

# CARRERA INGENIERÍA DE SOFTWARE

# Docente:

Juan Vilca Castro

### **Curso:**

Compiladores

# **Alumnos:**

Butrón Soria, Jarol Milón Guzman, Gianella Mendoza Melgar, Rodrigo Loza Peralta, Christian

# Tema:

DOCUMENTACIÓN DE UN ANALIZADOR LÉXICO Y SINTÁCTICO, PARA LA CONSTRUCCIÓN DE UN COMPILADOR

Saturday 22<sup>nd</sup> October, 2016

### Resumen

El presente documento detalla la información para la construcción de un analizador léxico y sintáctico que formará parte del compilador para un propio lenguaje definido, construido bajo la plataforma de Java, asimismo se especifica la lista de palabras reservadas, tokens, matrices, operadores, delimitadores, identificadores, literales, constantes y definición formal del autómata.

### 1. Lista de *Tokens*

### 1.1. Palabras reservadas

- programa: bloque de partida para la ejecución del programa.
- entero: Tipo de dato entero de 32 Bits (4 bytes), conformado por números positivos.
- si: Estructura condicional o instrucción de control simple, evalúa una expresión dada.
- **osino**: Intrucción de control doble, permite especificar una acción cuando la condición es verdadera y distinta cuando la condición es falsa.
- mientras: Instrucción de repetición, evalúa una expresión, en caso de devolver verdadero iniciará un ciclo (bucle) hasta que la condición sea verdadera.
- leerTeclado: Permite la lectura de datos mediante teclado.
- escribirPantalla: Muestra la salida de datos en pantalla.

## 1.2. Operadores

#### 1.2.1. Operadores unarios

- Operador "-" (menor): Indica la negación de una variable o una expresión.
- Operador "+" (mayor): Indica un valor positivo de una variable o una expresión.

## 1.2.2. Operadores binarios

- Operador " + " (suma) : Operador de adición.
- Operador " " (resta): Operador de resta.
- Operador " \* " (multiplicación): Operador de multiplicación.
- Operador "/" (resta): Operador de división.
- Operador " % " (módulo): Operador de reciduo de división.
- Operador " < " (menor): Operador de relación menor.
- Operador " > " (mayor): Operador de relación mayor.
- Operador " <= " (menor e igual): Operador de relación menor e igual.
- Operador ">= " (Mayor igual): Operador de relación mayor e igual.
- Operador " = " (igual): Operador de asignación.
- Operador " == " (exactamente igual): Operador de relación de igualdad de valor.
- Operador "!= " (diferencia): Operador de relación diferente entre valores de datos.
- Operador " && " (OR): Operador OR lógico.
- Operador " && " (AND): Operador AND lógico.

#### 1.2.3. Identificadores

- Identificador (a-z): Secuencia de caracteres de tipo letras en minúsculas del abecedario conformada por 26 letras.
- Identificador (A-Z): Secuencia de caracteres de tipo letras en mayúscula del abecedario conformada por 26 letras.
- Identificador (0-9): dígitos conformados desde el 0 al 9.
- Identificador (\_) guión bajo o underscore: dígitos conformados desde el 0 al 9.
- Identificador (\$): Símbolo de dólar,código ASCII 36, .

#### 1.2.4. Literales de cadena

• expresión: representación de una o varias palabras con alguna idea.

#### 1.2.5. Constantes numéricas

• Constante (0-9): Representación de números enteros del 0 al 9.

# 2. Autómata

# 2.1. Representación del autómata finito determinista

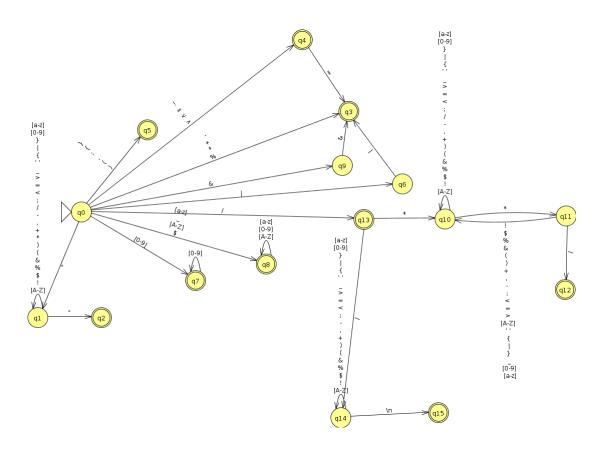


Figura. 1: Diagrama del autómata finito determinista.

