

# Los elementos del lenguaje Java

La sintaxis de un lenguaje define los elementos de dicho lenguaje y cómo se combinan para formar un programa. Los elementos típicos de cualquier lenguaje son los siguientes:

- Identificadores: los nombres que se dan a las variables
- Tipos de datos
- Palabras reservadas: las palabras que utiliza el propio lenguaje
- Sentencias
- Bloques de código
- Comentarios
- Expresiones
- Operadores

A lo largo de las páginas que siguen, examinaremos en detalle cada uno de estos elementos.

## Identificadores

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores:

- Todos los identificadores han de comenzar con una letra, el carácter subrayado ( `_` ) o el carácter dollar ( `$` ).
- Puede incluir, pero no comenzar por un número
- No puede incluir el carácter espacio en blanco
- Distingue entre letras mayúsculas y minúsculas
- No se pueden utilizar las palabras reservadas como identificadores

Además de estas restricciones, hay ciertas convenciones que hacen que el programa sea más legible, pero que no afectan a la ejecución del programa. La primera y fundamental es la de encontrar un nombre que sea significativo, de modo que el programa sea lo más legible posible. El tiempo que se pretende ahorrar eligiendo nombres cortos y poco significativos se pierde con creces cuando se revisa el programa después de cierto tiempo.

Tipo de identificador	Convención	Ejemplo
nombre de una clase	Comienza por letra mayúscula	String, Rectangulo, CinematicaApplet
nombre de función	Comienza con letra minúscula	calcularArea, getValue, setColor
nombre de	Comienza por letra	area, color, appletSize

variable	minúscula	
nombre de constante	En letras mayúsculas	PI, MAX_ANCHO

## Comentarios

Un comentario es un texto adicional que se añade al código para explicar su funcionalidad, bien a otras personas que lean el programa o al propio autor como recordatorio. Los comentarios son una parte importante de la documentación de un programa. Los comentarios son ignorados por el compilador, por lo que no incrementan el tamaño del archivo ejecutable; se pueden por tanto, añadir libremente al código para que pueda entenderse mejor.

La programación orientada a objetos facilita mucho la lectura del código, por lo que lo que no se precisa hacer tanto uso de los comentarios como en los lenguajes estructurados. En Java existen tres tipos de comentarios

- Comentarios en una sola línea
- Comentarios de varias líneas
- Comentarios de documentación

Como podemos observar un comentario en varias líneas es un bloque de texto situado entre el símbolo de comienzo del bloque `/*`, y otro de terminación del mismo `*/`. Teniendo en cuenta este hecho, los programadores diseñan comentarios como el siguiente:

```
/*-----|
|   (C) Ángel Franco García   |
|   fecha: Marzo 1999         |
|   programa: PrimeroApp.java |
|-----*/
```

Los comentarios de documentación es un bloque de texto situado entre el símbolo de comienzo del bloque `/**`, y otro de terminación del mismo `*/`. El programa *javadoc* utiliza estos comentarios para generar la documentación del código.

```
/** Este es el primer programa de una
serie dedicada a explicar los fundamentos del lenguaje Java */
```

Habitualmente, usaremos comentarios en una sola línea `//`, ya que no tiene el inconveniente de aprendernos los símbolos de comienzo y terminación del bloque u olvidarnos de poner este último, dando lugar a un error en el momento de la compilación. En la ventana de edición del Entorno Integrado de Desarrollo (IDE) los comentarios se distinguen del resto del código por el color del texto.

```
public class PrimeroApp{
    public static void main(String[] args) {
//imprime un mensaje
        System.out.println("El primer programa");
    }
}
```

```
}  
}
```

Un procedimiento elemental de depuración de un programa consiste en anular ciertas sentencias de un programa mediante los delimitadores de comentarios. Por ejemplo, se puede modificar el programa y anular la sentencia que imprime el mensaje, poniendo delante de ella el delimitador de comentarios en una sola línea.

```
//System.out.println("El primer programa");
```

Al correr el programa, observaremos que no imprime nada en la pantalla.

