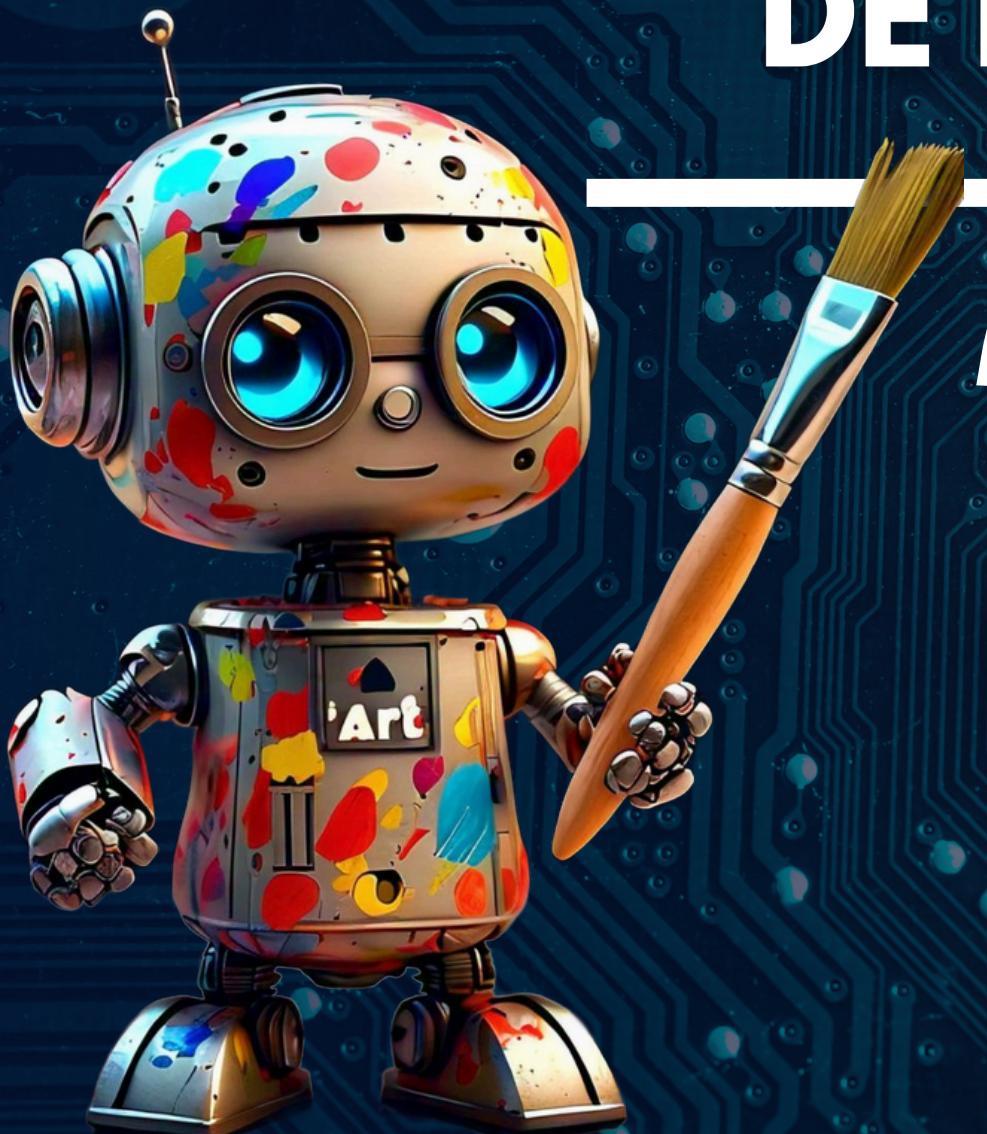




CONSTRUCCION DE UN COMPILADOR DE ESTRUCTURAS EN ESPAÑOL



Análisis Léxico y Análisis Sintáctico

Equipo:

Vanesa Hernández Martínez

Raul Ciriaco Castillo

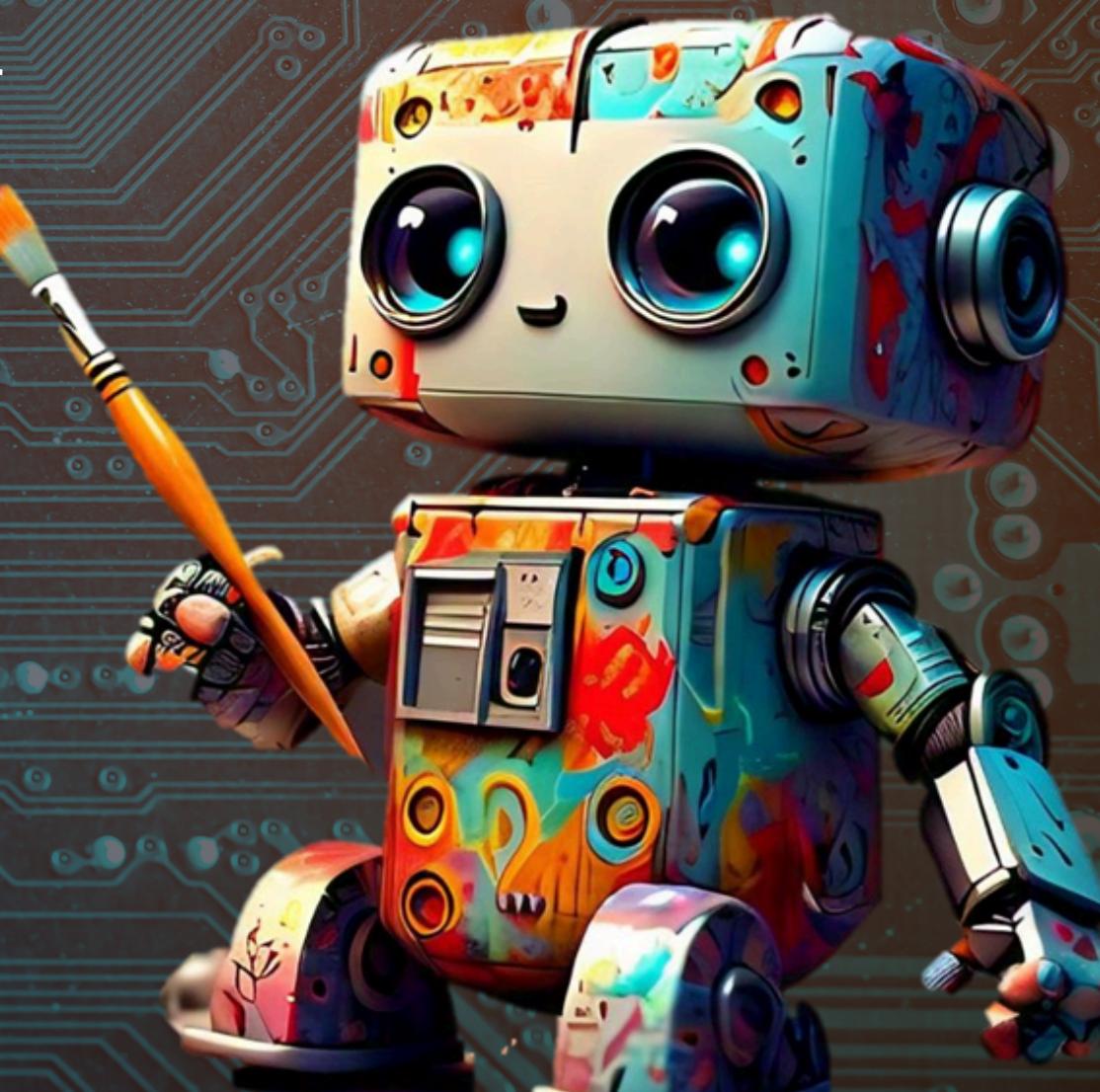
Brandon Mendoza Castillo

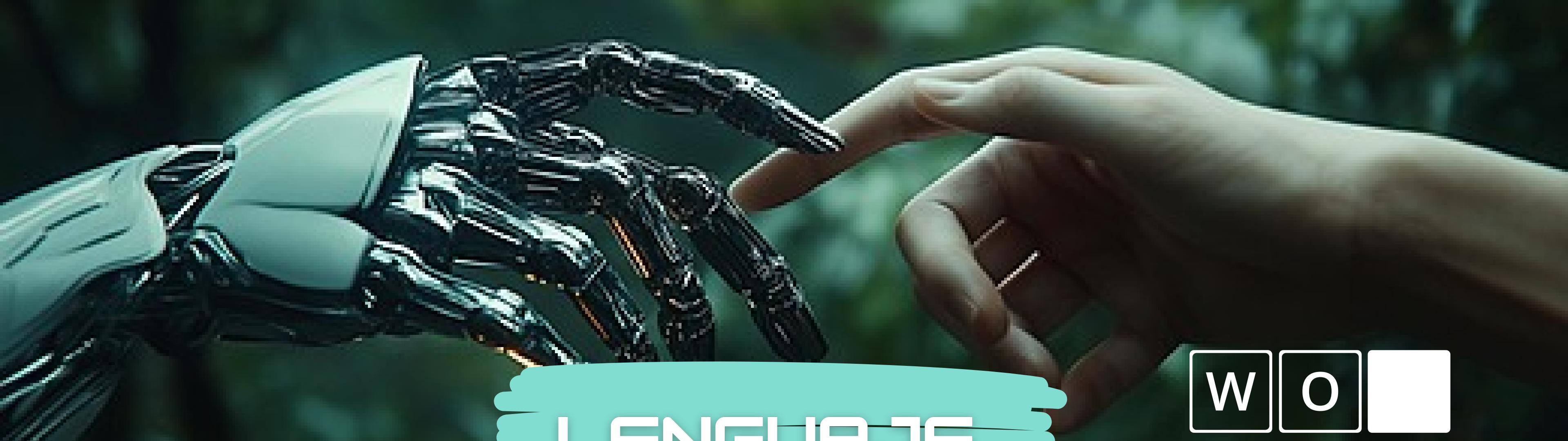
Emilio Rivera Facio



OBJETIVO

El objetivo de este proyecto fue crear un compilador de estructuras que use análisis léxico y análisis sintáctico para procesar y validar un lenguaje estructurado en español.





LENGUAJE

W	O	
	R	D



Conjunto bien definido de *cadenas* formadas por
simbolos pertenecientes a un *alfabeto*

alfabeto → cadenas → lenguaje

ELEMENTOS DEL LENGUAJE



- Alfabeto
- Palabras o tokens
- Sintaxis
- Semántica
- Gramática





ANÁLIZADOR LÉXICO

Es el componente encargado de:

Generar tokens, que son representaciones abstractas de los elementos del código.



Leer el texto fuente carácter por carácter.



Identificar patrones definidos por reglas.

identificador numeros
clases



OBJETIVOS

ANÁLISIS LÉXICO

1.

Simplificar
la tarea del
analizador
sintáctico



2.

Identifica
errores
léxicos





¿CÓMO FUNCIONA?

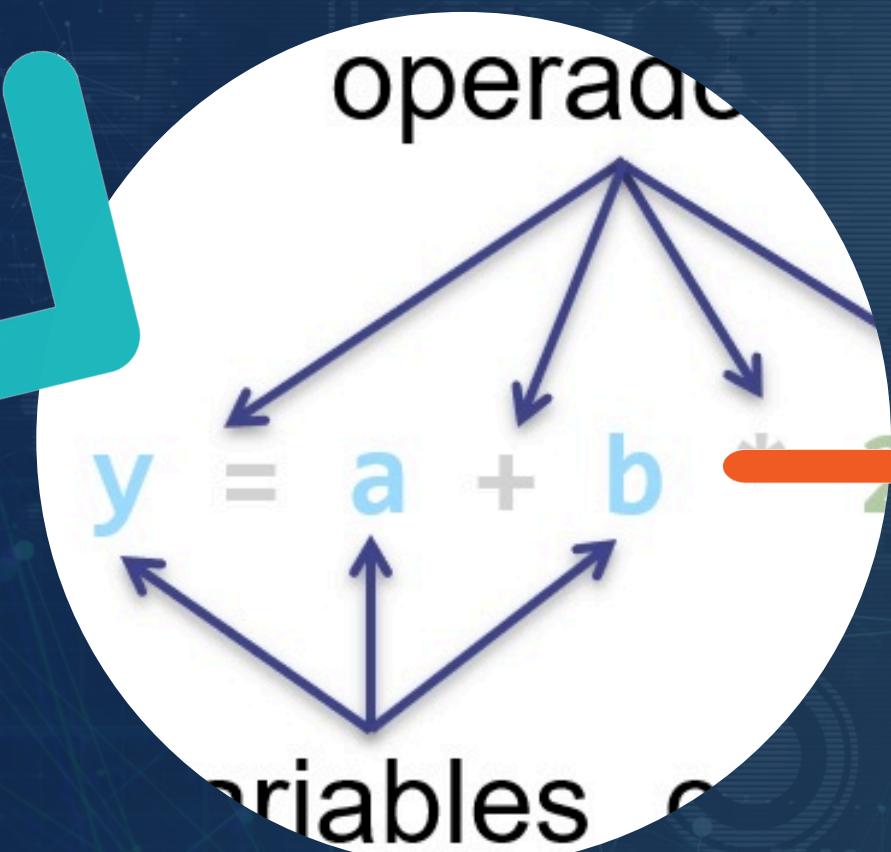
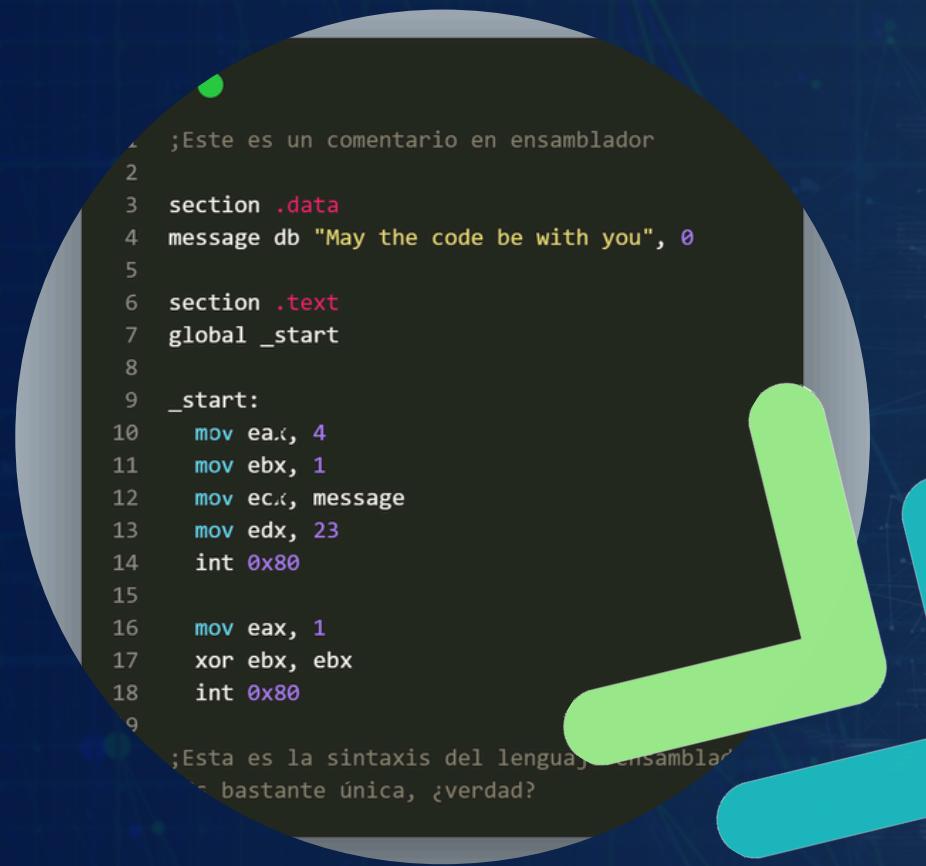


