Variables

Una variable es una representación, mediante un identificador, de un valor que puede cambiar durante la ejecución de un programa.

Identificador: el nombre con el que se identifica cada variable. Importante recordar que Java es un lenguaje *case-sensitive* (distingue entre mayúsculas y minúsculas), por lo que *edad, Edad* y *EDAD* son identificadores que identifican variables distintas.

Regla de estilo: toda variable se escribirá en formato *camelCase*, con la primera letra minúscula.

Tipos de datos simples:

Tipo de datos	Información representada	Rango	Tamaño
`byte`	Entero corto	De –128 a 127	8 bits
`short`	Entero	De –32 768 a 32 767	16 bits
`int`	**Entero**	**-2 147 483 648 a 2 147 483 647**	**32 bits**
`long`	Entero largo	±9 223 372 036 854 775 808	64 bits
`float`	Reales precisión sencilla	...	32 bits
`double`	Reales precisión doble	...	64 bits
`boolean`	Lógico	...	1 bit
`char`	Carácter	...	16 bits

Constantes: son un caso especial de variables, para las que una vez asignado el valor, éste permanece inmutable. Para declarar una constante utilizamos la palabra reservada `final`. Por convenio, las constantes se escriben en mayúscula: `final tipo NOMBRE_CONSTANTE;`.

Comentarios

La funcionalidad de los comentarios es describir la funcionalidad del código y facilitar la comprensión de la solución implementada. Se considera una **buena práctica** escribir códigos bien documentados.

Varios tipos de comentarios:

- Comentario multilínea: Empieza con los caracteres `/*` y termina con `*/`.
- Comentario hasta final de línea: Todo lo que sigue a los caracteres `//`.
- Javadoc: similar al multilínea, pero utilizando `/**` y `**/` con una notación específica para generar documentación.

API de Java

Java, al igual que cualquier otro lenguaje de programación, tiene a su disposición una enorme cantidad de bibliotecas de clases que podemos utilizar capaces de realizar tareas complejas de forma transparente. A toda esta biblioteca se le denomina API (*Application Programming Interface*).

Para facilitar la ordenación de todas estas clases se agrupan en paquetes y subpaquetes. Por ejemplo, la clase *Math* se encuentra en el paquete *lang*, que está dentro del paquete *java*. Para utilizar cualquier clase y método de la API podemos importarla, para no tener que escribir continuamente el nombre completo del paquete: `import java.lang.Math`.

Salida por consola: es una de las operaciones más básicas que proporciona la API de Java: `System.out.println("Hola mundo");`. Debido a lo mucho que se usa, en muchos IDE se utiliza la notación simplificada *sout* y después un tabulador (en NetBeans), o *syso* y después Ctrl+Enter (en Eclipse) para que se escriba de manera automática esta sentencia. Si utilizamos `print()`, el carácter especial `\n` representará el salto de carro.

Entrada de datos: Scanner es una clase de la API para leer datos.

```
Scanner sc = new Scanner(System.in);
```

System.in indica que leeremos de teclado. Para utilizar nuestra variable sc utilizaremos las sentencias adecuadas: `sc.nextInt()`, `sc.nextLine()`, etcétera.

Cuando utilizamos una de las sentencias de lectura, la ejecución del programa se detiene hasta que la persona usuaria introduzca datos.

Operadores

El operador de asignación es `=`.

Operadores aritméticos:

Operador	**Descripción**	**Uso**
+	Suma	a + b
+	Más unario (número positivo)	+a
-	Menos unario (número negativo)	-a
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
%	Módulo (resta de una división entera)	a % b
++	Incremento en 1	a++ o ++a
--	Decremento en 1	a-- o --a

Operadores relacionales:

Operador	**Descripción**	**Uso**
==	Igual que	a == b
!=	Distinto que	a != b

<	Menor que	a < b
<=	Menor o igual que	a <= b
>	Mayor que	a > b
>=	Mayor o igual que	a >= b

Operadores lógicos:

Operador	**Descripción**	**Uso**
&&	Y	a && b
||	O	a || b
!	No	!a

Operadores de asignación:

Operador	**Descripción**	**Uso**
=	Asignación	a = b
+=	Suma y asigna. Equivalente: a = a + b	a += b
-=	Resta y asignación	a < b
*=	Multiplicación y asignación	a <= b
/=	División y asignación	a > b
%=	Módulo y asignación	a >= b

Operador ternario:

```
expresionCondicional ? ValorSiVerdad : valorSiFalso
```

Por ejemplo: a = 3 < 5 ? 1 : -1. Al final a contiene el valor 1.

A la hora de utilizar los operadores debemos tener en cuenta su **precedencia**. Podemos utilizar **paréntesis** para alterarla.

Conversiones de tipos (cast):

Ya veremos con más detenimiento los *casting* más adelante. Pero por ahora diferenciaremos entre dos tipos de conversiones:

- Conversiones implícitas: Se realiza de forma automática entre dos tipos diferentes de datos. Requiere que la variable destino tenga más precisión que la variable origen:
  int a = 3;
  double x = a;

- Conversiones explícitas: al perder precisión, se debe forzar la conversión mediante una operación llamada *cast*:
  double x = 3.6;
  int a = (int) x;