La sentencia *System.out.println()* imprime un mensaje. La función *println* tiene un sólo argumento una cadena de caracteres u objeto de la clase *String*

## Sentencias

Una sentencia es una orden que se le da al programa para realizar una tarea específica, esta puede ser: mostrar un mensaje en la pantalla, declarar una variable (para reservar espacio en memoria), inicializarla, llamar a una función, etc. Las sentencias acaban con **;** este carácter separa una sentencia de la siguiente. Normalmente, las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea. He aquí algunos ejemplos de sentencias

```
int i=1;  
import java.awt.*;  
System.out.println("El primer programa");  
rect.mover(10, 20);
```

## Bloques de código

Un bloque de código es un grupo de sentencias que se comportan como una unidad. Un bloque de código está limitado por las llaves de apertura **{** y cierre **}**. Como ejemplos de bloques de código tenemos la definición de una clase, la definición de una función miembro, una sentencia iterativa **for**, los bloques **try ... catch**, para el tratamiento de las excepciones, etc.

En el lenguaje Java, los caracteres espacio en blanco se pueden emplear libremente. Como podremos ver en los sucesivos ejemplos, es muy importante para la legibilidad de un programa la colocación de unas líneas debajo de otras empleando tabuladores. El editor del IDE nos ayudará plenamente en esta tarea sin apenas percibirlo.

Veamos la diferencia en la lectura de la siguiente función en dos versiones distintas, la primera que no hace uso de los espacios en blanco.

```
void ordenar(){  
int aux;  
for(int i=0; i<n-1; i++){
```

```

for(int j=i+1; j<n; j++){
    if(x[i]>x[j]){
        aux=x[j];
        x[j]=x[i];
        x[i]=aux;
    }
}
}
}

```

Se hace uso de los espacios en blanco, para delimitar bloques de código que se ejecutan dentro de otros bloques de código

```

void ordenar(){
    int aux;
    for(int i=0; i<n-1; i++){
        for(int j=i+1; j<n; j++){
            if(x[i]>x[j]){
                aux=x[j];
                x[j]=x[i];
                x[i]=aux;
            }
        }
    }
}

```

## Expresiones

Una expresión es todo aquello que se puede poner a la derecha del operador asignación =. Por ejemplo:

```

x=123;
y=(x+100)/4;
area=circulo.calcularArea(2.5);
Rectangulo r=new Rectangulo(10, 10, 200, 300);

```

- La primera expresión asigna un valor a la variable *x*.
- La segunda, realiza una operación
- La tercera, es una llamada a una función miembro *calcularArea* desde un objeto *circulo* de una clase determinada
- La cuarta, reserva espacio en memoria para un objeto de la clase *Rectangulo* mediante la llamada a una función especial denominada constructor.

## Variables

Una variable es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha variable. Hay varios tipos de variables que requieren distintas cantidades de memoria para guardar datos.

Todas las variables han de declararse antes de usarlas, la declaración consiste en una sentencia en la que figura el tipo de dato y el nombre que asignamos a la variable. Una vez declarada se le podrá asignar valores.

Java dispone de tres tipos de variables:

- de instancia
- de clase
- locales

Las variables de instancia o miembros de dato como veremos más adelante, se usan para guardar los atributos de un objeto particular.

Las variables de clase o miembros de dato estáticos son similares a las variables de instancia, con la excepción de que los valores que guardan son los mismos para todos los objetos de una determinada clase. En el siguiente ejemplo, *PI* es una variable de clase y *radio* es una variable de instancia. *PI* guarda el mismo valor para todos los objetos de la clase *Circulo*, pero el radio de cada círculo puede ser diferente

```
class Circulo{
    static final double PI=3.1416;
    double radio;
//...
}
```

Las variables locales se utilizan dentro de las funciones miembro o métodos. En el siguiente ejemplo *area* es una variable local a la función *calcularArea* en la que se guarda el valor del área de un objeto de la clase *Circulo*. Una variable local existe desde el momento de su definición hasta el final del bloque en el que se encuentra.

```
class Circulo{
//...
    double calcularArea(){
        double area=PI*radio*radio;
        return area;
    }
}
```

En el lenguaje Java, las variables locales se declaran en el momento en el que son necesarias. Es una buena costumbre inicializar las variables en el momento en el que son declaradas. Veamos algunos ejemplos de declaración de algunas variables

```
int x=0;
String nombre="Ángel";
double a=3.5, b=0.0, c=-2.4;
boolean bNuevo=true;
int[] datos={2, 4, 7, -3};
```

Delante del nombre de cada variable se ha de especificar el tipo de variable que hemos destacado en letra negrita. Las variables pueden ser

