



# Unió Europea

Fons Social Europeu

*L'FSE inverteix en el teu futur*



GENERALITAT  
VALENCIANA



## UF04. - INTRODUCCIÓ A JAVA

*- Teoria -*

PROGRAMACIÓ  
CFGS DAM

Autor:

José Manuel Martí Fenollosa

Revisat per:

Àngel Olmos Giner

[a.olmosginer@edu.gva.es](mailto:a.olmosginer@edu.gva.es)

2022/2023

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA

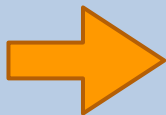


1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

# 1. INTRODUCCIÓ

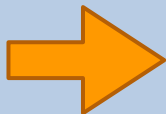
## INTRODUCCIÓ

### Què és Java?



Java és un **llenguatge de programació** de propòsit **general** (vàlid per a realitzar tot tipus d'aplicacions professionals) i **orientat a objectes** que va ser dissenyat (en 1995 per James Gosling) específicament per a tindre tan poques dependències d'implementació com fora possible

### El seu objectiu

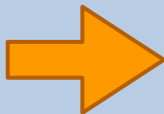


**WORA ("Write Onze, Run Anywhere")**: la qual cosa vol dir que el codi pot escriure's una sola vegada i ser executat en qualsevol mena de dispositius (PC, mòbil, etc.)

# 1. INTRODUCCIÓ

## INTRODUCCIÓ

### Característiques



**Senzill:** llenguatge senzill d'aprendre

**Orientat a Objectes:** a excepció dels tipus fonamentals de variables (int, char, long...), tot és un objecte

**Robust:** El compilador Java detecta molts més errors que altres compiladors

**Portable:** totes les implementacions de Java segueixen els mateixos estàndards quant a grandària i emmagatzematge de les dades

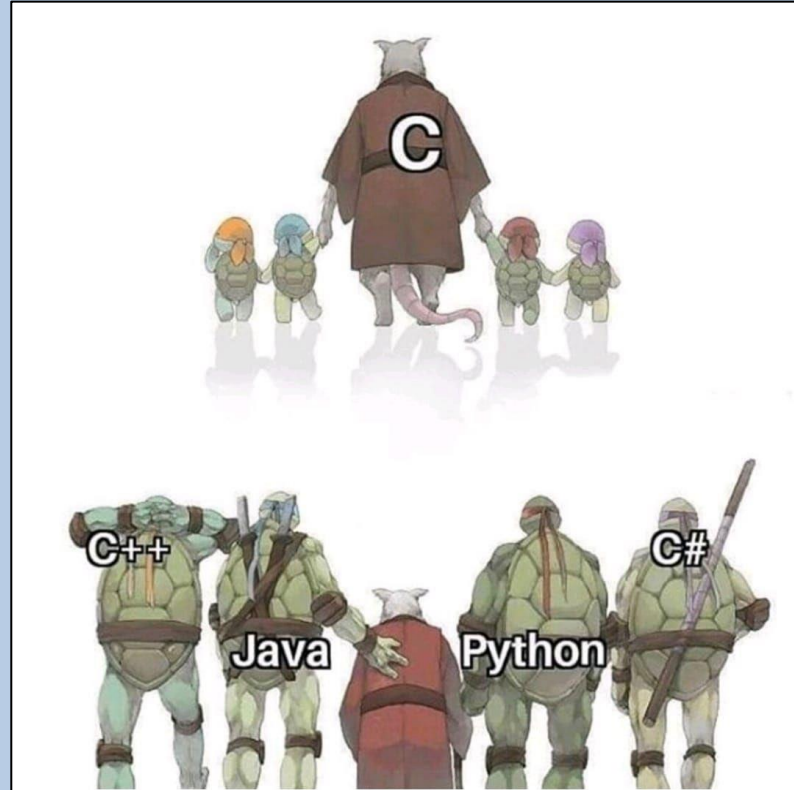
**Arquitectura Neutral:** El codi generat pel compilador Java és pot executar en un entorn UNIX (Linux, Mac), Windows, Mòbil, etc.

**Rendiment Mitjà:** velocitat de processament del codi adequat

**Multithread:** Suporta de manera nativa els threads (fils d'execució), sense necessitat de l'ús de de llibreries específiques

# 1. INTRODUCCIÓ

## INTRODUCCIÓ



# ABANS DE SEGUIR INSTAL·LAR NETBEANS



Instal·la't

## JAVA 17 (JRE + JDK<sup>(2)</sup>) + l'IDE NetBeans 15<sup>(1)</sup>

- (1) Definició de IDE (Integrated Development Environment -- Entorn de Desenvolupament Integrat) <https://www.unir.net/ingenieria/revista/ide-programacion/>
- (2) Diferencias entre JDK, JRE y JVM: <https://javadesdecero.es/fundamentos/diferencias-jdk-jre-jvm/>

