Java para programadores

Java y Servicios Web I
Master en Ingeniería Matemática

Manuel Montenegro
Dpto. Sistemas Informáticos y Computación

Desp. 467 (Mat)

montenegro@fdi.ucm.es

Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Variables

- Toda variable ha de ser declarada con su tipo.
- La declaración de una variable ha de hacerse antes de su primer uso.

Tipo

Nombre de variable

```
int a;
a = 5;
int b;
b = 3;
System.out.print("a vale: ");
System.out.println(a);
```

Variables

- Toda variable ha de ser declarada con su tipo.
- La declaración de una variable ha de hacerse antes de su primer uso.

```
int a, b;
a = 5;
b = 3;
System.out.print("a vale: ");
System.out.println(a);
```

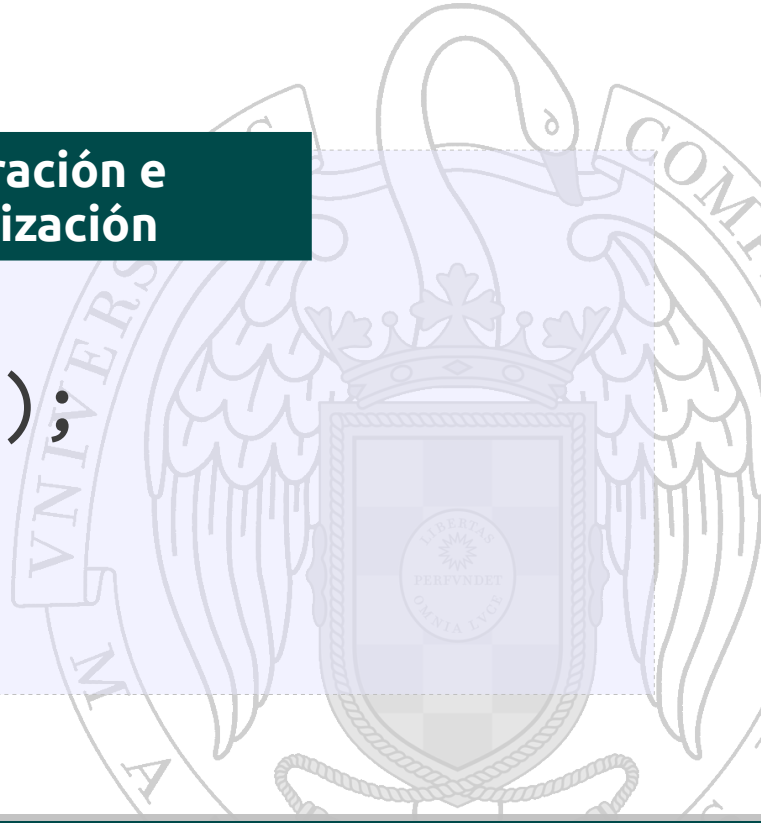
Declaración de varias variables

Variables

- Toda variable ha de ser declarada con su tipo.
- La declaración de una variable ha de hacerse antes de su primer uso.

```
int a = 5, b = 3;

System.out.print("a vale: ");
System.out.println(a);
```

Declaración e inicialización

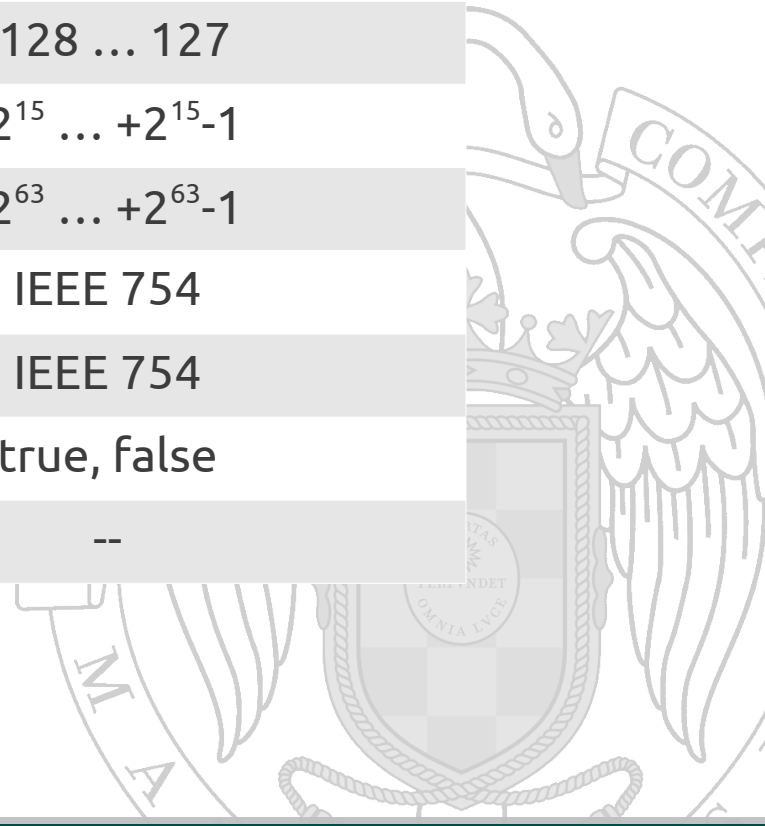


Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Tipos de datos primitivos

Tipo	Tamaño	Rango
`int`	4 bytes	-2^{31} … $+2^{31}-1$
`char`	2 bytes	Caracteres Unicode
`byte`	1 byte	-128 … 127
`short`	2 bytes	-2^{15} … $+2^{15}-1$
`long`	8 bytes	-2^{63} … $+2^{63}-1$
`float`	4 bytes	IEEE 754
`double`	8 bytes	IEEE 754
`boolean`	--	true, false
`void`	--	--



Literales

- Literales de tipo entero:

Hex.

Octal

```
0     157     -23     0x2b5     023
```

Literales

- Literales de tipo entero:

```
0     157     -23     0x2b5     023
```

Hex.