Tabla de símbolos
y patrones



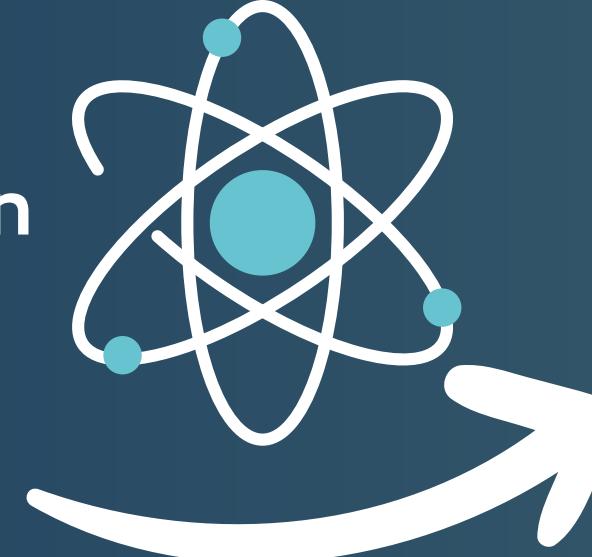
Producción de
tokens



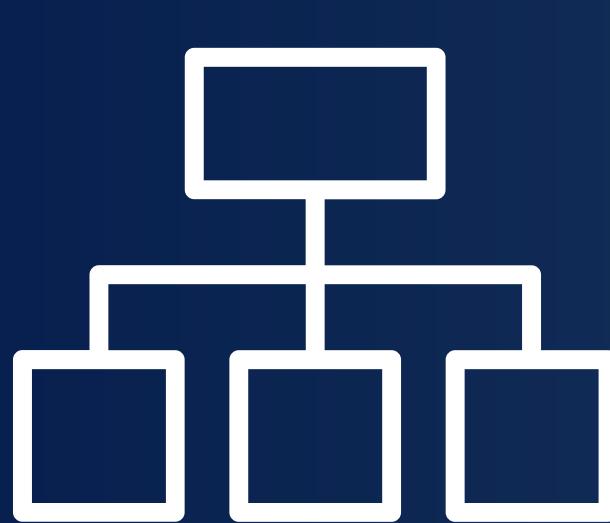
Salida:
Tokens



Unidad mínima en un lenguaje de programación

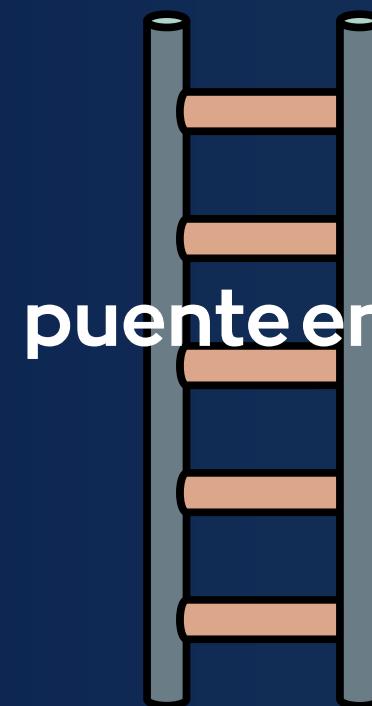


Indivisible



La base para los árboles sintácticos

código fuente

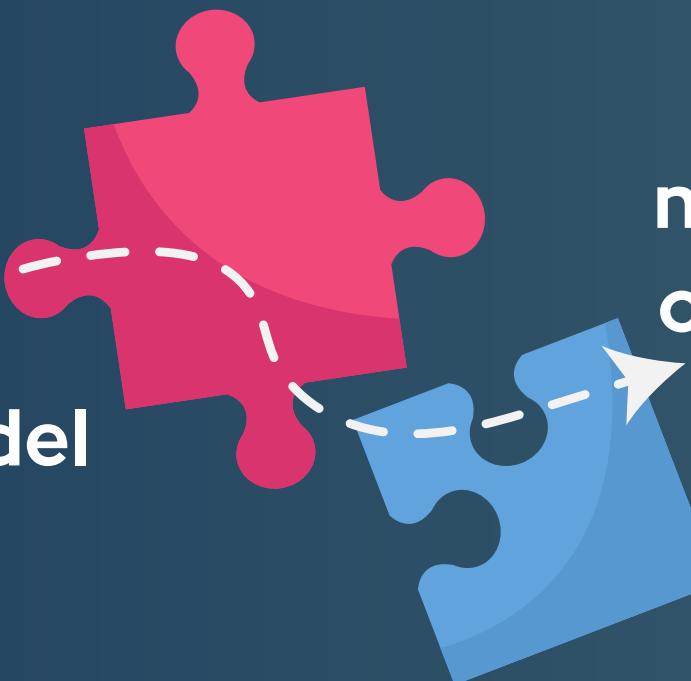


puente entre

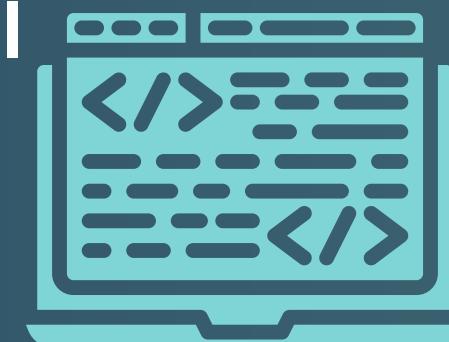
procesamiento interno

Tokens

Facilitan el procesamiento del lenguaje.



Piezas manejables y clasificables.



Son generados por el analizador léxico al procesar el código fuente.