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA



1. Introducció
2. **Primer exemple**
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

## 2. PRIMER EXEMPLE

### PRIMER EXEMPLE



GENERALITAT  
VALENCIANA



Crea un mètode (funció) en Java que ens retorne el missatge “**Hola Mundo!**” per pantalla:

```
[acceso] [modificador] tipo nombreMétodo([tipo nombreArgumento, [tipo nombreArgumento]...])  
{  
    /*  
    * Bloque de instrucciones  
    */  
    [return valor;]  
}
```

```
12 public class HolaMundo {  
13  
14     public static void main(String[] args) {  
15         // TODO code application logic here  
16         System.out.println("Hola Mundo!");  
17     }  
18  
19 }
```




## 2. PRIMER EXEMPLE

### PRIMER EXEMPLE

Aquesta línia declara la classe **HolaMundo**. El nom de la classe especificat en el fitxer font s'utilitza per a crear un fitxer ***nombredeclase.class*** en el directori en el qual es compila l'aplicació. En aquest cas, el compilador crearà un fitxer anomenat ***HolaMundo.class***.

#### CLASS

*HolaMundo.class*



```
12 public class HolaMundo {
13
14     public static void main(String[] args) {
15         // TODO code application logic here
16         System.out.println("Hola Mundo!");
17     }
18
19 }
```

## 2. PRIMER EXEMPLE

### PRIMER EXEMPLE

Aquesta línia especifica un **Mètode** (funció) que l'interpret Java busca per a executar en primer lloc. Java utilitza la **paraula clau main** per a especificar **la primera funció a executar**

- **public:** significa que el mètode *main()* pot ser cridat per qualsevol, incloent l'interpret Java
- **static:** li diu al compilador que *main* es refereix a la pròpia classe HolaMundo i no a cap instància de la classe. D'aquesta manera, si algú intenta fer una altra instància de la classe, el mètode *main()* no s'instanciarà
- **void** indica que *main()* no retorna res. Això és important ja que Java realitza una estricta comprovació de tipus
- **String [] args** és la declaració d'un array de Strings. Aquests són els arguments escrits després del nom de la classe en la línia de comandos: *java HolaMundo arg1 arg2 ...*

#### METHOD

*main()*

```
12 public class HolaMundo {
13
14     public static void main(String[] args) {
15         // TODO code application logic here
16         System.out.println("Hola Mundo!");
17     }
18
19 }
```

## 2. PRIMER EXEMPLE

### PRIMER EXEMPLE

Aquesta és la **funcionalitat de l'aplicació**. Aquesta línia mostra l'ús d'un nom de classe i un mètode. S'usa:

- el **mètode println()**
- de la **classe out**
- que està en el **paquet System**.

**Totes les instruccions** (creació de variables, crides a mètodes, assignacions) **s'han de finalitzar amb un punt i coma ";"**

```
12 public class HolaMundo {  
13  
14     public static void main(String[] args) {  
15         // TODO: code application logic here  
16         System.out.println("Hola Mundo!");  
17     }  
18  
19 }
```

LA FUNCIONALITAT  
DEL PROGRAMA

El **mètode println()** agafa una **cadena com a argument** i l'escriu en el **stream d'eixida estàndard**; en aquest cas, la finestra on es llança l'aplicació.

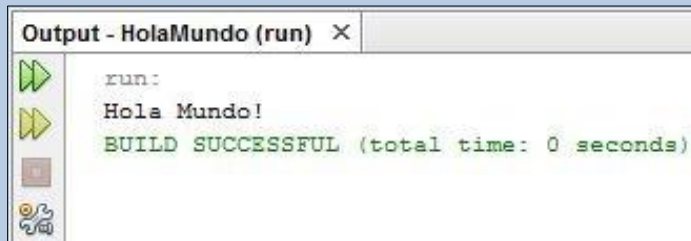
## 2. PRIMER EXEMPLE

### PRIMER EXEMPLE

DIY

```
12 public class HolaMundo {  
13  
14     public static void main(String[] args) {  
15         // TODO code application logic here  
16         System.out.println("Hola Mundo!");  
17     }  
18  
19 }
```

Resultat:



Output - HolaMundo (run) X

```
run:  
Hola Mundo!  
BUILD SUCCESSFUL (total time: 0 seconds)
```



# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA

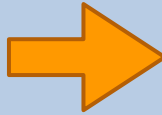


1. Introducció
2. Primer exemple
- 3. Elements bàsics**
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

### 3. ELEMENTS BÀSICS

#### DEFINICIÓ

**2 elements bàsics**



1. Comentaris
2. Identificadors

# 3. ELEMENTS BÀSICS

## 3.1 Comentaris

### 3 tipus de comentaris

// comentaris per a **una sola línia**

/\*  
comentaris d'**una o més línies**  
\*/

/\*\* **comentari de documentació**, d'una o més línies  
\*/

Són ignorats pel compilador i ens  
permeten explicar el que fa en  
nostre codi

**COMENTARIS DE DOCUMENTACIÓ:** *Col·locats immediatament abans d'una declaració (de variable o funció), indiquen que aqueix comentari ha de ser col·locat en la documentació que es genera automàticament quan s'utilitza l'eina **javadoc***

