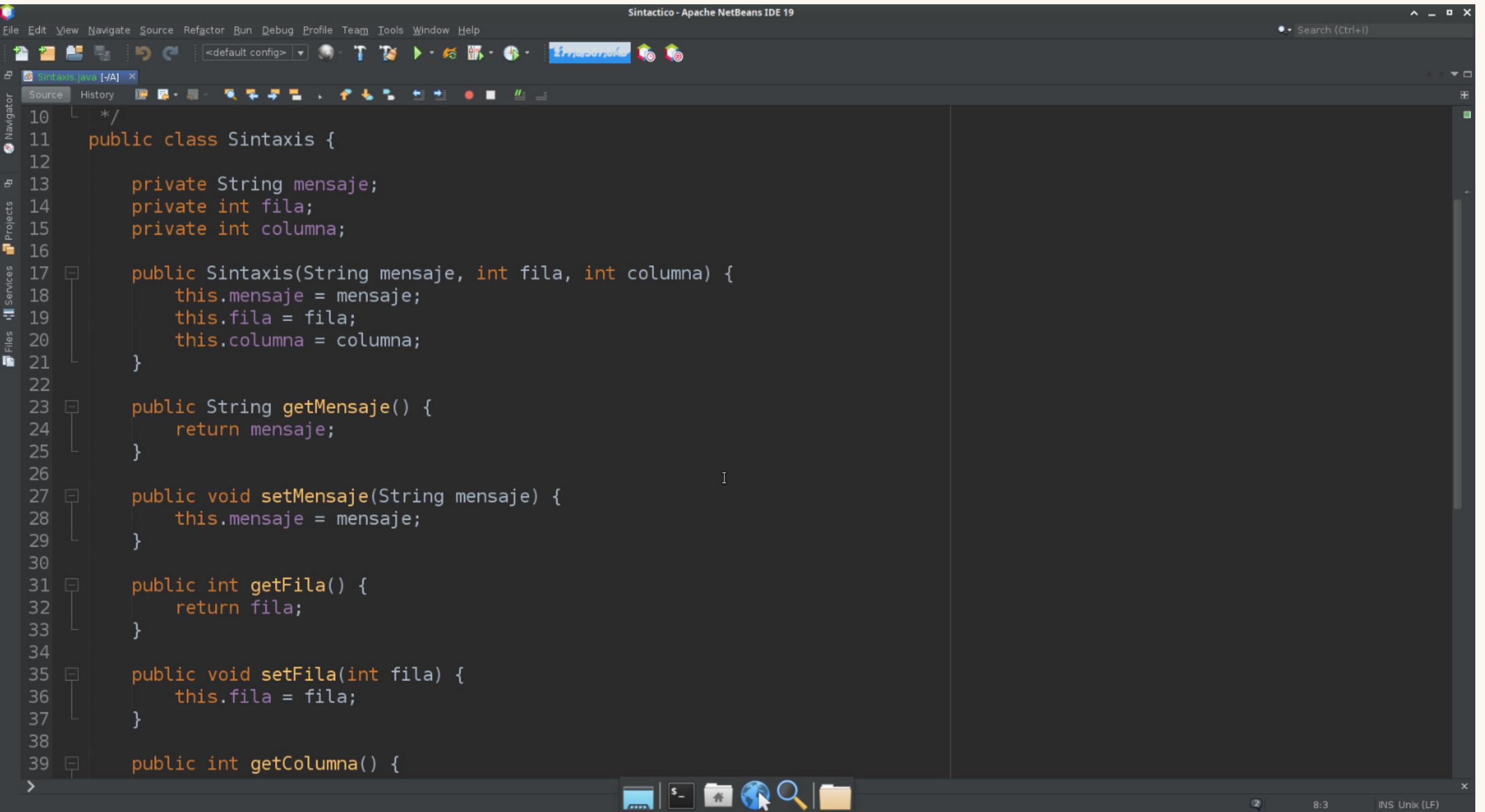
sentencia_funcion:
DEF IDENTIFICADOR
[" lista_parametros "]
DOSPUNTOS bloque
[RETURN expresion]

lista_parametros:
IDENTIFICADOR [";"
lista_parametros]

lista_argumentos:
expresion [";"
lista_argumentos]