Octal

- Literales de tipo carácter:

```
'a'    'V'    '&'    '\n'  '\u0041'  '\\'
```

Salto de línea

0x41 = 65 = 'A'

Literales

- Literales de tipo entero:

```
0     157     -23     0x2b5     023
```

Hex.

Octal

- Literales de tipo carácter:

```
'a'    'V'    '&'    '\n' '\u0041' '\\'
```

Salto de línea

0x41 = 65 = 'A'

- Literales de tipo `float` y `double`:

```
0.234f  4.30     3.54E+23
```

$3{,}54 \times 10^{23}$

Literales

- Literales de tipo entero:

```
0     157     -23     0x2b5     023
```

Hex. 0x2b5

Octal 023

- Literales de tipo carácter:

```
'a'    'V'    '&'    '\n' '\u0041' '\\'
```

Salto de línea '\n'

0x41 = 65 = 'A' '\u0041'

- Literales de tipo `float` y `double`:

```
0.234f   4.30     3.54E+23
```

$3{,}54 \times 10^{23}$

- Literales de tipo `boolean`: `true`, `false`

Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

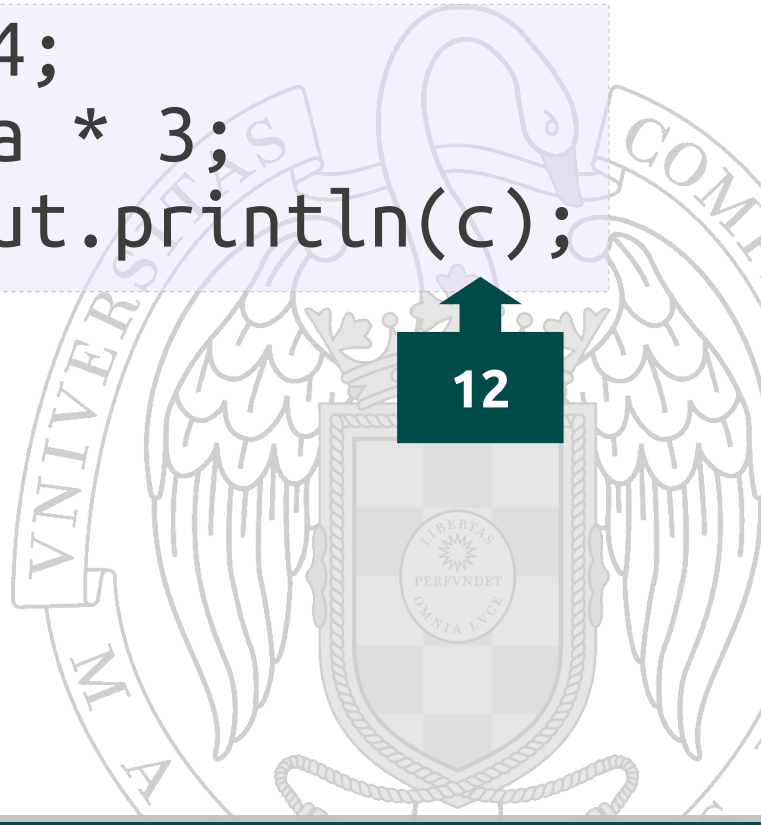
Operadores aritméticos

- Devuelven un número como resultado

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (resto)

```
int a = 4;
int c = a * 3;
System.out.println(c);
```

12



Operadores aritméticos

- La división devuelve un número entero si sus dos operandos son de tipo `int`

```
int a = 4;
float c = a / 3;
```

c = 1.0

```
int a = 4;
float c = a / 3f;
```

c = 1.333...

```
int a = 4;
int b = 3;
float c = a / (float) b;
```

c = 1.333...

Conversión

Conversión de tipos

- Para asignar un valor a una variable de un tipo más restrictivo, hay que hacer una conversión explícita.

```
int a = 3;
float b = 4.21f;
b = a;              // Correcto
a = b;              // Error
a = (int) b;        // Correcto
```

Conversión implícita

Conversión explícita

Operadores relacionales

- Devuelven un booleano como resultado

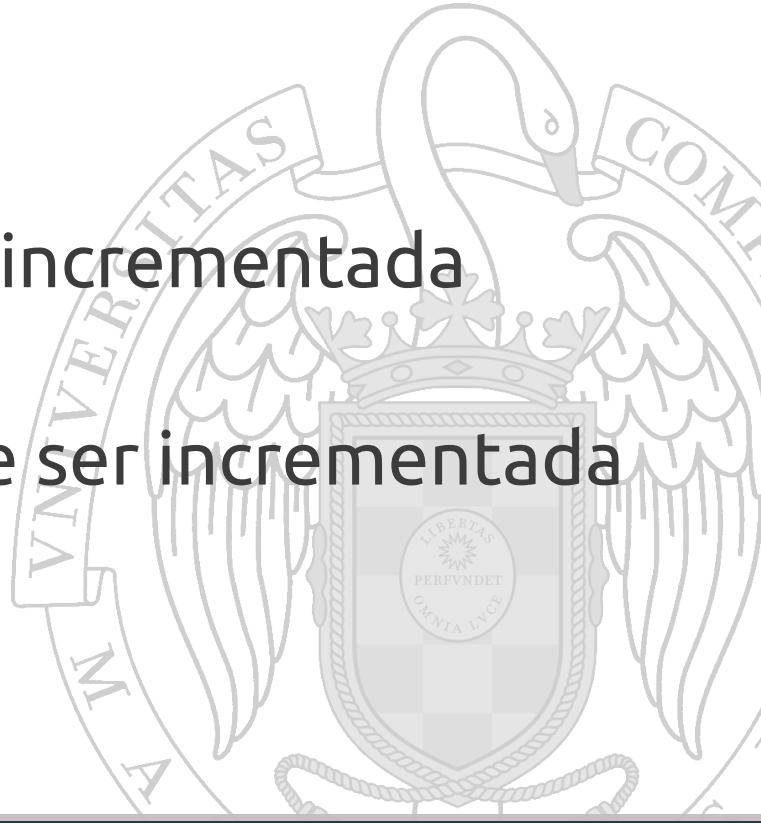
Operador	Significado
==	Igual que
!=	Distinto de
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

```
int a = 4;
int b = 3;

if (a > b)
   System.out.println("a es mayor");
else if (a == b)
   System.out.println("a es igual");
else
   System.out.println("a es menor");
```

