



## **Instituto Tecnológico de Costa Rica**

Sede Interuniversitaria, Alajuela

Escuela de Ingeniería en Computación

Compiladores E Intérpretes - IC5701

# **Reporte Proyecto Analizador Léxico Java Tropicalizado**

## **Avance #3**

Natán Fernández de Castro - 2017105774

Fabrizio Alvarado Barquero - 2017073935

Emanuelle Jiménez Sancho - 2017136727

Prof. Emmanuel Ramírez Segura

Fecha de Entrega: 26 de Mayo, 2020

Semestre I

## 1. Gramática

<i>Programa</i>	→ <i>ClasePrincipal DeclClase*</i>
<i>ClasePrincipal</i>	→ <b>publico</b> <b>clase</b> <i>iden</i> { <b>publico</b> <b>estatico</b> <b>vacio</b> <b>principal</b> ( <i>Hilera</i> [] <i>iden</i> ) { <i>Declaracion</i> } }
<i>DeclClase</i>	→ <b>class</b> <i>iden</i> { <i>DeclVar*</i> <i>DeclMetodo*</i> } → <b>class</b> <i>iden</i> <b>extiende</b> <i>iden</i> { <i>DeclVar*</i> <i>DeclMetodo*</i> }
<i>DeclVar</i>	→ <i>Tipo</i> <i>iden</i> ;
<i>DeclMetodo</i>	→ <b>public</b> <i>Tipo</i> <i>iden</i> ( <i>ListaFormal</i> ) { <i>DeclVar*</i> <i>Declaracion*</i> <b>retornar</b> <i>Expren</i> ; }
<i>ListaFormal</i>	→ <i>Tipo</i> <i>iden</i> <i>RestricFormal*</i> →
<i>RestricFormal</i>	→ , <i>Tipo</i> <i>iden</i>
<i>Type</i>	→ <b>ent</b> [] → <b>booleano</b> → <b>ent</b> → <i>iden</i>
<i>Declaracion</i>	→ { <i>Declaracion*</i> } → <b>si</b> ( <i>Expren</i> ) <i>Declaracion</i> <b>sino</b> <i>Declaracion</i> → <b>mientras</b> ( <i>Expren</i> ) <i>Declaracion</i> → <b>Sistema.salida.imprimirnl</b> ( <i>Expren</i> ) ; → <b>iden</b> = <i>Expren</i> ; → <b>iden</b> [ <i>Expren</i> ] = <i>Expren</i>
<i>Exp</i>	→ <i>Expren</i> <i>op</i> <i>Expren</i> → <i>Expren</i> [ <i>Expren</i> ] → <i>Expren</i> . <b>largo</b> → <i>Expren</i> . <i>iden</i> ( <i>ListaExp</i> ) → <b>ENTERO_LITERAL</b> → <b>verdad</b> → <b>falso</b> → <i>iden</i> → <b>esto</b> → <b>nuevo</b> <b>ent</b> [ <i>Expren</i> ] → <b>nuevo</b> <i>iden</i> ( ) → ! <i>Expren</i> → ( <i>Expren</i> )
<i>ExpLista</i>	→ <i>Expren</i> <i>RestricExp*</i> →
<i>RestricExp</i>	→ , <i>Expren</i>

## 2. Reconocimiento de Tokens

Token	Expresión Regular
identificadores	[A-Za-z0-9_]+
entero	[0-9]+
operadores	-\  \  \* \ +=
operador lógico	== > = <= > < &&
hilera	"[A-Za-z0-9]+"
Símbolo Especial	.; \\( \\) {\\} \\[\\] \\.
abstracto	[abstracto]
asertar	[asertar]
booleano	[booleano]
romper	[romper]
byte	[byte]
caso	[caso]
atrapar	[atrapar]
caracter	[caracter]
clase	[clase]
continuar	[continuar]
predeterminado	[predeterminado]
hacer	[hacer]
doble	[doble]
sino	[sino]
enum	[enum]
extender	[extender]
final	[final]
finalmente	[finalmente]
flotante	[flotante]

para	[para]
si	[si]

implementa	[implementa]
importar	[importar]
instanciade	[instanciade]
ent	[ent]
interfaz	[interfaz]
largo	[largo]
nativo	[nativo]
nuevo	[nuevo]
nulo	[nulo]
paquete	[paquete]
privado	[privado]
protegido	[protegido]
publico	[publico]
retornar	[retornar]
corto	[corto]
estatico	[estatico]
estrictofp	[estrictofp]
super	[super]
cambiar	[cambiar]
sincronizado	[sincronizado]
este	[este]
arroja	[arroja]
arrojan	[arrojan]
transitorio	[transitorio]

tratar	[tratar]
vacio	[vacio]
volatil	[volatil]
mientras	[mientras]