sentencia_funcion

Clases sintaxis

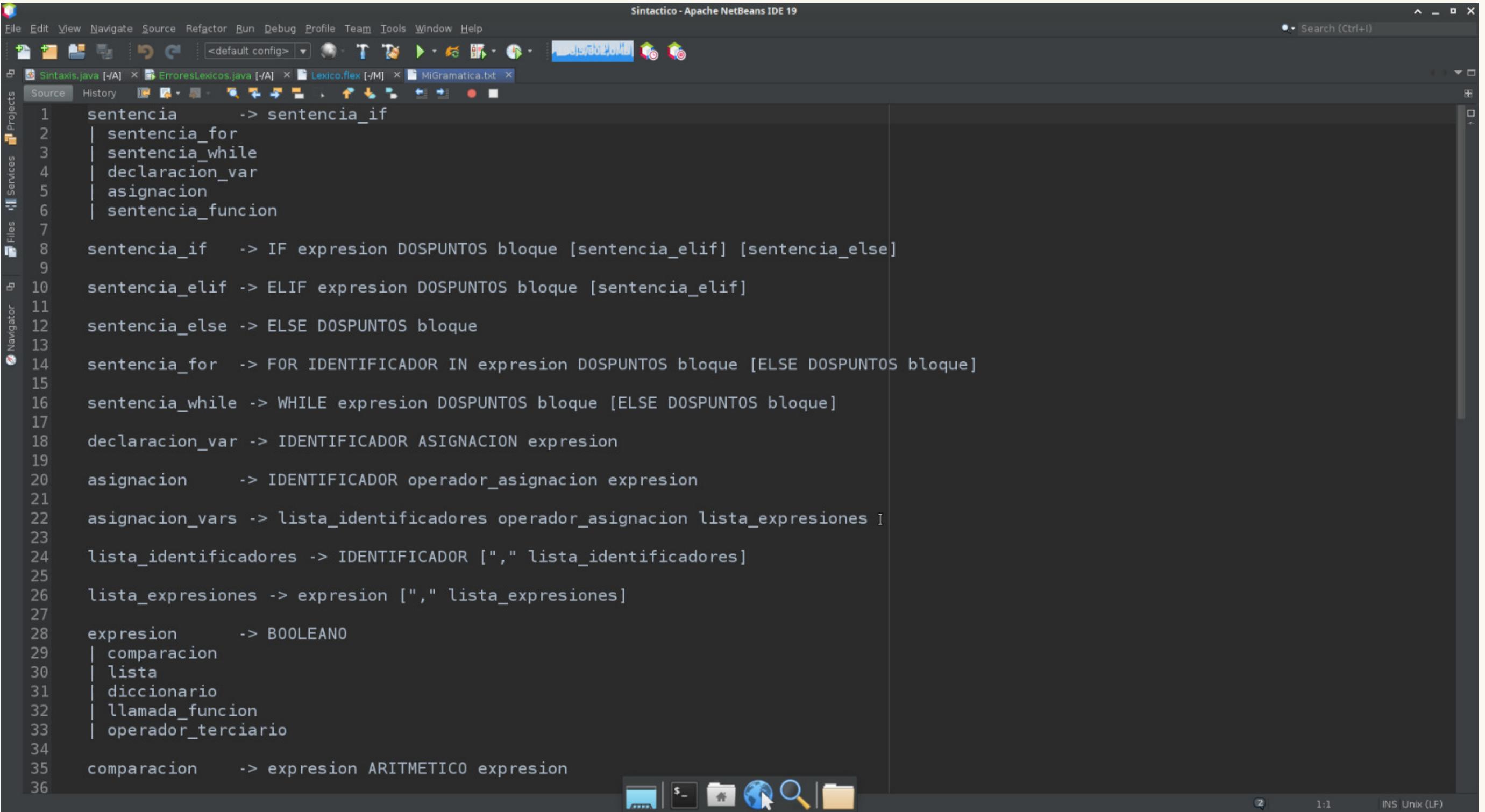


The screenshot shows the Apache NetBeans IDE 19 interface with a dark theme. The main window displays a Java file named `Sintaxis.java`. The code defines a class `Sintaxis` with private attributes `mensaje`, `fila`, and `columna`. It includes a constructor that initializes these variables, and methods to get and set each attribute. The code is color-coded, with keywords in blue, strings in red, and comments in green. The NetBeans toolbar at the top contains various icons for file operations like Open, Save, and Run. The bottom status bar shows the file path as `Sintaxis.java`, the line number as `8:3`, and the encoding as `INS Unix (LF)`.

```
10  /*
11   * Sintaxis.java
12   */
13  public class Sintaxis {
14
15      private String mensaje;
16      private int fila;
17      private int columna;
18
19      public Sintaxis(String mensaje, int fila, int columna) {
20          this.mensaje = mensaje;
21          this.fila = fila;
22          this.columna = columna;
23      }
24
25      public String getMensaje() {
26          return mensaje;
27      }
28
29      public void setMensaje(String mensaje) {
30          this.mensaje = mensaje;
31      }
32
33      public int getFila() {
34          return fila;
35      }
36
37      public void setFila(int fila) {
38          this.fila = fila;
39      }
39
40      public int getColumna() {
41
42      }
43
44  }
```