Incremento y decremento

- Sintaxis: `x++` `x--`
- Equivalen a: `x=x+1` `x=x-1`

- Dos variantes:
  - Preincremento (`++x`)
    Devuelven el valor de x tras ser incrementada
  - Postincremento (`x++`)
    Devuelven el valor de x antes de ser incrementada



Operador y asignación

```
a += 2
```

```
a = a + 2
```

```
a *= c
```

```
a = a * c
```

Operadores lógicos

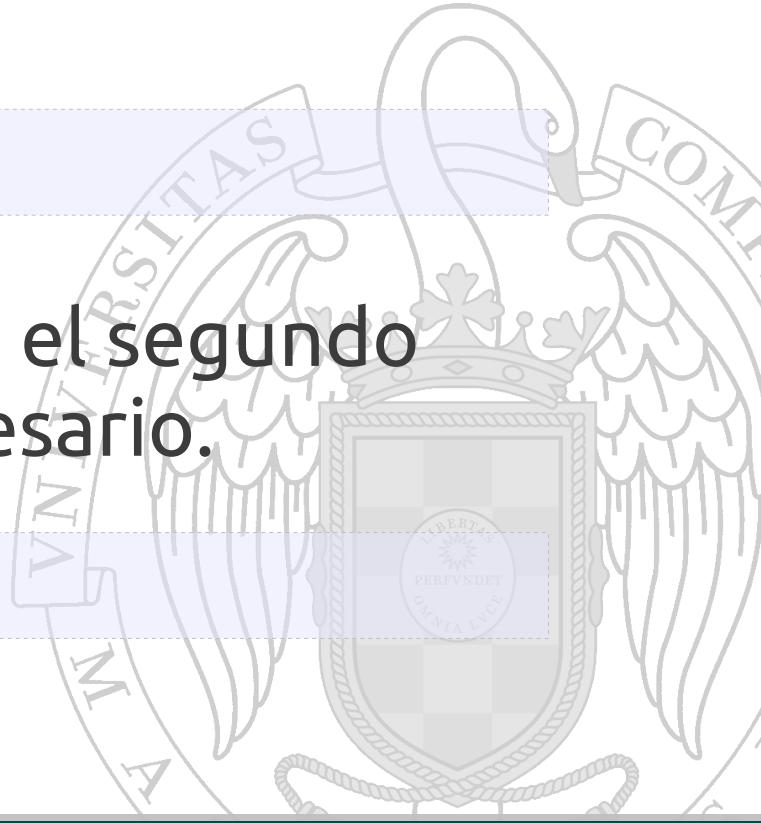
- Operan con booleanos.

Operador	Significado
&&	And
||	Or
!	Not

```
if ((x >= 10) && (x <= 20))
```

- Cortocircuito: && y || no evalúan el segundo operando, a menos que sea necesario.

```
if ((x != 0) && (1/x > 3))
```



Contenidos

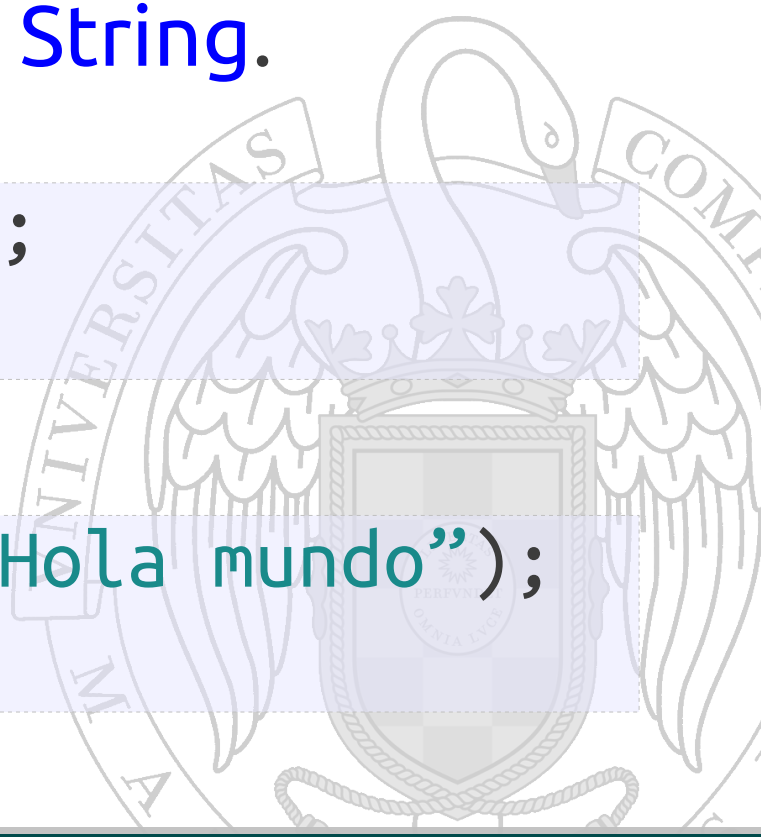
- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Cadenas

- Una cadena es una secuencia de caracteres.
- Los literales de tipo cadena se delimitan entre comillas dobles (“).
- El tipo de datos de cadenas es String.

```
String texto = “Hola mundo”;
System.out.println(texto);
```

```
String texto = new String(“Hola mundo”);
System.out.println(texto);
```



Operaciones con cadenas

- Concatenación de cadenas (+):

```
String texto1 = “Hola ”;
String texto2 = “ mundo”;
String texto3 = texto1 + texto2;
```

“Hola mundo”

- Obtener la longitud de una cadena:

```
String texto = “Cadena de prueba”;
System.out.println(texto.length());
```