### 3. Capturas del código

The screenshot shows an IDE with a Java project named 'AnalizadorLexico'. The main class is 'AnalizadorLexico.java'. The code processes input lines and tokenizes them. The output window shows the following execution:

```

Escribe el programa:
Ent 0 0 0 0 0 0
Hilera 0 0
0 0 0 0
0 0 0 0 0 0
0 0 0 0 0
Respuesta 0 0
  
```

The code in the editor is as follows:

```

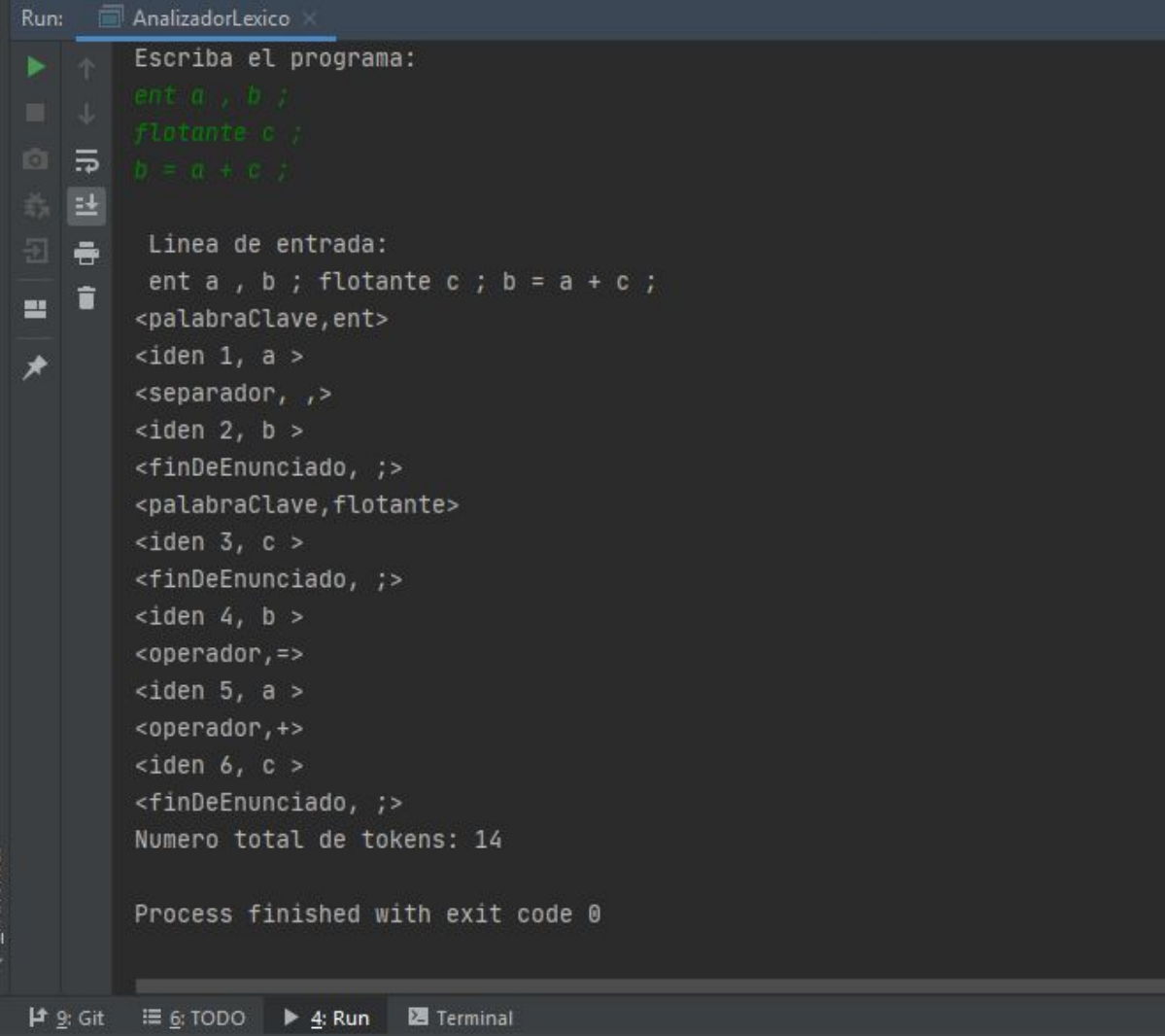
14 System.out.println("Escribe el programa: ");
15
16 for (int i = 0; i < input.length; i++){
17     input[i] = s.nextLine();
18     hilera = input[i];
19 }
20
21 System.out.println("\n linea de entrada: ");
22
23 String output[] = hilera.split(" ");
24
25 for (String output1 : output) {
26     System.out.print(" " + output1);
27 }
28
29 System.out.println("");
30
31 int id = 0, tokenNo = 0;
32
33 for (int i = 0; i < output.length; i++) {
34     if (null != output[i]) {
35         switch (output[i]) {
36             case "hilera":
37             case "ent":
38             case "flotante":
39             case "entero":
40             case "booleano":
41             case "retornar":
42             case "vacio":
43             case "nuevo":
  
