Elementos léxicos del lenguaje de programación

Java

Elementos léxicos del lenguaje de programación Java

- Palabras reservadas
- Identificadores
- Literales
- Operadores
- Delimitadores
- Comentarios

Apéndices

- Operadores de Java
- Sintaxis de Java

Elementos léxicos de Java

Token

Componente léxico de un lenguaje de programación

Palabras reservadas

Palabras que tiene un significado concreto en el lenguaje de programación, sin necesidad de que se lo asignemos nosotros.

```
abstract  continue  for         new       switch
boolean   default   goto        null      synchronized
break     do        if          package   this
byte      double    implements  private   threadsafe
byvalue   else      import      protected throw[s]
case      extends   instanceof  public    transient
catch     false     int         return    true
char      final     interface   short     try
class     finally   long        static    void
const     float     native      super     while

cast      future    generic     inner
operator  outer     rest        var
```

Identificadores

Palabras que podemos utilizar para denominar algo en el lenguaje.

Identificadores en Java

- El primer símbolo del identificador será un carácter alfabético (a, …, z, A, …, Z, '_', '$') pero no un dígito. Después de ese primer carácter, podremos poner caracteres alfanuméricos (a, …, z) y (0, 1, …, 9), signos de dólar '$' o guiones de subrayado '_'.
- Los identificadores no pueden coincidir con las palabras reservadas.
- Las mayúsculas y las minúsculas se consideran diferentes.
- El signo de dólar y el guión de subrayado se interpretan como una letra más.

  Ejemplos válidos

  a, pepe, r456, tu_re_da, AnTeNa, antena, usd$

  Ejemplos no válidos

  345abc, mi variable, Nombre.Largo, cañada, camión

Literal

Especificación de un valor concreto de un tipo de dato.

Números enteros
21 (int), 21L (long), 077 (en octal), 0xDC00 (en hexadecimal)

Números reales
3.14 (double), 3.14f (float), 3.14d (double), 2e12, 3.1E12

Valores booleanos
true (verdadero), false (falso)

Caracteres
'p', '\u????' (código UNICODE en hexadecimal), '\t' (tabulador)…

Cadenas de caracteres
"mensaje", "" (cadena vacía)

Operadores

Igual que en Matemáticas, realizan una acción específica:

- Suelen estar definidos en el núcleo del compilador
  (aunque también pueden estar definidos en bibliotecas externas)
- Suelen representarse con tokens formados por símbolos.
- Suelen utilizar notación infija.
- Pueden aplicarse a uno o varios operandos (argumentos).
- Suelen devolver un valor.

Operadores en Java, por orden de precedencia

```
. [ ] ( )
++ --
! ~ instanceof
new
* / %
+ -
<< >> >>>
< > <= >= == !=
& ^ |
&& ||
? :
= op= ,
```

Delimitadores

Símbolos utilizados como separadores de las distintas construcciones de un lenguaje de programación (esto es, los signos de puntuación de un lenguaje de programación.

`( )` PARÉNTESIS: Listas de parámetros en la definición y llamada a métodos, precedencia en expresiones, expresiones para control de flujo y conversiones de tipo.

`{ }` LLAVES: Inicialización de arrays, bloques de código, clases, métodos y ámbitos locales.

`[ ]` CORCHETES: Arrays.

`;` PUNTO Y COMA: Separador de sentencias.

`,` COMA: Identificadores consecutivos en una declaración de variables y sentencias encadenadas dentro de una sentencia for.

`.` PUNTO: Separador de nombres de paquetes, subpaquetes y clases; separador entre variables y métodos/miembros.

Comentarios

Aclaración que el programador incluye en el texto del programa para mejorar su inteligibilidad.

En Java hay tres tipos de comentarios:

```
// Comentario de una sola línea

/* Comentario de una o más líneas */

/** Comentario de documentación, una o más líneas */
```

> La herramienta `javadoc` genera automáticamente en HTML la documentación del código a partir de los comentarios `/** … */`

Apéndices

P	A	Operador	Operando(s)	Operación
15	I	. [ ] (args) ++, - -	Objeto, método (miembro) Array (indice) Método, lista de argumentos variable	Acceso a un miembro del objeto Acceso a un elemento de un array Llamada a un método Post incremento, post decremento
14	D	++,- - +,- ~ !	Variable Número Entero Booleano	Pre incremento, Pre decremento Cambio de signo (-) Complemento a nivel de bit NOT booleano
13	D	new (type)	Clase, lista de argumentos Tipo, cualquier tipo	Creación de objetos Cast (conversión de tipos)
12	I	* ,/, %	Número, número	Multplicación, división, módulo. Válido tb para fp
11	I	+,- +	Numero, número String, cualquiera	Suma, resta Concatenación de cadenas
10	I	<< >> >>>	Entero, entero Entero, entero Entero, entero	Desplazamiento a izquierda Desplazamiento a derecha con signo Desplazamiento a derecha con ceros
9	I	<, <= >, >= instance of	Número, Número Número, Número Referencia, tipo	Menor que, menor igual que Mayor que, mayor igual que Comparación de tipo
8	I	== != == !=	Primitiva , primitiva Primitiva , primitiva Referencia, referencia Referencia, referencia	Igual (tiene el mismo valor) No igual (diferente valor) Igual (mismo objeto) No igual (diferente objeto)
7	I	& &	Entero, entero Booleano, booleano	And booleano a nivel de bits And Booleano
6	I	^ ^	Entero, Entero Booleano, Booleano	XOR booleano a nivel de bits XOR Booleano
5	I	| |	Entero, Entero Booleano, Booleano	OR booleano a nivel de bits OR Booleano
4	I	&&	Booleano, Booleano	AND Condicional
3	I	||	Booleano, Booleano	OR Condicional
2	D	?:	Booleano, otro, otro	Operador condicional (if)
1	D	= *=,/=,%= +=,-= ,<<=,>>= >>>=, &=,^=,|=	Variable, otro	Asignación con operación

P: Precedencia
A : Asociatividad (I=Izquierda, D=Derecha)

Elemento	Objetivo	Sintaxis
Asignación	Evaluación de una expresión y almacenamiento del valor obtenido como resultado	var **=** expr**;** expr**++;**
Llamada	Llamada a un método	method**();**
Instanciación	Creación de un objeto	**new** Type**( )**
Secuencia	Grupo de instrucciones	**{** instrucciones **}**
Vacía	No hacer nada	**;**
Etiqueta	Etiquetado de una instrucción	etiqueta**:** instrucción
Variable	Declaración de una variable	[final] tipo nombre [=valor] [, nombre [=valor]]…;
if	Condicional	**if** (expr) instrucción [**else** instrucción]

switch	Condicional	**switch** (expr) {
		[**case** expr **:** instrucciones]…
		[**default :** instrucciones]
		}

while	Bucle	**while** (expr) instrucción
do	Bucle	**do** instrucción **while** (expr);
for	Bucle	**for** (init; test; increment) instrucción
break	Salir de un bloque	**break** [etiqueta] ;
continue	Reiniciar un bucle	**continue** [etiqueta];
return	Resultado de un método	**return** [expr];
synchronized	Sección crítica	**synchronized** (expr) {instrucciones}
throw	Lanzamiento de excepciones	**throw** expr;
try	Manejo de excepciones	**try** {instrucciones} [**catch** (tipo) {instrucciones}]… [**finally** {instrucciones}]