16

- Conversión a tipos de datos numéricos:

```
String numero = “23”;
int a = Integer.parseInt(numero);
double b = Double.parseDouble(numero);
```

Operaciones con cadenas

- Conversión desde tipos de datos numéricos:

```
int a = 23;
float b = 2.34f;
String cadena = String.valueOf(a);
String cadena2 = String.valueOf(b);
```

"23"

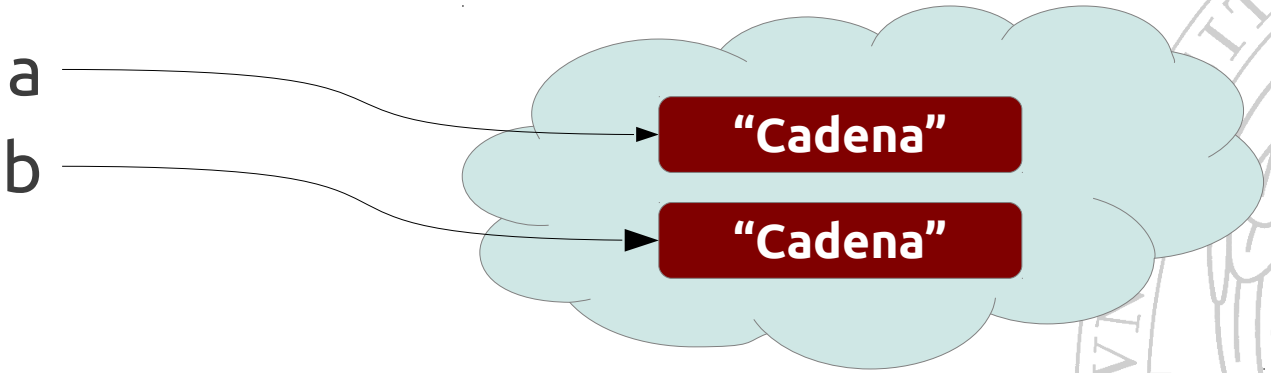
"2.34"

Operaciones con cadenas

¡Ojo con la igualdad entre cadenas!

```
String a = “Cadena”;
String b = “Cadena”;
if (a == b)
  System.out.println(“Las cadenas son iguales”);
```

false

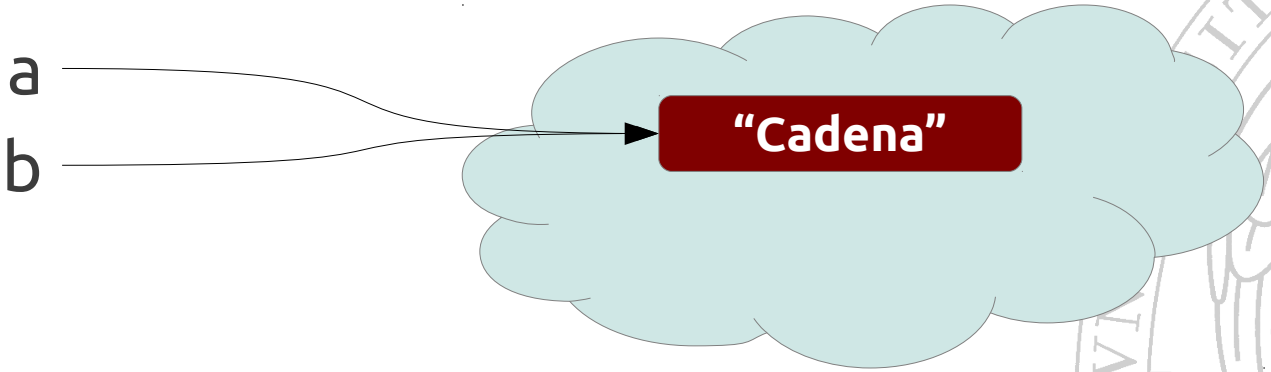


Operaciones con cadenas

¡Ojo con la igualdad entre cadenas!

```
String a = “Cadena”;
String b = a;
if (a == b)
  System.out.println(“Las cadenas son iguales”);
```

true

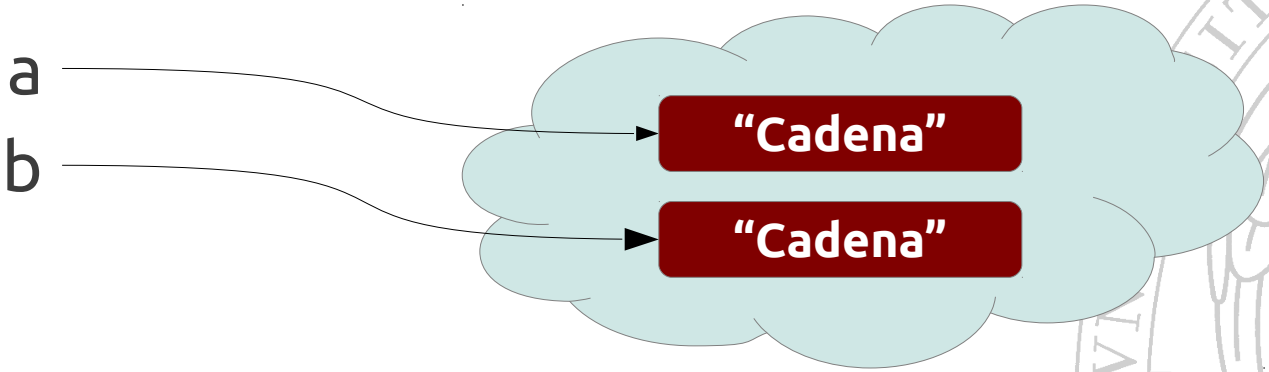


Operaciones con cadenas

¡Ojo con la igualdad entre cadenas!

```
String a = “Cadena”;
String b = “Cadena”;
if (a.equals(b))
  System.out.println(“Las cadenas son iguales”);
```

true



Igualdad entre cadenas

- Las cadenas son objetos que se almacenan en el *heap*.
- Las variables de tipo cadena son referencias a estos objetos.
- El operador == comprueba si dos variables hacen referencia al mismo objeto.
- Esto se aplica:
  - a los tipos compuestos: objetos, `String`, arrays.
  - pero NO a los tipos básicos: `int`, `char`, `float`, ...

Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Sentencias de control

- Sentencia `if`
- Sentencia `switch`
- Sentencia `while`
- Sentencia `do - while`
- Sentencia `for`

Sentencia `if`

```
if (condición) {
    sentencias
}
```

```
if (condición) {
    sentencias
} else {
    sentencias
}
```

- Si sólo hay una sentencia, pueden omitirse las llaves { }

```
if (condición)
    sentencia;
else
    sentencia;
```

Sentencia if

```
Scanner sc = new Scanner(System.in);
System.out.print("Dime tu edad: ");
int edad = sc.nextInt();
boolean admitido;
if (edad <= 18) {
    System.out.println("Eres demasiado joven");
    admitido = false;
} else {
    System.out.println("De acuerdo, pase");
    admitido = true;
}
```

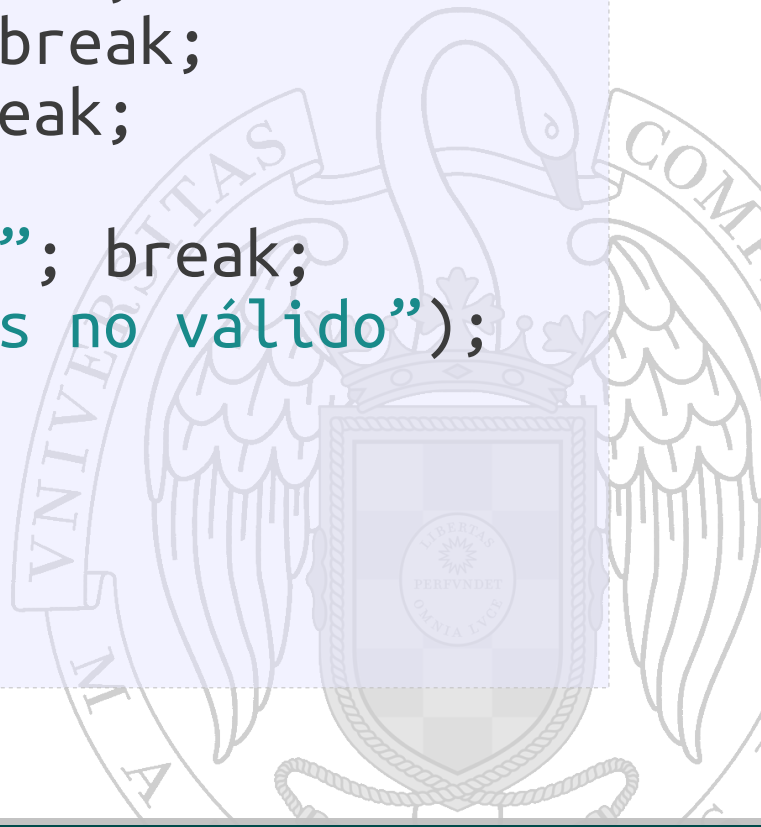
Sentencia switch

```
switch(expresion) {
   case valor1:
      sentencias
      break;

   case valor2:
      sentencias
      break;


   …

   default:
      sentencias
}
```

- Solución elegante alternativa a las cadenas
  `if → else if → else if...`

Sentencia switch

```
int mes = new Scanner(System.in).nextInt();
String nombreMes;

switch (mes) {

   case 1: nombreMes = “Enero”; break;
   case 2: nombreMes = “Febrero”; break;
   case 3: nombreMes = “Marzo”; break;
   …
   case 12: nombreMes = “Diciembre”; break;
   default: System.out.println(“Mes no válido”);
}
```



Sentencia switch

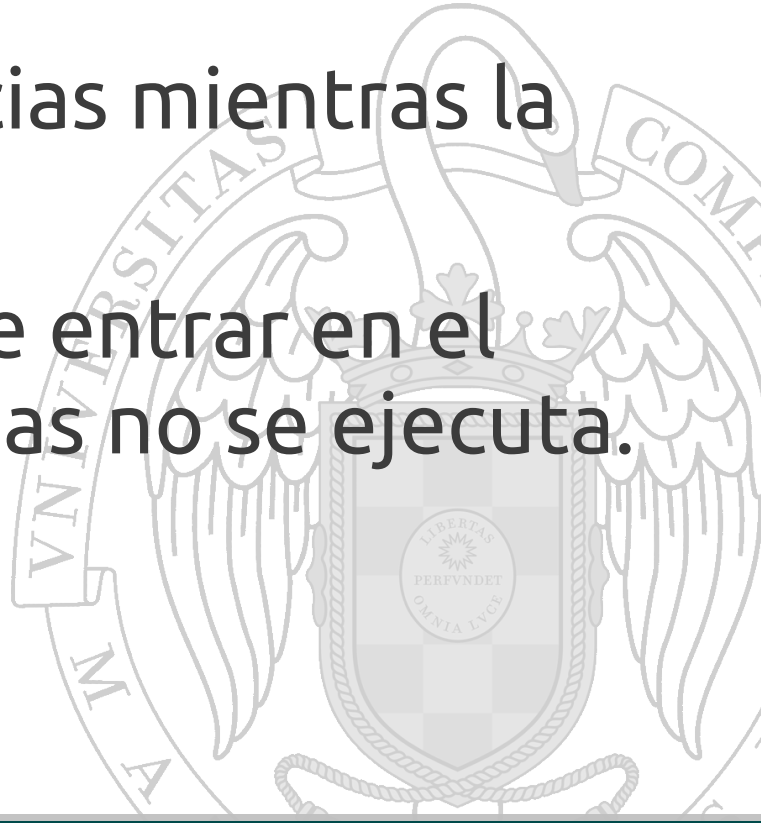
```
int mes = new Scanner(System.in).nextInt();
int numeroDias;

switch (mes) {

   case 2:
      numeroDias = 28; break;
   case 4:
   case 6:
   case 9:
   case 11:
      numeroDias = 30; break;
   default:
      numeroDias = 31;
}
```

Si no hay sentencia break, la ejecución continúa por la rama siguiente

Sentencia `while`

```
while (condición) {
    sentencias
}
```

- Repite la secuencia de sentencias mientras la condición sea cierta.
- Si la condición es falsa antes de entrar en el bucle, la secuencia de sentencias no se ejecuta.