Operadores aritméticos

Operadores relacionales

Operadores lógicos

Operadores

Signos

Constantes Matemáticas

Número de token	Token	Lexema
1	suma	"+"
2	resta	"-"
3	multiplicacion	"**"
4	division	"/"
5	mayor	
6	menor	<
7	mayorIgual	>=
8	menorIgual	<=
9	comparacion	"=="
10	distinto	"!="
11	y	"&"
12	o	" "
13	no	"!"
14	incremento	"++"
15	decremento	"--"
16	modulo	"%"
17	potencia	"**"
18	raiz	"raiz"
19	punto	"."
20	asignacion	"=>"
21	igual	"="
22	concatenacion	"con"
23	positivo	"pos"
24	negativo	"neg"
25	pi	"PI"
26	euler	"E"
27	infinito	"inf"

Delimitadores

Delimitadores de cadenas

Comentarios

Tipos de Datos

Estructuras de control

27	parentesisApertura	"("
28	parentesisCierre	")"
29	corcheteApertura	
30	corcheteCierre	
31	llaveApertura	
32	llaveCierre	}
33	inicioTexto	
34	finalTexto	>>
35	comillaDoble	"
36	comillaSimple	'
37	comentario	"#"
38	inicioComentario	"/*"
39	finComentario	"/**"
40	entero	"ent"
41	decimal	"dec"
42	verdadero	"v"
43	falso	"f"
44	cadena	"cadena"
45	si	"si"
46	siNo	"siNo"
47	siNoSi	"siNoSi"
48	cicloRepite	"repite"
49	mientras	"mientras"
50	hacer	"hacer"
51	seleccionador	"seleccionar"
52	caso	"caso"
53	predeterminado	"predeterminado"
54	detener	"detener"

Declaraciones

Entrada y salida de Datos

Componentes de una cadena

55	metodo	"metodo"
56	arreglo	"arreglo"
57	funcion	"fun"
58	clase	"clase"
59	procedimiento	"proc"
60	principal	"principal"
61	biblioteca	"biblioteca"
62	imprimir	imprime
63	leer	leer
64	importar	"importa"
65	retornar	"retorna"
66	nulo	"nulo"
67	salto de linea	"\n"
68	caracteresEspeciales	



ANÁLIZADOR SINTÁCTICO



VERIFICAR QUE LA SECUENCIA DE TOKENS CUMPLA LAS REGLAS GRAMATICALES DEL LENGUAJE.



ORGANIZAR LOS TOKENS EN UNA ESTRUCTURA LÓGICA QUE CAPTURE LAS RELACIONES ENTRE ELLOS.



OBJETIVOS

ANÁLISIS SINTÁCTICO

1.

Validar la
estructura
del
programa



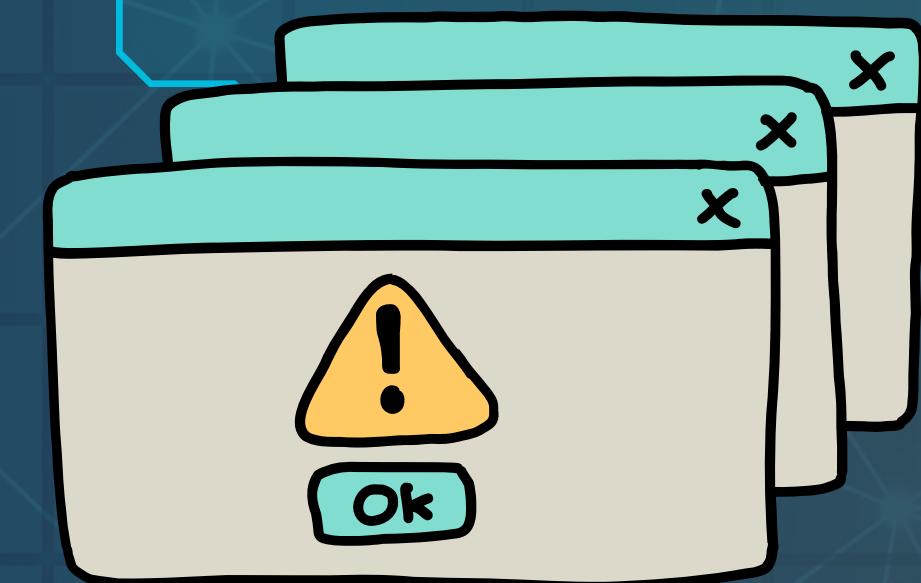
2.

Validar la
estructura
del
programa



3.

Detectar
errores
sintácticos





¿CÓMO FUNCIONA?

Entrada:
Tokens

asignacion
suma
resta
verdadero

Gramática

SENTENCIA_BOOLEANA ::=
Identificador Mayor Numero ;

Salida:árbol
sintáctico

Análisis





ASPECTO	ANALIZADOR LÉXICO	ANALIZADOR SINTÁCTICO
ENTRADA	Código fuente (caracteres)	Secuencia de tokens
SALIDA	Tokens	Árbol sintáctico o error
PROpósito	Identificar elementos básicos	Verificar estructura gramática
ERRORES DETECTADOS	Errores léxicos (caracteres inválidos)	Errores de gramática





HERRAMIENTAS



Lexer.flex

¿Qué es?

Archivo de especificación donde defines las reglas de cómo dividir un texto en partes más pequeñas (tokens).



¿Como funciona?

Defines patrones para identificar diferentes tipos de texto

Asocias una acción a cada patrón.



¿Cúal es su función?

Especificar las reglas que el generador de código (JFlex) utilizará para identificar las partes del texto.



Este archivo está diseñado para ser usado con un lexer independiente

Lexer.java

¿Qué es?

Archivo generado automáticamente por JFlex para reconocer palabras o símbolos en un texto

¿Como funciona?

Usa reglas definidas en otro archivo (Lexer.flex).

Convierte esos reglas en código que sabe cómo leer y reconocer diferentes partes del texto.

¿Cúal es su función?

Leer un texto y dividirlo en tokens

lexerCup.flex

¿Qué es?

Archivo que define las reglas para reconocer diferentes tokens en un texto.

¿Como funciona?

Defines patrones para identificar diferentes tipos de texto

Asocias una acción a cada patrón.

¿Cúal es su función?

Como dividir el texto en partes significativas (tokens)

Está especialmente diseñado para trabajar en conjunto con el parser generado por CUP

lexerCup.java

¿Qué es?

Archivo generado automáticamente, es un analizador léxico o scanner

¿Como funciona?

Lee caracteres del texto de entrada uno por uno.

Cada carácter leído se clasifica en una de las clases de caracteres definidas

¿Cúal es su función?

Lee el texto de entrada, lo escanea y lo divide en tokens.

LEXER

Ejemplo

```
ent principal(){  
    nombre = "Eiden";  
}
```

Analizador
léxico

<Identificador> . nombre
<Operador igualdad> =
<Texto> "Eiden"
<Punto y coma> ;

Tokens.java

¿Qué es?

El archivo que define una enumeración (enum) en Java llamada Tokens.

¿Como funciona?

El enum define una serie de tokens

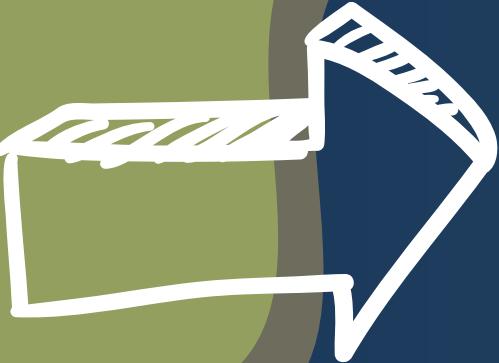
El lexer es el componente encargado de convertir el código fuente en una secuencia de estos tokens. Cuando el lexer encuentra un símbolo en el código, lo convierte en uno de estos valores.