Archivo JFlex

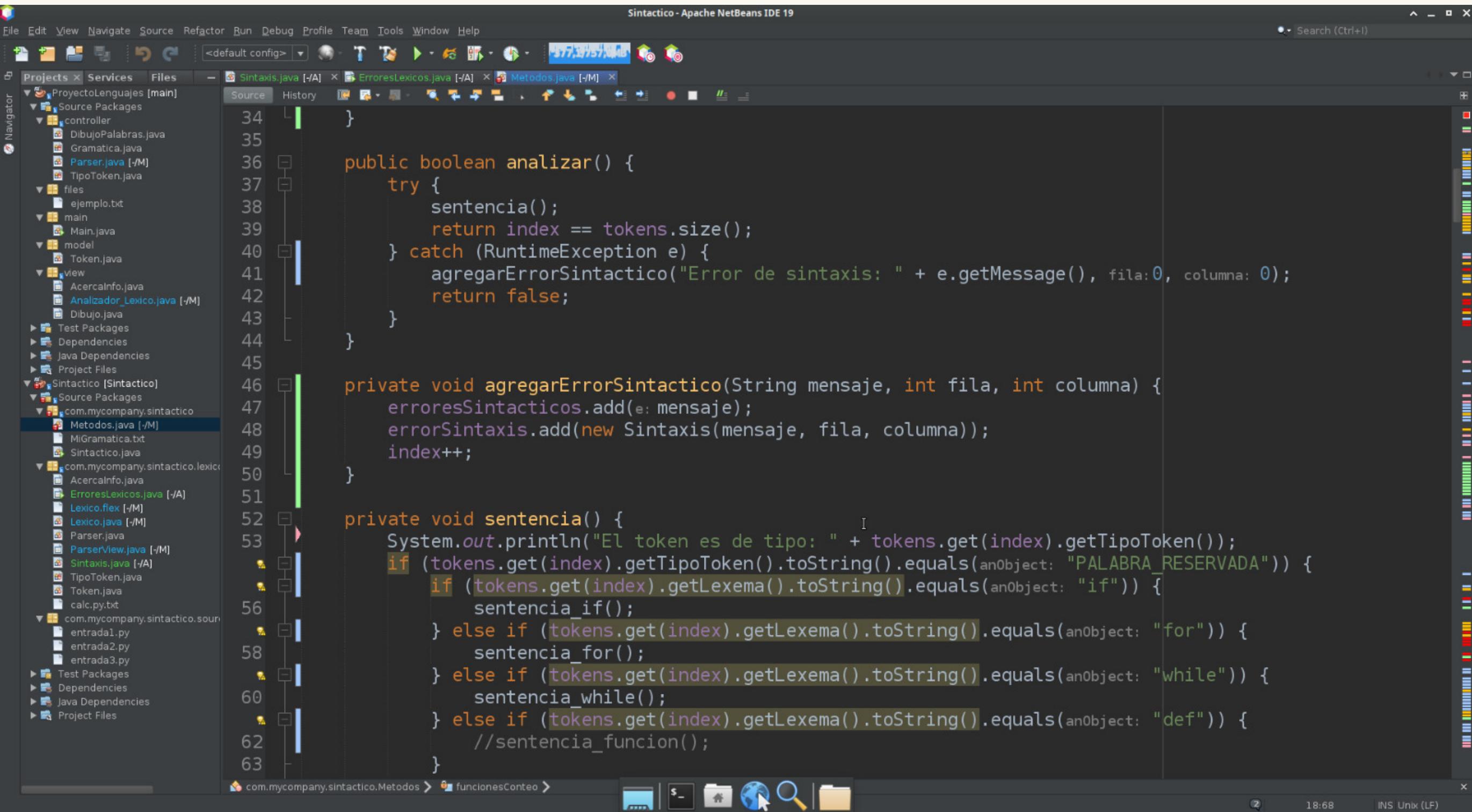
La gramática utilizada



The screenshot shows the Apache NetBeans IDE 19 interface with the title bar "Sintactico - Apache NetBeans IDE 19". The main window displays a text editor with the file "MiGramatica.txt" open. The content of the file is a context-free grammar (CFG) defined using EBNF notation. The grammar rules are as follows:

```
1 sentencia      -> sentencia_if
2 | sentencia_for
3 | sentencia_while
4 | declaracion_var
5 | asignacion
6 | sentencia_funcion
7
8 sentencia_if    -> IF expresion DOSPUNTOS bloque [sentencia_elif] [sentencia_else]
9
10 sentencia_elif -> ELIF expresion DOSPUNTOS bloque [sentencia_elif]
11
12 sentencia_else  -> ELSE DOSPUNTOS bloque
13