Sentencia while

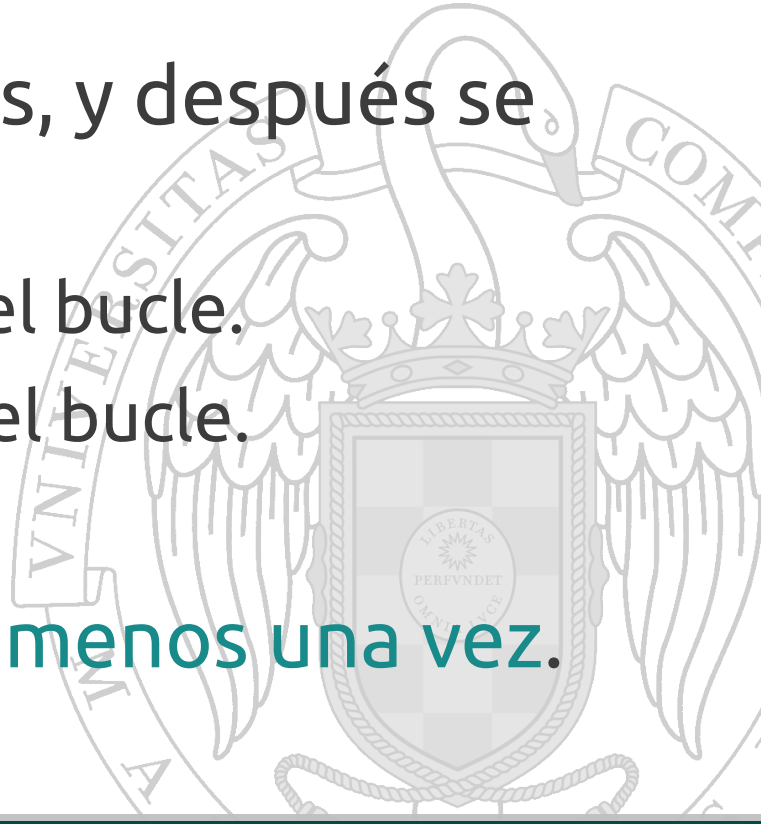
```
int n = 20;

while (n >= 0) {
    System.out.print("n = ");
    System.out.println(n);
    n--;
}
```

Sentencia do-while

```
do {
    sentencias
} while (condición)
```

- Ejecuta el conjunto de sentencias, y después se evalúa la condición.
  - Si la condición es cierta, se repite el bucle.
  - Si la condición es falsa, abandona el bucle.

- El cuerpo del bucle se ejecuta al menos una vez.



Sentencia do-while

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

do {
    System.out.print("Díme un número entre 1 y 10: ");
    numero = sc.nextInt();
} while ((numero < 1) || (numero > 10))
```

Sentencia for

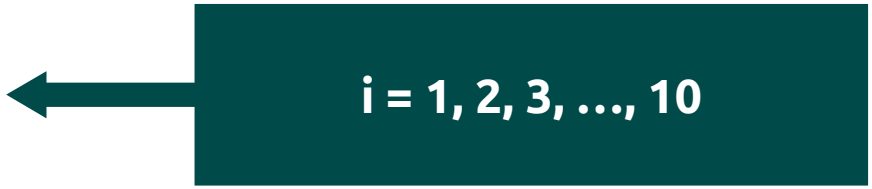
```
for (inicialización; condición; incremento) {
   sentencias
}
```

- Ejecuta la sentencia *inicialización*. Repite el conjunto de *sentencias* mientras la *condición* sea cierta. Al finalizar cada iteración del bucle, ejecuta la sentencia de *incremento*.
- Equivale al siguiente bucle while:

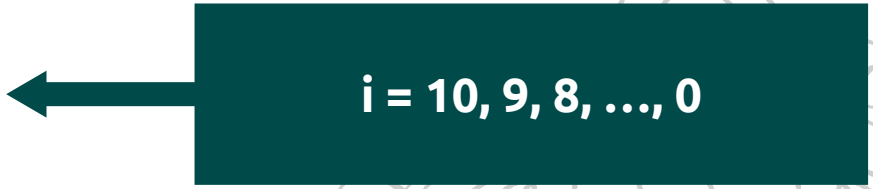
```
inicialización;
while (condición) {
   sentencias
   incremento;
}
```

Sentencia for

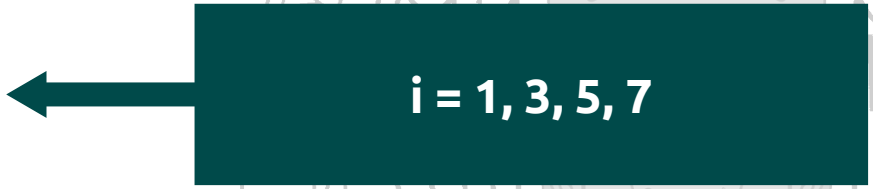
```
int i;
for (i = 1; i <= 10; i++) {
   System.out.println(i);
}
```