¿Cúal es su función?

Proporcionar una lista centralizada y organizada de todos los tipos de tokens que el lexer puede reconocer.

TOKENS

```
public enum Tokens {  
    Suma,  
    Resta,  
    Multiplicacion,  
    Division  
}
```



Token	Lexema
suma	"+"
resta	"_"
multiplicacion	"*"
division	"/"

Sintaxis.cup

¿Qué es?

Representa una gramática para un analizador sintáctico escrito en Java CUP

¿Como funciona?

Los terminales representan los tokens léxicos

Los no terminales representan construcciones sintácticas complejas

Las reglas definen cómo los no terminales se descomponen en terminales y otros no terminales.

¿Cúal es su función?

Reglas léxicas y sintácticas

Manejo de errores

Manejo de errores

CUP (Constructor of Useful Parsers)

Sintaxis.java

¿Qué es?

Es un parser
generado
automáticamente
por la herramienta
CUP

¿Como funciona?

Lectura de tokens

Reducción de
producciones

Acciones asociadas

Manejo de errores

Devolución del
resultado

¿Cúal es su función?

Validar la sintaxis del
código de entrada

Crear un árbol de
derivación

Informar errores de
sintaxis

Utiliza un autómata LR (Left-to-right parsing, Rightmost derivation) generado por CUP.

Principal.java

¿Cuál es su función?

Se encarga de ejecutar las herramientas JFlex y CUP con los archivos de configuración correspondientes, y organiza los archivos generados dentro de la estructura del proyecto.

LEXER CUP

2

LINEA 1
<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>
<Identificador>
<Operador decremento>
<Paréntesis cierre>
<Llave apertura>

LINEA 2
<Palabra reservada 'Imprimir'>

<Paréntesis apertura>
<Identificador> i
<Paréntesis cierre>
<Punto y coma>

LINEA 3
<Llave cierre>

SIMBOLO
repite
(

ent

i

<=

0

;

i

<

10

;

i

--

)

{

imprime

(

)

;

;

}

1

repite (ent i = 0; i >= 10; i--) {
 imprime(i);
}

3

FOR ::=
CicloRepite ParentesisApertura SENTENCIA_FOR ParentesisCierre
LlaveApertura SENTENCIA LlaveCierre;

SENTENCIA_FOR ::=
Entero Identificador Igual Numero P_coma
SENTENCIA_BOLEANA P_coma DECLARACION_FOR ;

SENTENCIA_BOLEANA ::=
Identificador MayorIgual Numero ;

SENTENCIA_BOLEANA ::=
Identificador MayorIgual Numero ;

DECLARACION_FOR ::=
Identificador Incremento ;

SENTENCIA ::=
Imprimir ParentesisApertura Identificador ParentesisCierre
P_coma;

4

Análisis realizado correctamente

LEXER CUP

1

```
si (numero %= 2 == 0 ) {  
    impri ("El número es par");  
}
```

2

LINEA 1
<Palabra reservada 'Si'>
<Paréntesis apertura>
<Identificador>
<Operador módulo>
<Número>
<Operador comparación>
<Número>
<Paréntesis cierre>
<Llave apertura>

LINEA 2
<Identificador>
<Paréntesis apertura>
<Texto>
<Paréntesis cierre>
<Punto y coma>

LINEA 3
<Llave cierre>

SIMBOLO
si
(
numero
%
2
==
0
)
{

imprl
(
"El número es par"
)
;
}

3

```
IF ::=  
    Si ParentesisApertura SENTENCIA_BOOLEANA ParentesisCierre  
        LlaveApertura SENTENCIA LlaveCierre;  
  
    SENTENCIA_BOOLEANA ::=  
        Identificador Modulo Numero Comparacion Numero;  
  
    SENTENCIA ::=  
        Imprimir ParentesisApertura Texto ParentesisCierre P_coma ;
```

4

Error de sintaxis. Línea: 4, Columna: 8, Texto: "("

`sym.java`

¿Qué es?

Clase generada automáticamente por CUP

Contiene una serie de constantes que representan los símbolos terminales del lenguaje que está siendo analizado

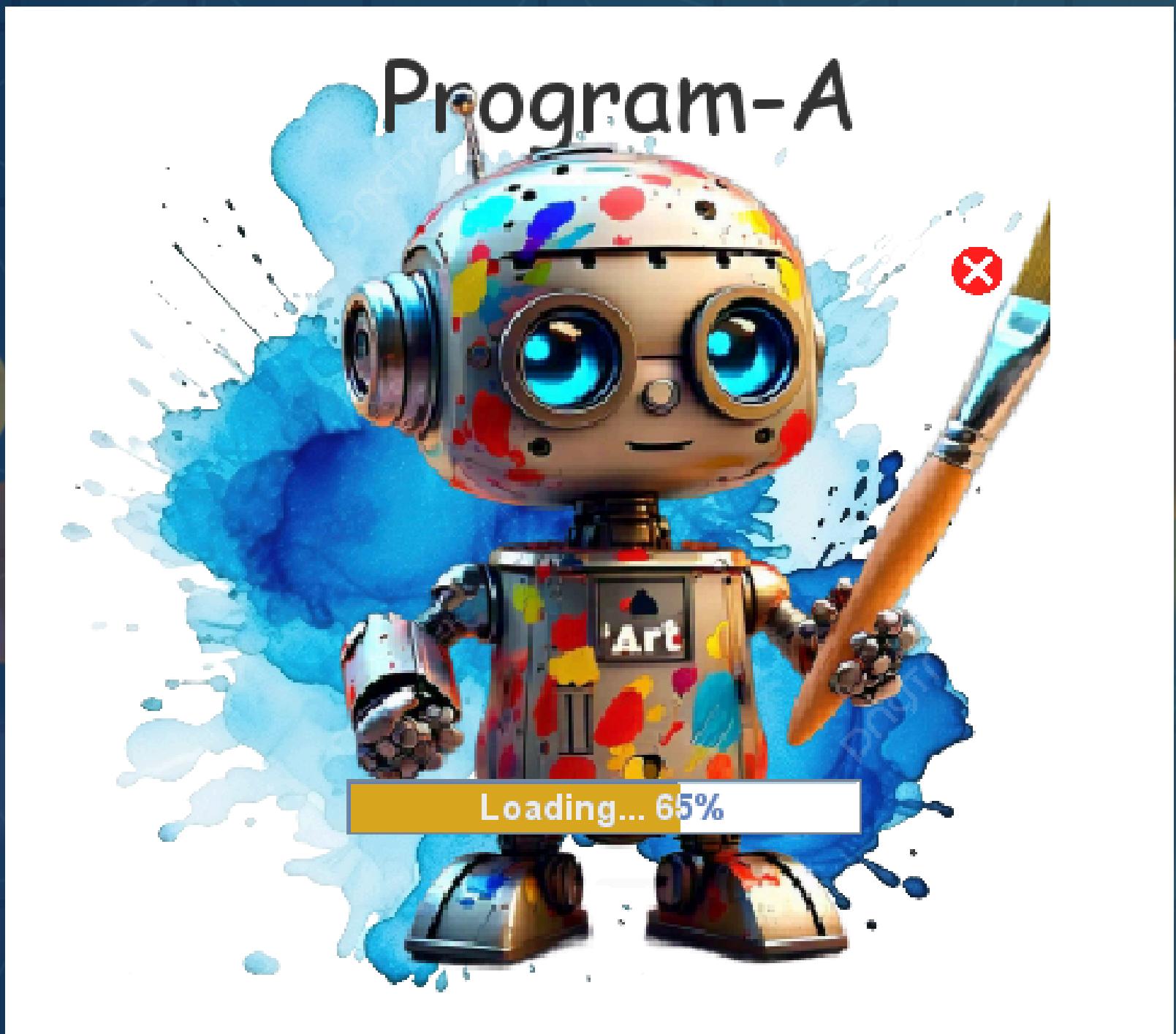
¿Como funciona?