```

Img#1: Aqui una captura de cómo es que nuestro programa procesa el código ingresado por el usuario.

```
Run: AnalizadorLexico x
C:\Program Files\Java\jdk-10.0.2\bin\java.exe - javadagent.0.\Program Files
>>> Número de líneas:
2
Escriba el programa:
ent a , b ;
Hilera c ;

Linea de entrada:
ent a , b ; Hilera c ;
<palabraClave,ent>
<iden 1, a >
<separador, ,>
<iden 2, b >
<finDeEnunciado, ;>
<palabraClave,Hilera>
<iden 3, c >
<finDeEnunciado, ;>
Numero total de tokens: 8
Process finished with exit code 0
```

## Ejemplo#1



The screenshot shows a VS Code terminal window titled "Run: AnalizadorLexico x". The terminal output is as follows:

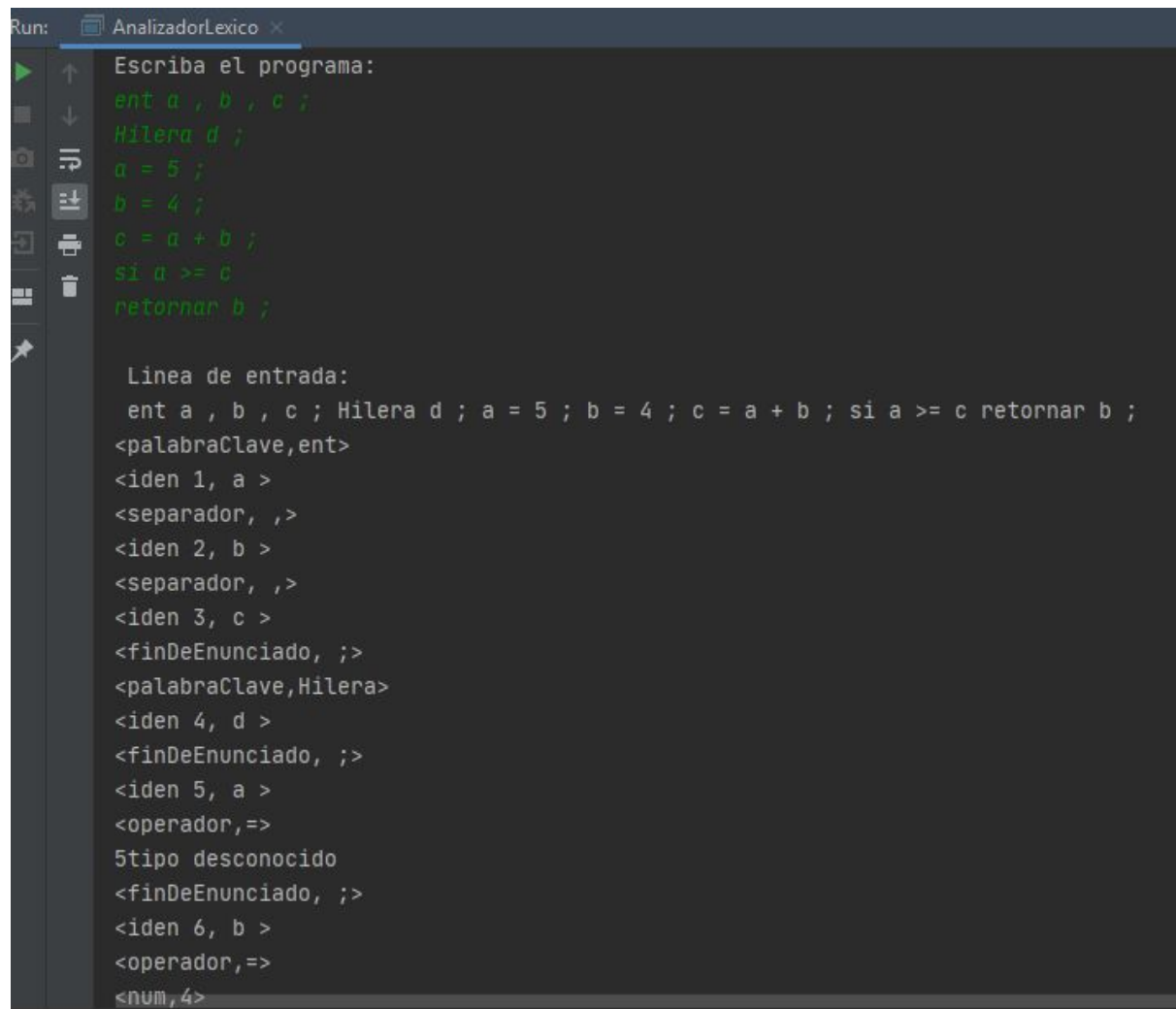
```
Run: AnalizadorLexico x
> Escriba el programa:
ent a , b ;
flotante c ;
b = a + c ;

Linea de entrada:
ent a , b ; flotante c ; b = a + c ;
<palabraClave,ent>
<iden 1, a >
<separador, ,>
<iden 2, b >
<finDeEnunciado, ;>
<palabraClave,flotante>
<iden 3, c >
<finDeEnunciado, ;>
<iden 4, b >
<operador,=>
<iden 5, a >
<operador,+>
<iden 6, c >
<finDeEnunciado, ;>
Numero total de tokens: 14

Process finished with exit code 0
```