14 sentencia_for   -> FOR IDENTIFICADOR IN expresion DOSPUNTOS bloque [ELSE DOSPUNTOS bloque]
15
16 sentencia_while -> WHILE expresion DOSPUNTOS bloque [ELSE DOSPUNTOS bloque]
17
18 declaracion_var -> IDENTIFICADOR ASIGNACION expresion
19
20 asignacion       -> IDENTIFICADOR operador_asignacion expresion
21
22 asignacion_vars -> lista_identificadores operador_asignacion lista_expresiones I
23
24 lista_identificadores -> IDENTIFICADOR [," lista_identificadores]
25
26 lista_expresiones -> expresion [," lista_expresiones]
27
28 expresion        -> BOOLEANO
29 | comparacion
30 | lista
31 | diccionario
32 | llamada_funcion
33 | operador_terciario
34
35 comparacion     -> expresion ARITMETICO expresion
36
```

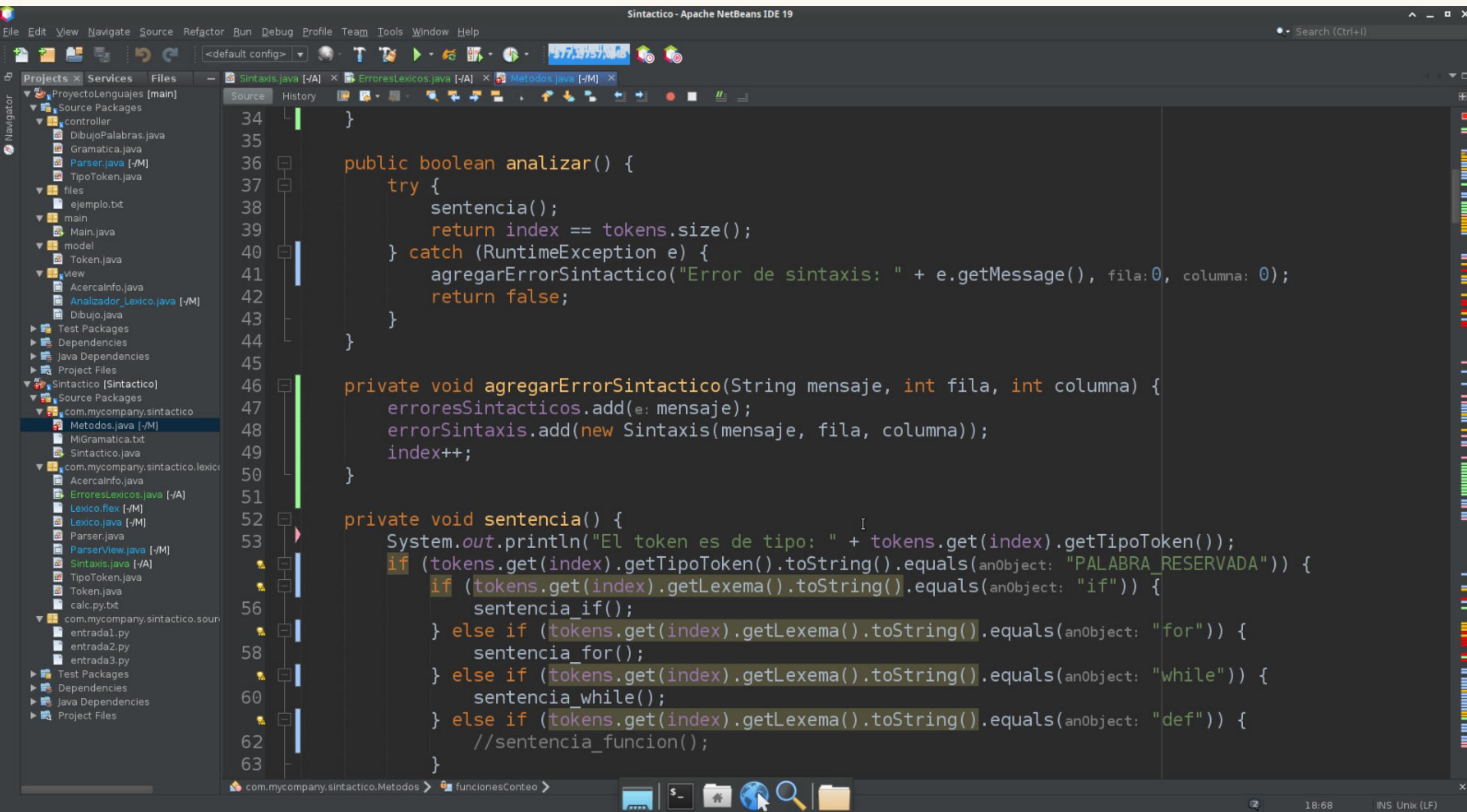
Clase métodos la utilizamos para poder verificar nuestros tokens