En un análisis sintáctico, el parser generado por CUP utiliza estas constantes para identificar las partes del código (tokens) que provienen del lexer.

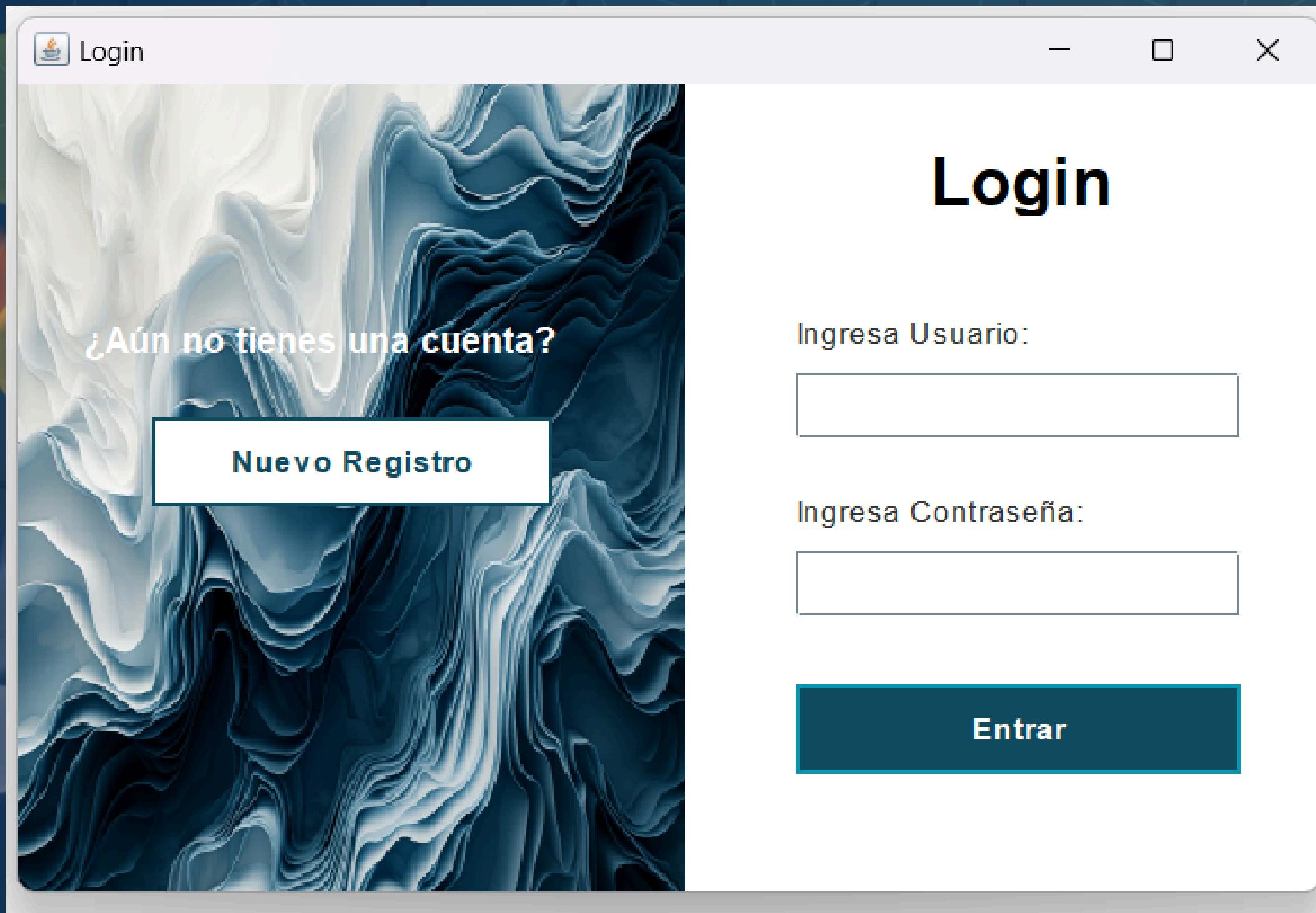
¿Cúal es su función?

almacenar los valores numéricos de cada tipo de token que el analizador sintáctico utilizará.