The bottom status bar of VS Code shows "9: Git", "6: TODO", "4: Run", and "Terminal".

Ejemplo#2



The screenshot shows a software interface for a lexical analyzer. At the top, a tab labeled 'AnalizadorLexico' is active. The main window is divided into two sections. The top section, titled 'Escriba el programa:', contains a code snippet in a green monospaced font: `ent a , b , c ;`, `Hilera d ;`, `a = 5 ;`, `b = 4 ;`, `c = a + b ;`, `si a >= c`, and `retornar b ;`. The bottom section, titled 'Linea de entrada:', displays the full input string: `ent a , b , c ; Hilera d ; a = 5 ; b = 4 ; c = a + b ; si a >= c retornar b ;`. Below this, a list of tokens is shown, each on a new line, representing the output of the lexical analysis. The tokens are: `<palabraClave,ent>`, `<iden 1, a >`, `<separador, ,>`, `<iden 2, b >`, `<separador, ,>`, `<iden 3, c >`, `<finDeEnunciado, ;>`, `<palabraClave,Hilera>`, `<iden 4, d >`, `<finDeEnunciado, ;>`, `<iden 5, a >`, `<operador,=>`, `5tipo desconocido`, `<finDeEnunciado, ;>`, `<iden 6, b >`, `<operador,=>`, and `<num,4>`. A vertical toolbar on the left side of the window contains various icons for file operations and editing.

```
Run: AnalizadorLexico x
Escriba el programa:
ent a , b , c ;
Hilera d ;
a = 5 ;
b = 4 ;
c = a + b ;
si a >= c
retornar b ;

Linea de entrada:
ent a , b , c ; Hilera d ; a = 5 ; b = 4 ; c = a + b ; si a >= c retornar b ;
<palabraClave,ent>
<iden 1, a >
<separador, ,>
<iden 2, b >
<separador, ,>
<iden 3, c >
<finDeEnunciado, ;>
<palabraClave,Hilera>
<iden 4, d >
<finDeEnunciado, ;>
<iden 5, a >
<operador,=>
5tipo desconocido
<finDeEnunciado, ;>
<iden 6, b >
<operador,=>
<num,4>
```

Ejemplo#3