The screenshot shows the Apache NetBeans IDE 19 interface with the title bar "Sintactico - Apache NetBeans IDE 19". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for file operations like New, Open, Save, and Run. The Projects tab is selected, showing a project named "ProyectoLenguajes [main]" with packages controller, files, main, model, and view. Under view, there is an "Analizador_Lexico.java" file. The Files tab shows several Java files: Sintaxis.java, ErroresLexicos.java, and Metodos.java. The Metodos.java file is open in the editor, displaying Java code for a syntax checker. The code includes methods for analyzing tokens and adding errors. The code is as follows:

```
34     }
35
36     public boolean analizar() {
37         try {
38             sentencia();
39             return index == tokens.size();
40         } catch (RuntimeException e) {
41             agregarErrorSintactico("Error de sintaxis: " + e.getMessage(), fila:0, columna: 0);
42             return false;
43         }
44     }
45
46     private void agregarErrorSintactico(String mensaje, int fila, int columna) {
47         erroresSintacticos.add(e: mensaje);
48         errorSintaxis.add(new Sintaxis(mensaje, fila, columna));
49         index++;
50     }
51
52     private void sentencia() {
53         System.out.println("El token es de tipo: " + tokens.get(index).getTipoToken());
54         if (tokens.get(index).getTipoToken().toString().equals(anObject: "PALABRA_RESERVADA")) {
55             if (tokens.get(index).getLexema().toString().equals(anObject: "if")) {
56                 sentencia_if();
57             } else if (tokens.get(index).getLexema().toString().equals(anObject: "for")) {
58                 sentencia_for();
59             } else if (tokens.get(index).getLexema().toString().equals(anObject: "while")) {
60                 sentencia_while();
61             } else if (tokens.get(index).getLexema().toString().equals(anObject: "def")) {
62                 //sentencia_funcion();
63             }
64         }
65     }
66 }
```

Clase métodos la utilizamos para poder verificar nuestros tokens



The screenshot shows the Apache NetBeans IDE 19 interface with the following details:

- Title Bar:** Sintactico - Apache NetBeans IDE 19
- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help
- Toolbar:** Includes icons for New Project, Open Project, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, Run, Stop, and others.
- Project Explorer:** Shows the project structure under "Projects".
 - ProyectoLenguajes [main] (selected)
 - Sintactico [Sintactico]
 - com.mycompany.sintactico
 - com.mycompany.sintactico.lexico
 - com.mycompany.sintactico.sourc
- Code Editor:** Displays the content of the Metodos.java file.

```
34     }
35
36     public boolean analizar() {
37         try {
38             sentencia();
39             return index == tokens.size();
40         } catch (RuntimeException e) {
41             agregarErrorSintactico("Error de sintaxis: " + e.getMessage(), fila:0, columna: 0);
42             return false;
43         }
44     }
45
46     private void agregarErrorSintactico(String mensaje, int fila, int columna) {
47         erroresSintacticos.add(e: mensaje);
48         errorSintaxis.add(new Sintaxis(mensaje, fila, columna));
49         index++;
50     }
51
52     private void sentencia() {
53         System.out.println("El token es de tipo: " + tokens.get(index).getTipoToken());
54         if (tokens.get(index).getTipoToken().toString().equals(anObject: "PALABRA_RESERVADA")) {
55             if (tokens.get(index).getLexema().toString().equals(anObject: "if")) {
56                 sentencia_if();
57             } else if (tokens.get(index).getLexema().toString().equals(anObject: "for")) {
58                 sentencia_for();
59             } else if (tokens.get(index).getLexema().toString().equals(anObject: "while")) {
60                 sentencia_while();
61             } else if (tokens.get(index).getLexema().toString().equals(anObject: "def")) {
62                 //sentencia_funcion();
63             }
64         }
65     }
66 }
```
- Bottom Status Bar:** Shows the current file (com.mycompany.sintactico.Metodos), the function (funcionesConteo), the time (18:68), and the operating system (INS Unix (LF)).