```
int i;
for (i = 10; i >= 0; i--) {
   System.out.println(i);
}
```




```
int i;
for (i = 1; i < 9; i += 2) {
   System.out.println(i);
}
```



Sentencia for

- Es posible declarar el tipo de la variable en la misma sentencia de inicialización.
- En este caso, la variable deja de estar definida fuera del bucle.

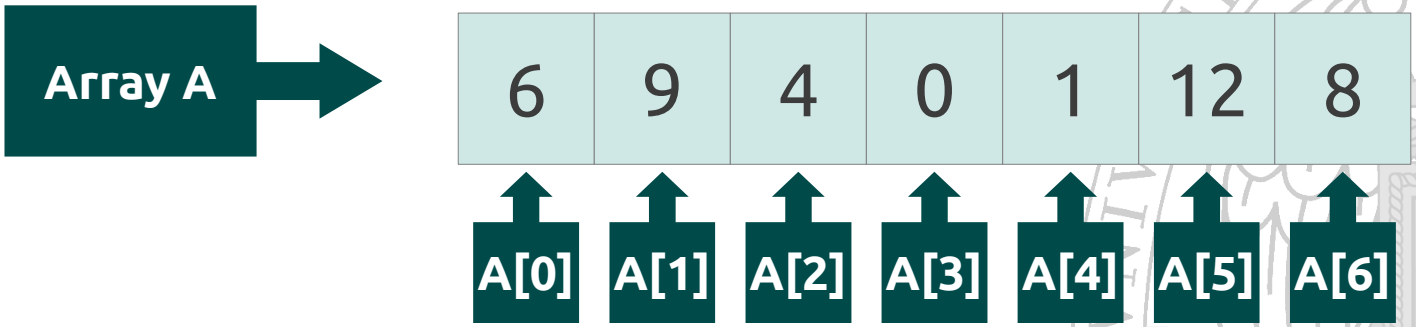
```
for (int i = 1; i <= 10; i++) {
   System.out.println(i);
}


System.out.println(i); // Error
```

Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Arrays

- Un array es una lista de elementos del mismo tipo.
- Cada elemento está identificado por su posición dentro del array (índice).



Arrays

- Los arrays se declaran como una variable normal. Hay que añadir corchetes [] al tipo de los elementos del array.

```
int[] edades;
String[] nombres;
```

- Una vez declarados, han de inicializarse indicando el número de elementos que tendrán.

```
edades = new int[10];
nombres = new String[15];
```

Arrays

- Los arrays también pueden ser inicializados enumerando sus elementos.

```
edades = {23, 24, 27, 19, 18, 20};
nombres = {“Pepe”, “Luis”, “María”};
```

- El acceso a cualquier elemento de un array se realiza mediante el índice del mismo entre corchetes.

```
System.out.println(edades[1]);
System.out.println(nombres[3]);
```

24

Error

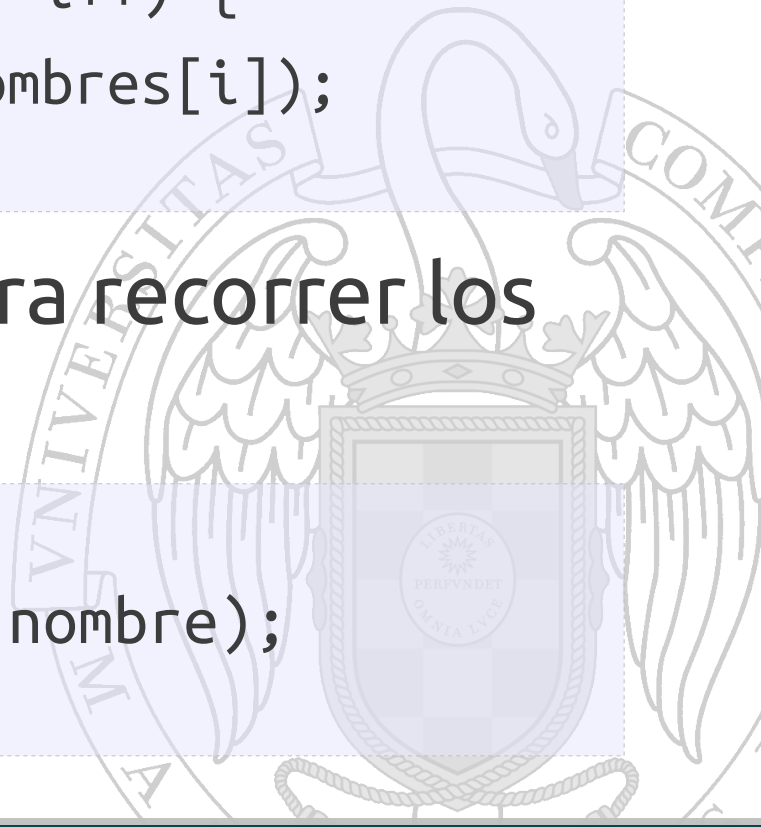
Arrays

- Una vez creado, se puede obtener la longitud de un array, mediante el atributo `length`.

```
nombres = {“Pepe”, “Luis”, “María”};
for (int i = 0; i < nombres.length; i++) {
   System.out.println(“Hola, ” + nombres[i]);
}
```

- Existe una sintaxis especial para recorrer los elementos de un array.

```
for (String nombre : nombres) {
   System.out.println(“Hola, ” + nombre);
}
```



Arrays

- Los arrays se crean en el heap. Por tanto, se le aplican las mismas consideraciones en relación al operador de igualdad (==)