- Un tipo de dato primitivo
- El nombre de una clase
- Un array

El lenguaje Java utiliza el conjunto de caracteres Unicode, que incluye no solamente el conjunto ASCII sino también caracteres específicos de la mayoría de los alfabetos. Así, podemos declarar una variable que contenga la letra ñ

```
int año=1999;
```

Se ha de poner nombres significativos a las variables, generalmente formados por varias palabras combinadas, la primera empieza por minúscula, pero las que le siguen llevan la letra inicial en mayúsculas. Se debe evitar en todos los casos nombres de variables cortos como *xx*, *i*, etc.

```
double radioCirculo=3.2;
```

Las variables son uno de los elementos básicos de un programa y se deben

- Declarar
- Inicializar
- Usar

## Tipos de datos primitivos

Tipo	Descripcion
<b>boolean</b>	Tiene dos valores <b>true</b> o <b>false</b> .
<b>char</b>	Caracteres Unicode de 16 bits. Los caracteres alfa-numéricos son los mismos que los ASCII con el bit alto puesto a 0. El intervalo de valores va desde 0 hasta 65535 (valores de 16-bits sin signo).
<b>byte</b>	Tamaño 8 bits. El intervalo de valores va desde $-2^7$ hasta $2^7 - 1$ (-128 a 127)
<b>short</b>	Tamaño 16 bits. El intervalo de valores va desde $-2^{15}$ hasta $2^{15} - 1$ (-32768 a 32767)
<b>int</b>	Tamaño 32 bits. El intervalo de valores va desde $-2^{31}$ hasta $2^{31} - 1$ (-2147483648 a 2147483647)
<b>long</b>	Tamaño 64 bits. El intervalo de valores va desde $-2^{63}$ hasta $2^{63} - 1$ (-9223372036854775808 a 9223372036854775807)
<b>float</b>	Tamaño 32 bits. Números en coma flotante de simple precisión. Estándar IEEE 754-1985 (de 1.40239846e-45f a 3.40282347e+38f)
<b>double</b>	Tamaño 64 bits. Números en coma flotante de doble precisión. Estándar IEEE 754-1985. (de 4.94065645841246544e-324d a 1.7976931348623157e+308d.)

Los tipos básicos que utilizaremos en la mayor parte de los programas serán **boolean**, **int** y **double**.

## Caracteres

En Java los caracteres no están restringidos a los ASCII sino son Unicode. Un carácter está siempre rodeado de comillas simples como 'A', '9', 'ñ', etc. El tipo de dato **char** sirve para guardar estos caracteres.

Un tipo especial de carácter es la secuencia de escape, similares a las del lenguaje C/C++, que se utilizan para representar caracteres de control o caracteres que no se

imprimen. Una secuencia de escape está formada por la barra invertida (\) y un carácter. En la siguiente tabla se dan las secuencias de escape más utilizadas.

Carácter	Secuencia de escape
retorno de carro	\r
tabulador horizontal	\t
nueva línea	\n
barra invertida	\\

## Variables booleanas

En el lenguaje C/C++ el valor 0 se toma como falso y el 1 como verdadero. En el lenguaje Java existe el tipo de dato **boolean**. Una variable booleana solamente puede guardar uno de los dos posibles valores: true (verdadero) y false (falso).

```
boolean encontrado=false;
{...}
encontrado=true;
```

## Variables enteras

Una variable entera consiste en cualquier combinación de cifras precedidos por el signo más (opcional), para los positivos o el signo menos, para los negativos. Son ejemplos de números enteros:

12, -36, 0, 4687, -3598

Como ejemplos de declaración de variables enteras tenemos:

```
int numero=1205;
int x,y;
long m=30L;
```

**int** es la palabra reservada para declarar una variable entera. En el primer caso, el compilador reserva una porción de 32 bits de memoria en el que guarda el número 1205. Se accede a dicha porción de memoria mediante el nombre de la variable, *numero*. En el segundo caso, las porciones de memoria cuyos nombres son *x* e *y*, guardan cualquier valor entero si la variable es local o cero si la variable es de instancia o de clase. El uso de una variable local antes de ser convenientemente inicializada puede conducir a consecuencias desastrosas. Por tanto, declarar e inicializar una variable es una práctica aconsejable.