# 2.2. Representación del autómata finito determinista

estadoActual	estadoSiguiente	caracteresAceptados
0	8	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_\$
0	7	0123456789
0	5	08;:
0	4	<>!=
0	3	+-96*
0	9	&
0	6	I
0	13	l .
0	1	н
1	1	$/ \&+-\%^*<> =()\{,;^*\_\$0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz=0\} $
1	2	и
4	3	=
6	3	I
7	7	0123456789
8	8	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
9	3	&
10	10	/ &+-%<>!=0{},;_\$0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
10	11	*
11	12	l .
11	10	* &+-%<>!=0{},;_\$0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
13	14	I
13	10	*
14	14	/ &+-%*<>!=(){},;_\$0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkImnopqrstuvwxyz
14	15	

Figura. 2: Representación del autómata finito determinista

# 3. Analizador Sintáctico

# 3.1. Gramática

- 1. R0 : < P > -> Programa() < BS >
- 2. *R*1 :< *BS* > **->**< *S*\* >
- 3. R1 : < BS > -> < S\* >
- 4. *R*2 :< *S*\* > **->** < *S* >< *S*\* >
- 5. *R*3 :< *S*\* > **->** *lambda*
- 6. *R*4 :< *S* > **->** < *SD* >
- 7. R5 : < S > -> < SA >
- 8. *R*6 :< *S* > **->** < *SC* >
- 9. *R*7 :< *S* > **->** < *SM* >
- 10. *R*8 :< *S* > **−>** < *SL* >
- 11. *R*9 :< *S* > **->** < *SE* >
- 12. *R*10 :< *S* > **->** < *SHM* >
- 13. *R*11 :< *SD* > **->** < *TD* >< *VA* >< *RD*\* >;

14. 
$$R12 :< RD* > ->, < VA > < RD* >$$

15. 
$$R13 :< RD* > -> \lambda$$

16. 
$$R14 : \langle VA \rangle \rightarrow IDENTIFICADOR \langle A \rangle$$

17. 
$$R15 :< A > -> = < EM >$$

18. 
$$R16 :< A > -> \lambda$$

19. 
$$R17 :< TD > ->CONST_NUMERICA$$

20. 
$$R18 : \langle SA \rangle \rightarrow IDENTIFICADOR = \langle EM \rangle$$
;

21. 
$$R19 :< SM > -> Mientras(< EM >) < SubB >$$

22. 
$$R20 : < SubB > -> < S >$$

24. 
$$R22 :< SC > -> Si(< EM >) < SubB >< ELSE >$$

26. 
$$R24 :< ELSE > -> \lambda$$

27. 
$$R25 : \langle SHM \rangle \rightarrow Hacer \langle subB \rangle Mientras(\langle EM \rangle);$$

28. 
$$R26 : < EM > -> < T > < T* >$$

29. 
$$R27 :< T* > -> \lambda$$

33. 
$$R31 : < T > -> < F > < F* >$$

34. 
$$R32 : < F* > -> < OP2 > < F > < F* >$$

35. 
$$R33 :< F* > -> \lambda$$

36. 
$$R34 : < F > -> (< EM >)$$

37. 
$$R35 :< F > ->CONST_NUMERICA$$

38. 
$$R36 : < F > -> IDENTIFICADOR$$

42. 
$$R40 :< SL > -> LeerTeclado(IDENTIFICADOR)$$

43. 
$$R41 : \langle SE \rangle \rightarrow EscribirPantalla(\langle E \rangle);$$

45. 
$$R43 : \langle E \rangle \rightarrow IDENTIFICADOR$$

46. 
$$R44 : \langle E \rangle - LITERAL\_CADENA$$

47. 
$$R45 : \langle E \rangle - CONST_NUMERICA$$

48. 
$$R46 :< E > -> \lambda$$

# Terminología

### 1. Terminal

- Programa
- (
- )
- {
- }
- ;
- ,
- =
- Si
- Osino
- Hacer
- Mientras
- -
- –
- /
- \*
- %
- LeerTeclado
- EscribirPantalla
- IDENTIFICADOR
- CONST\_NUMERICA
- LITERAL\_CADENA

### 2. No Terminales

- Programa
- $\bullet$  < P > Programa
- < BS > (bloquedesentencias)
- $\langle S* \rangle$  (Listadesentencias)
- < SD > (Sentenciadedeclaración)
- < TD > (tipodedato)
- < VA > (variableasignada)
- < RD\* > (restodedatos)
- $\langle A \rangle$  (asignacin)
- < EM > (expresinmatemática)
- < SA > (sentenciadeasignación)
- < SC > (sentenciadecondicional)
- < SM > (sentenciademientras)
- < SL > (sentenciadelectura)
- < SE > (sentenciadeescritura)
- < SubB > (subbloque)
- < ELSE > (condicionalalternativa)
- < SHM > (sentenciahacermientras)