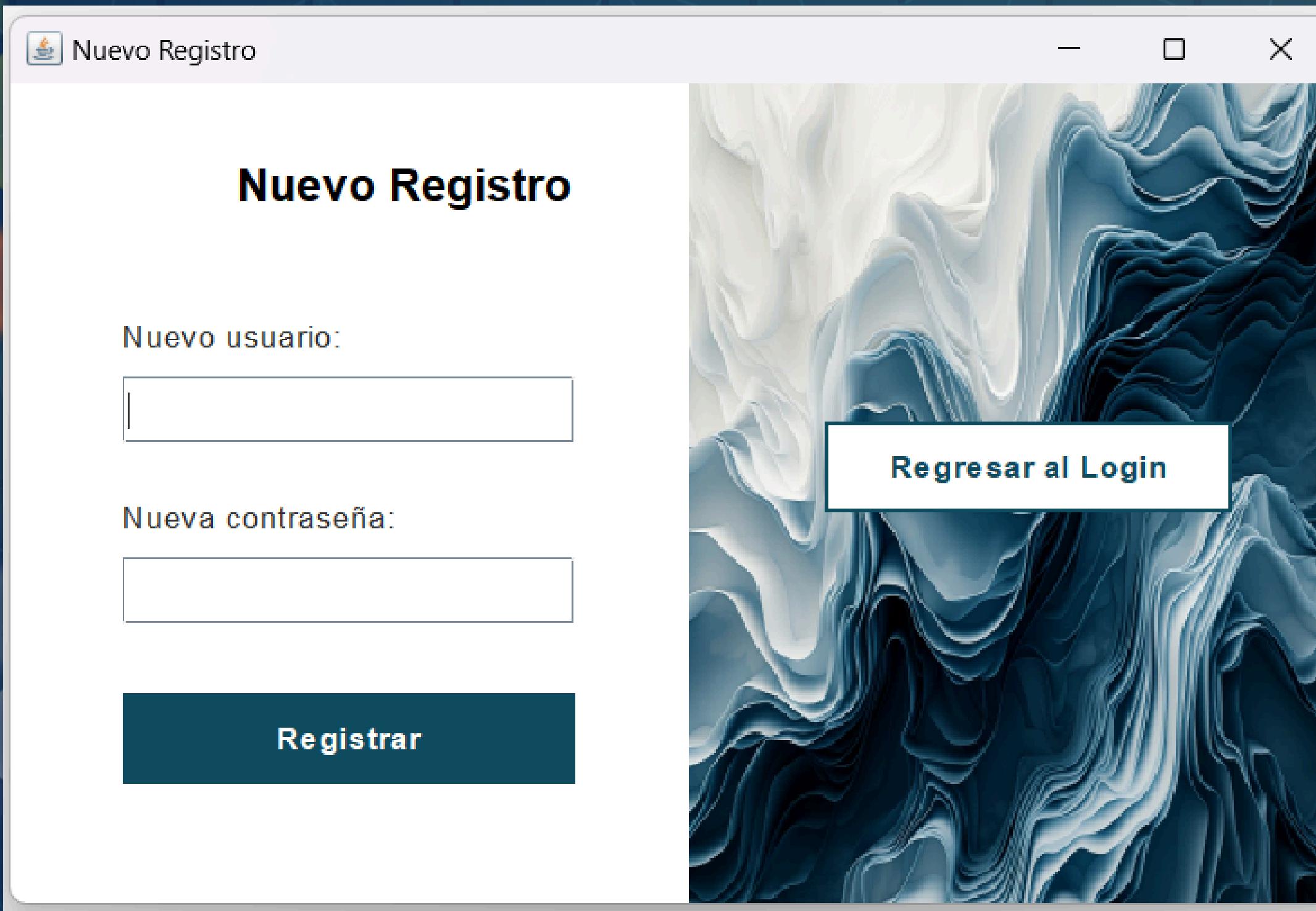
Sintaxis.java



LoginForm.java



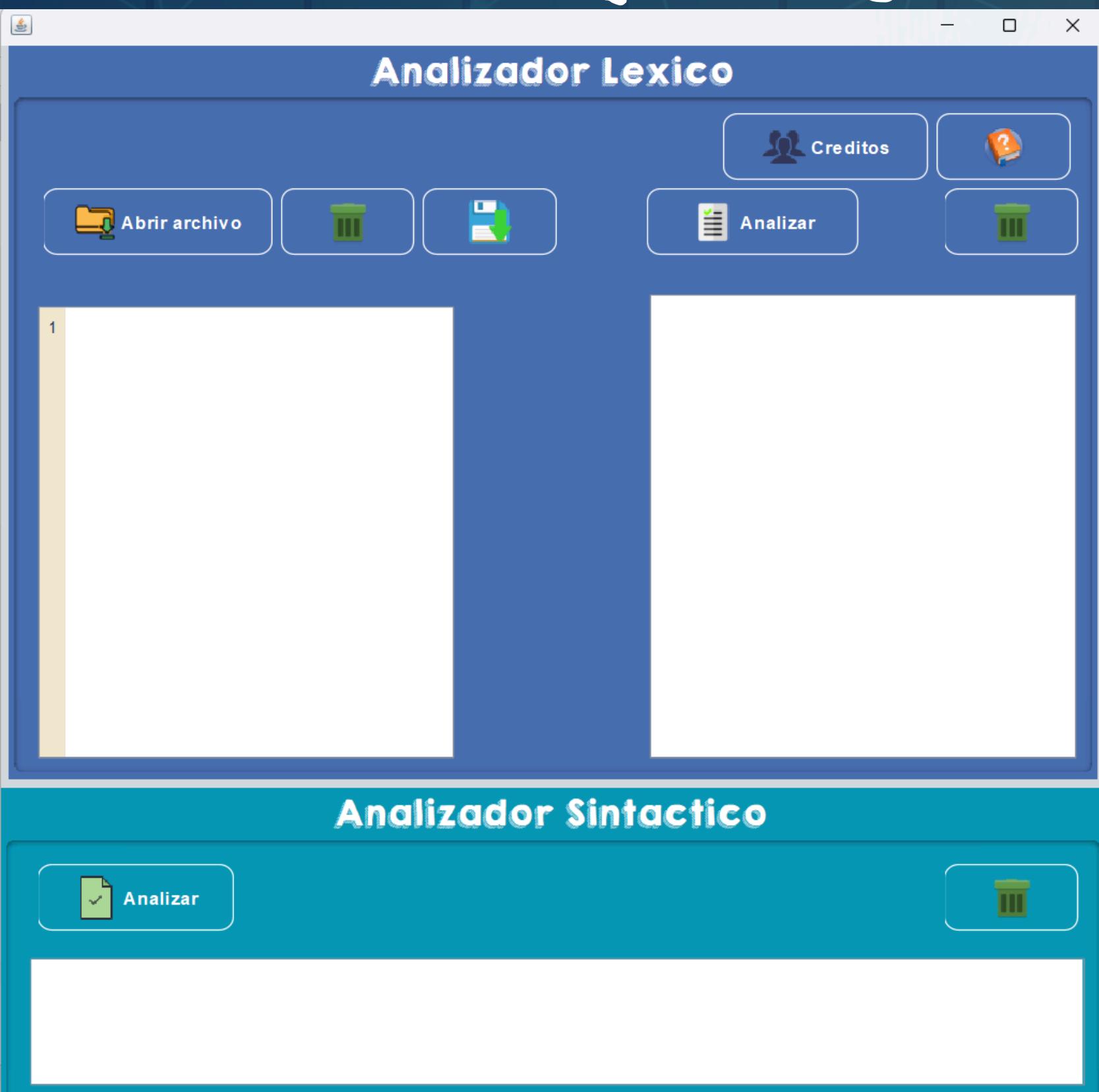
RegisterForm.java



FrmBienvenida.java



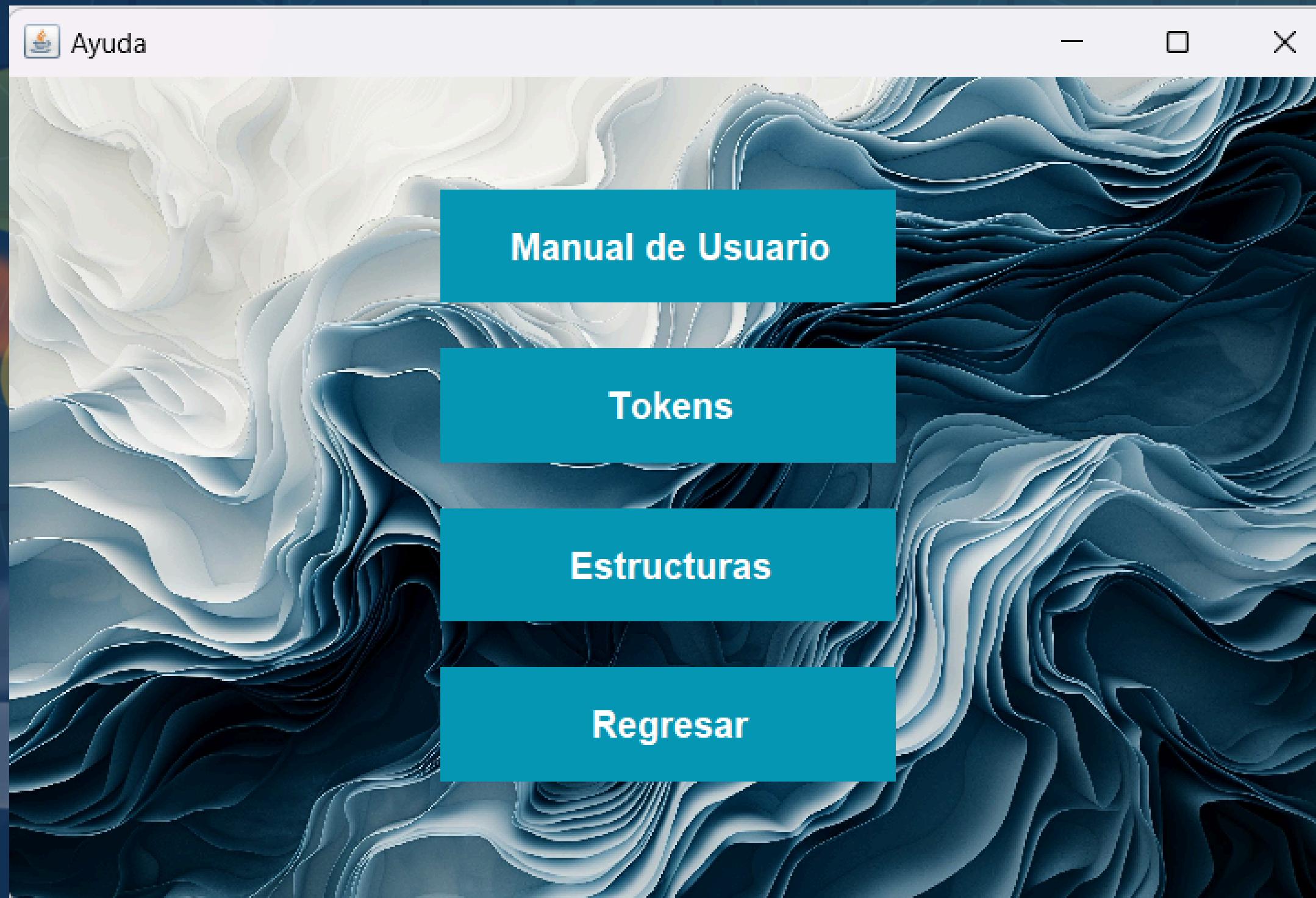
FrmPrincipal.java



Creditos.java



Ayuda.java





BIBLIOGRAFIA

Charlead [@charlead4113]. (s/f-a). JCup y JFlex | Analizador sintáctico con Java (explicación paso a paso). Youtube. Recuperado el 8 de enero de 2025, de <https://www.youtube.com/watch?v=4Z6Tnit810Y>

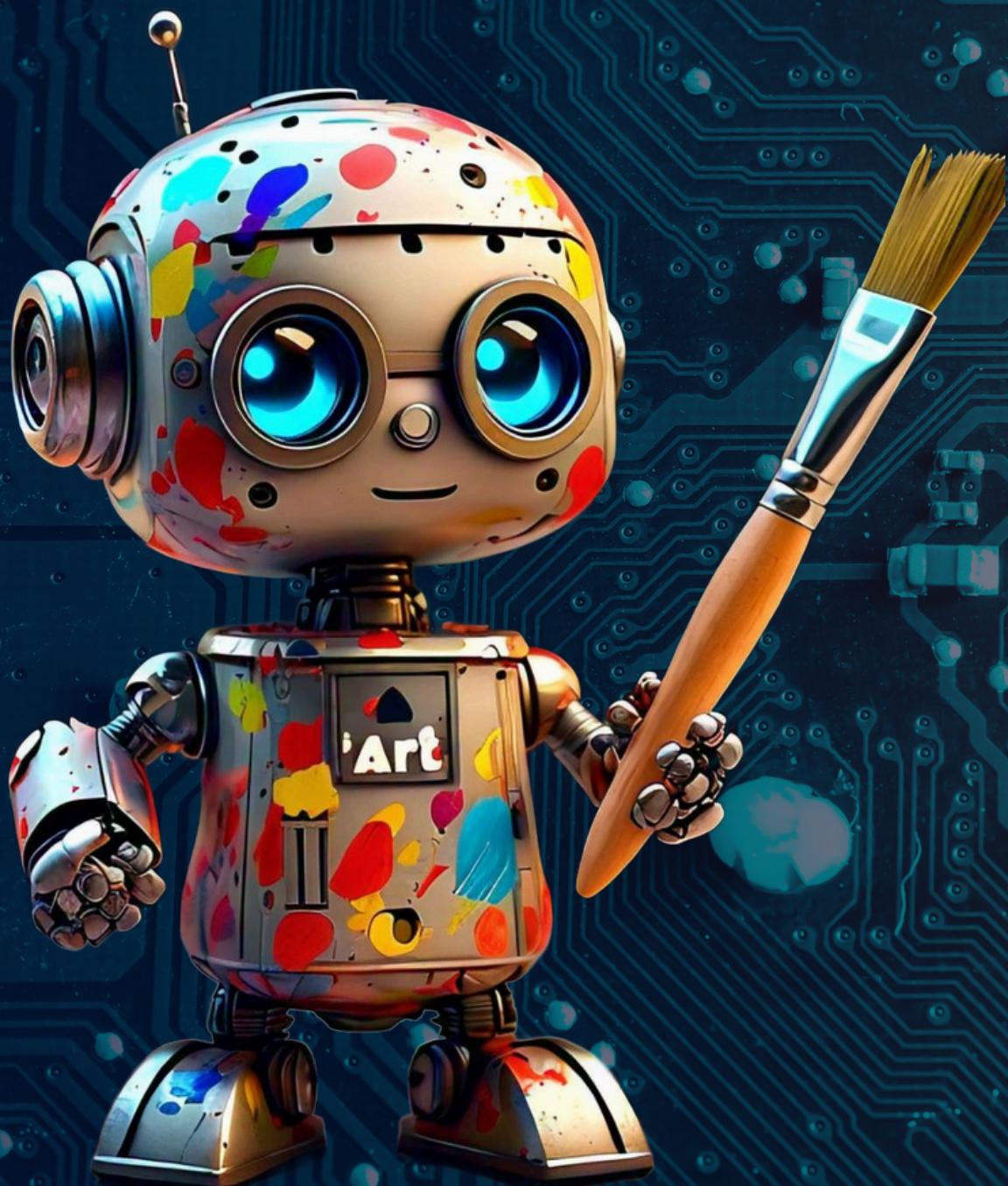
Charlead [@charlead4113]. (s/f-b). JFlex | Analizador léxico con Java (explicación paso a paso). Youtube. Recuperado el 8 de enero de 2025, de <https://www.youtube.com/watch?v=5mIRrn2yEe8>

Download java-cup-11.jar: java cup «j «Jar File Download. (s/f). Java2s.com. Recuperado el 8 de enero de 2025, de <http://www.java2s.com/Code/Jar/j/Downloadjavacup11jar.htm>

Klein, G. (s/f). JFlex - download. Jflex.de. Recuperado el 8 de enero de 2025, de <https://jflex.de/download.html>

NetBeans, A. (s/f). Apache NetBeans releases. Apache.org. Recuperado el 8 de enero de 2025, de <https://netbeans.apache.org/front/main/download/>

Conclusión



**GRACIAS POR
SU ATENCIÓN**