En la tercera línea 30 es un número de tipo **int** por defecto, le ponemos el sufijo **L** en mayúsculas o minúsculas para indicar que es de tipo **long**.

Existen como vemos en la tabla varios tipos de números enteros (**byte**, **short**, **int**, **long**) y también, existe una clase denominada *BigInteger* cuyos objetos pueden guardar un número entero arbitrariamente grande.

## Variables en coma flotante

Las variables del tipo **float** o **double** (coma flotante) se usan para guardar números en memoria que tienen parte entera y parte decimal.

```
double PI=3.14159;  
double g=9.7805, c=2.9979e8;
```

El primero es una aproximación del número real  $\pi$ , el segundo es la aceleración de la gravedad a nivel del mar, el tercero es la velocidad de la luz en m/s, que es la forma de escribir  $2.9979 \cdot 10^8$ . El carácter punto '.', separa la parte entera de la parte decimal, en vez del carácter coma ',' que usamos habitualmente en nuestro idioma.

Otras ejemplos son los siguientes

```
float a=12.5f;  
float b=7f;  
double c=7.0;  
double d=7d;
```

En la primera línea 12.5 lleva el sufijo **f**, ya que por defecto 12.5 es **double**. En la segunda línea 7 es un entero y por tanto 7f es un número de tipo **float**. Y así el resto de los ejemplos.

Conceptualmente, hay infinitos números de valores entre dos números reales. Ya que los valores de las variables se guardan en un número prefijado de bits, algunos valores no se pueden representar de forma precisa en memoria. Por tanto, los valores de las variables en coma flotante en un ordenador solamente se aproximan a los verdaderos números reales en matemáticas. La aproximación es tanto mejor, cuanto mayor sea el tamaño de la memoria que reservamos para guardarlo. De este hecho, surgen las variables del tipo **float** y **double**. Para números de precisión arbitraria se emplea la clase *BigDecimal*.

## Valores constantes

Cuando se declara una variable de tipo **final**, se ha de inicializar y cualquier intento de modificarla en el curso de la ejecución del programa da lugar a un error en tiempo de compilación.

Normalmente, las constantes de un programa se suelen poner en letras mayúsculas, para distinguirlas de las que no son constantes. He aquí ejemplos de declaración de constantes.

```
final double PI=3.141592653589793;
final int MAX_DATOS=150;
```

## Las cadenas de caracteres o strings

Además de los ocho tipos de datos primitivos, las variables en Java pueden ser declaradas para guardar una instancia de una clase, como veremos en el siguiente capítulo (Clases y objetos).

Las cadenas de caracteres o strings son distintas en Java y en el lenguaje C/C++. En este último, las cadenas son arrays de caracteres terminados en el carácter '\0'. Sin embargo, en Java son objetos de la clase *String*.

```
String mensaje="El primer programa";
```

```
public class PrimeroApp{
    public static void main(String[] args) {
//imprime un mensaje
        String mensaje="El primer programa";
        System.out.println(mensaje);
    }
}
```

En una cadena se pueden insertar caracteres especiales como el carácter tabulador '\t' o el de nueva línea '\n'

```
String texto="Un string con \t un carácter tabulador y \n un
salto de línea";
```

## Palabras reservadas

En el siguiente cuadro se listan las palabras reservadas, aquellas que emplea el lenguaje Java y que el programador no puede utilizar como identificadores. Algunas de estas palabras le resultarán familiares al programador del lenguaje C/C++. Las palabras reservadas señaladas con un asterisco (\*) no se utilizan.

abstract	boolean	break	byte	byvalue*
case	cast*	catch	char	class
const*	continue	default	do	double
else	extends	false	final	finally
float	for	future*	generic*	goto*

if	implements	import	inner*	instanceof
int	interface	long	native	new
null	operator*	outer*	package	private
protected	public	rest*	return	short
static	super	switch	synchronized	this
throw	transient	true	try	var*
void	volatile	while		

Las palabras reservadas se pueden clasificar en las siguientes categorías:

- Tipos de datos: **boolean, float, double, int, char**
- Sentencias condicionales: **if, else, switch**
- Sentencias iterativas: **for, do, while, continue**
- Tratamiento de las excepciones: **try, catch, finally, throw**
- Estructura de datos: **class, interface, implements, extends**
- Modificadores y control de acceso: **public, private, protected, transient**
- Otras: **super, null, this.**