```
int[] a = {1, 3, 4};
int[] b = {1, 3, 4};
int[] c = a;
```

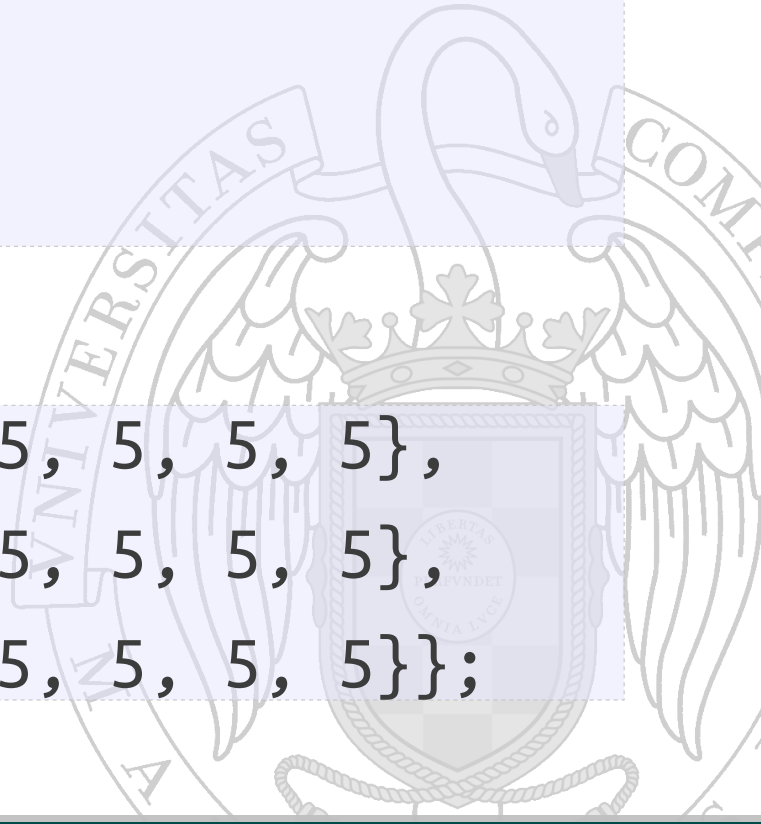
```
a == b   false
b == c   false
a == c   true
```

Arrays bidimensionales

- Se manejan de modo similar:

```
int[][] a = new int[3][9];
for (int i = 0; i < a.length; i++) {
  for (int j = 0; j < a[i].length; j++) {
    a[i][j] = 5;
  }
}
```

```
int[][] a = { {5, 5, 5, 5, 5, 5, 5, 5, 5},
              {5, 5, 5, 5, 5, 5, 5, 5, 5},
              {5, 5, 5, 5, 5, 5, 5, 5, 5}};
```

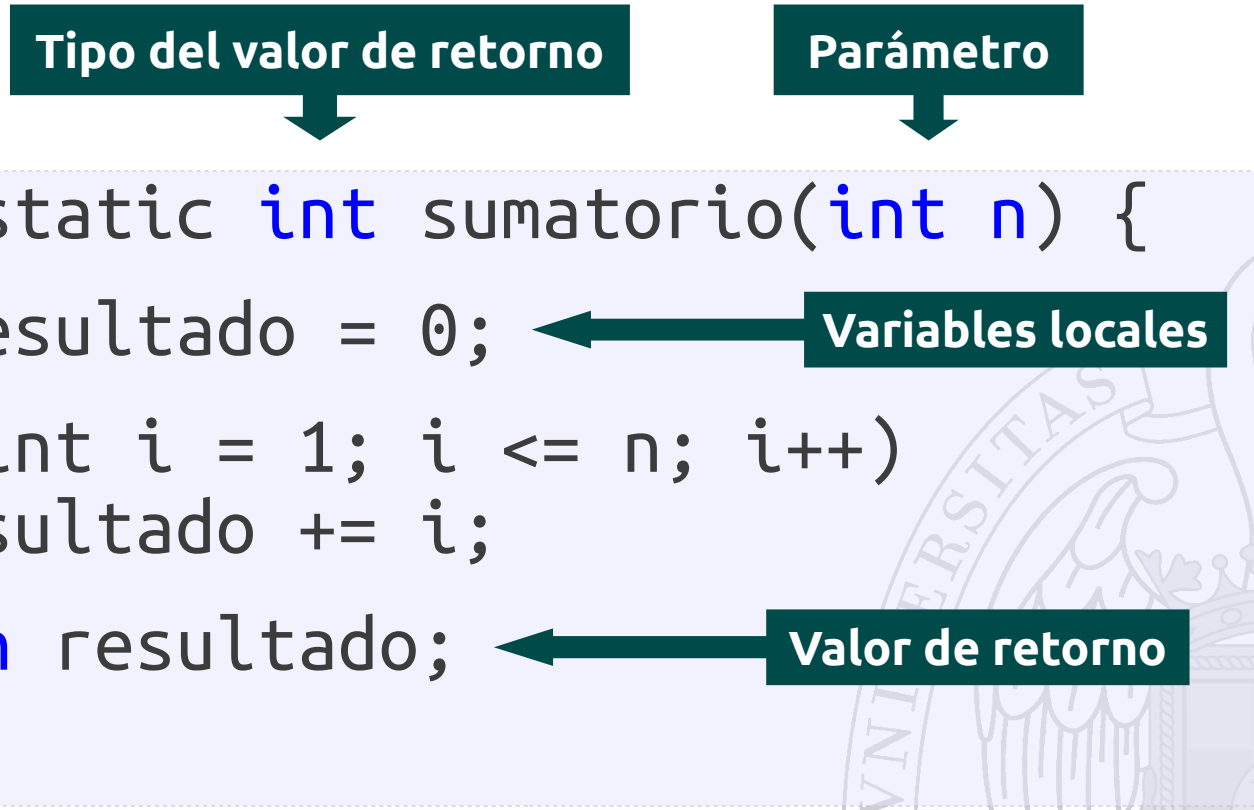


Contenidos

- Variables.
- Tipos de datos primitivos y literales.
- Operadores.
- Cadenas.
- Sentencias de control.
- Arrays.
- Procedimientos y funciones.

Procedimientos y funciones

- Han de ser declarados dentro de una clase.



Procedimientos y funciones

- Los procedimientos se declaran con tipo `void`.

```
public static void listaPersonas(String[] nombres,
                                 int[] edades) {

  for (int i = 0; i <= nombres.length; i++) {
     System.out.print("Nombre: " + nombres[i]);
     System.out.print("Edad: ");
     System.out.println(edades[i]);

  }
}
```

```
String[] nombres = {"Fran", "Estela"};
String[] edades = {34, 33};
listaPersonas(nombres, edades);
```

Referencias

- P. Deitel, H. Deitel
  Java. How to Program (9th Edition)
  Caps. 2, 4, 5.
- B. Eckel
  Thinking in Java (3rd Edition)
  Cap. 3.