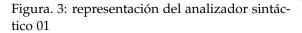
- $< T > (T\acute{e}rmino)$
- < T\* > (ListadeTérmino)
- < OP1 > (Operadoresnivel1)
- < OP2 > (Operadoresnivel2)
- < F > (Factor)
- < F\* > (ListadeFactores)
- < E > (escribir)

### 3. lamdba

• lambda ( $\lambda$ ) o vacío.

# 3.2. Representación del analizador sintáctico

noTerminal	derivacion				
	tipoToken	lexema	valor		
		Programa			
valor <p></p>		(			
		)			
			<bs></bs>		
	tipoToken	lexema	valor		
		{			
valor <bs></bs>			<s*></s*>		
		}			
	valor				
valor <s*></s*>	<s></s>				
	<s*></s*>				
valor <s*></s*>					
	valor				
valor <s></s>	<sd></sd>				
	valor				
valor <s></s>	<sa></sa>				
valor <s></s>	valor				
vaior <5>	<sc></sc>				
valor <s></s>	valor				
vaior <5>	<sm></sm>				
valor <s></s>	valor				
valor <s></s>	<sl></sl>				



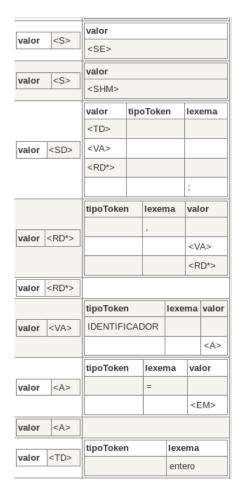


Figura. 4: representación del analizador sintáctico 02

	tipoToken		lexe	ma	valor	
	IDENTIFICA	ADOR	·exe		22101	
	IDENTIFICADOR					
valor <sa></sa>			=			
					<em></em>	
			;			
	tipoToken			valor		
		Mientras				
		(				
valor <sm></sm>					<em></em>	
		)				
				<5	<subb></subb>	
valor						
valor <subb></subb>	<s></s>					
	valor					
valor <subb></subb>	valor <bs></bs>					
	tipoToken		ma valo		or	
		Si				
		(				
valor <sc></sc>			<em></em>		M>	
		)				
			<subb></subb>		ubB>	
				<ei< td=""><td>LSE&gt;</td></ei<>	LSE>	
	tipoToken	lexe	na	val	or	
valor <else></else>		Osin	Osino			
				<s< td=""><td>ubB&gt;</td></s<>	ubB>	

valor <else></else>					
	tipoToken	lexema		valor	
		Hacer			
				<subb></subb>	
		Mientras			
valor <shm></shm>		(			
				<em></em>	
		)			
		;			
	valer				
	valor				
valor <em></em>	<t></t>				
	<t*></t*>				
	valor				
	<op1></op1>				
valor <t*></t*>	<t></t>				
	<t*></t*>				
valor <t*></t*>					
	tipoToken	lex		ema	
valor <op1></op1>			+		
untan dané:	tipoToken le:		lex	kema	
valor   <op1></op1>			-		
	valor				
valor <t></t>	<f></f>				
	<f*></f*>				

Figura. 6: representación del analizador sintáctico 03

Figura. 6: representación del analizador sintáctico 04

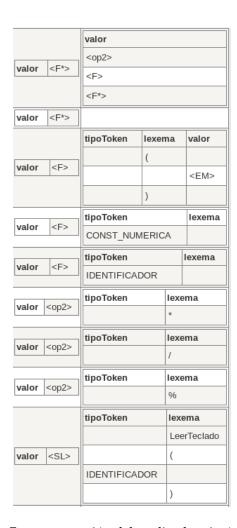
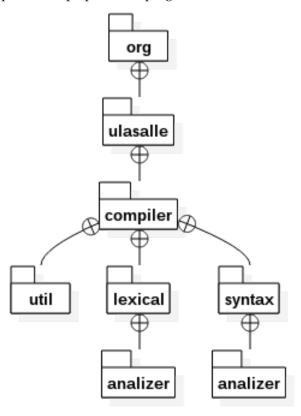


Figura. 7: representación del analizador sintáctico 05

	tipoToken	lexema		valor
		EscribirPantalla		
		(		
valor <se></se>				<e></e>
		)		
		;		
	valor			
valor <e></e>	<em></em>			
	tipoToken		lexema	
valor <e></e>	IDENTIFICADOR			
	tipoToken		lexema	
valor <e></e>	LITERAL_CADENA			
	tipoToken lex		lex	ema
valor <e></e>	CONST_NUMERICA			

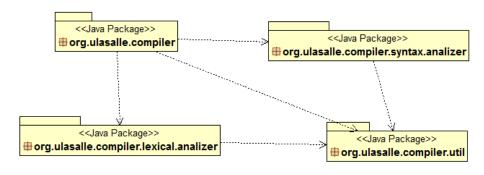
Figura. 8: representación del analizador sintáctico 06

Figura. 9: Jerarquía de los paquetes del programa mediante un Diagrama de paquetes.



Fuente: Propia.

Figura. 10: Diagrama de paquetes.



Fuente: Propia.

<<Java Package>> org.ulasalle.compiler.util <<Java Class>> <<.lava Class>> **⊙** Token CargadorPropiedades <<.lava Interface>> org.ulas alle.compiler.util **⊕**Resultado lexema: String o<sup>c</sup>Token() escribirJSON(Object, String):voidleerJSON(Class<?>, String):Object Token(Token) ©Token(String) ● getTipoToken():TipoToken <<Java Interface>> setTipoToken(TipoToken):void Analizador getLexema():String
 setLexema(String):void org.ulas alle.compiler.util o equals(Object):boolean analizar(List<Resultado>):Respuesta -tipoToken 0..1 <<Java Class>> <<Java Enumeration>> **G**Respuesta **⊙** TipoToken org.ulas alle.compiler.uti org.ulas alle.compiler.util nombreArchivo: String IDENTIFICADOR: TipoToken <<.lava Interface>> Respuesta(String,List<Resultado>,List<Error>) OPERADOR: TipoToken
DELIMITADOR: TipoToken **10** Error getNombreArchivo():String
 getResultados():List<? extends Resultado> org.ulas alle.compiler.util PALABRA\_RESERVADA: TipoToken
CONST\_NUMERICA: TipoToken getErrores():List<? extends Error> getDescripcion():String FLITERAL\_CADENA: TipoToken
COMENTARIO: TipoToken SCOMENTARIO\_MULTILINEA: TipoToken

Figura. 11: Clases utiles.

Fuente: Propia.

<sup>c</sup>TipoToken()

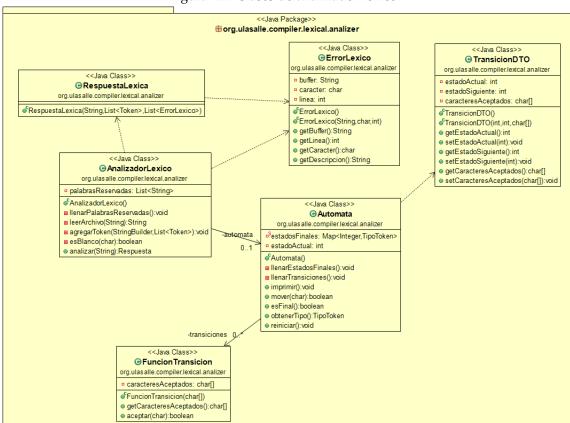


Figura. 12: Clases de analizador léxico.

Fuente: Propia.

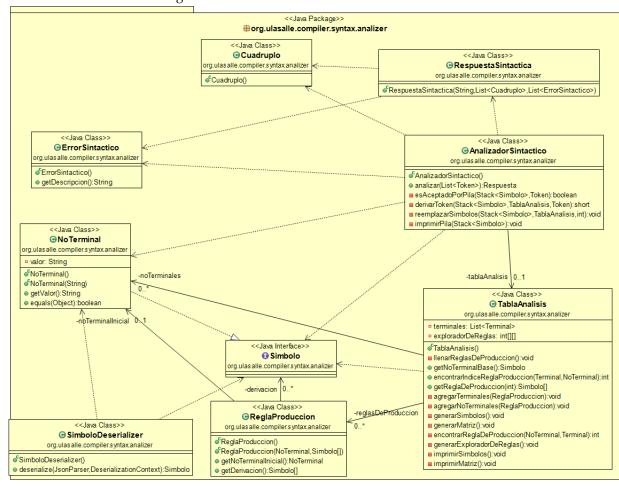


Figura. 13: Clases de analizador sintactico.

Fuente: Propia.