### 3. ELEMENTS BÀSICS

#### 3.1 Comentarís





# 3. ELEMENTS BÀSICS

## 3.2 Identificadors

Els identificadors **nomenen variables, funcions, classes i objectes**; qualsevol cosa que el programador o programadora necessite identificar o usar

### Regles per a crear identificadors

- Java és **CASE SENSITIVE** (*var1, Var1 i VAR1 són diferents*)
- Poden estar formats per **qualsevol dels caràcters** del codi **Unicode** (*podem declarar variables amb el nom: añoDeCreación, raïm, etc.*), encara que:
  - El **primer caràcter no** pot ser un **dígit numèric**
  - **No** poden utilitzar-se **espais en blanc ni símbols** coincidents amb **operadors**
- La **longitud** màxima dels identificadors és pràcticament **il·limitada**
- **No** pot ser una **paraula reservada del llenguatge** ni els valors lògics **true/ false**
- **No** poden ser **iguals a un altre identificador declarat** en el mateix àmbit

# 3. ELEMENTS BÀSICS

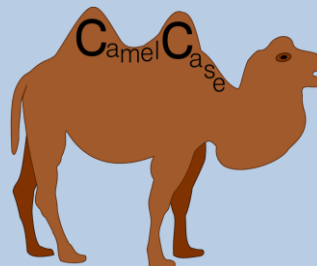
## 3.2 Identificadors

### Regles per a crear identificadors

**IMPORTANT** → Per conveni:

- Els **noms** de les **variables** i els **mètodes** haurien de començar per una **lletra minúscula** i els de les **classes** per **majúscula**.
- Si l'identificador està format per diverses **paraules**, la **primera** s'escriu en **minúscules** (*excepte per a les classes*) i la **resta** de paraules es fa **començar per majúscula** (*per exemple: añoDeCreación*).

Aquestes **regles** no són obligatòries, però **són convenientes** ja que ajuden al procés de codificació d'un programa, així com a la seua llegibilitat. És **més senzill distingir entre classes i mètodes o variables**.



# 3. ELEMENTS BÀSICS

## 3.2 Identificadors

### Exemples d'identificadors:

*comptador*

*suma*

*edat*

*sueldoBruto*

*sueldoNeto*

*nom\_usuari*

*nom\_Complet*

*letraDni*

### I el seu ús seria, per exemple:

*int comptador;*

*float sueldoNeto;*

*char letraDni;*

*// crea variable de tipus **int** anomenada **comptador***

*// crea variable de tipus **float** anomenada **sueldoNeto***

*// crea variable de tipus **char** anomenada **letraDni***

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA

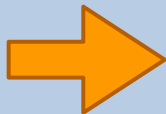


1. Introducció
2. Primer exemple
3. Elements bàsics
4. **Tipus de dades**
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

## 4. TIPUS DE DADES

### DEFINICIÓ

### 2 tipus de dades



- **Dades Simples:** ens permeten crear variables que emmagatzemen un sol valor (comptador, edat, preu, etc.)
- **Dades Compostes:** aquestes ens permeten emmagatzemar moltes dades (vectors, objectes, etc.) → *Les veurem més endavant*

Existeix un tipus de dada composta anomenada **String**, que convé conèixer ja, que permet representar text. Més endavant veurem com s'utilitza

## 4. TIPUS DE DADES

### Dades Simples

S'utilitzen per a	Tipus	Descripció	Memòria ocupada	Rang de valors permesos (* Java no realitza comprovació de rangs)
nombres enters	<b>byte</b>	Nombre enter d'1 byte	1 byte	-128 ... 127
	<b>short</b>	Nombre enter curt	2 bytes	-32768 ... 32767
	<b>int</b>	Nombre enter	4 bytes	-2147483648 ... 2147483647
	<b>long</b>	Nombre enter llarg	8 bytes	-9223372036854775808 ... 9223372036854775807
nombres reals	<b>float</b>	Nombre real amb coma flotant de precisió simple	32 bits	$\pm 3,4 \cdot 10^{-38} \dots \pm 3,4 \cdot 10^{38}$
	<b>double</b>	Nombre real amb coma flotant de precisió doble	64 bits	$\pm 1,7 \cdot 10^{-308} \dots \pm 1,7 \cdot 10^{308}$
caràcters	<b>char</b>	Un sol caràcter	2 bytes	
valors lògics	<b>boolean</b>	Valor lògic	1 bit	true o false

**\*Per exemple:** si a una variable de tipus **short** amb el valor **32.767** se li suma **1**, sorprenentment el **resultat serà -32.768** (**no produeix un error** de tipus desbordament com en altres llenguatges de programació, sinó que **es comporta de manera cíclica**)

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA



1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
- 5. Declaració de variables**
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

## 5. DECLARACIÓ DE VARIABLES

### DEFINICIÓ

**tipus** **identificador** [ = valor ] [, identificador [= valor] ... ] ;

OBLIGATORI

Opcional  
INICIALITZAR

Opcional  
DECLARAR MÉS  
VARIABLES

; per acabar TOTES les instruccions

Declarem un exemple:

int edat;

Ara inicialitzem-lo:

int edat = 25;    ≡    int edat;  
                              edat = 25;

Declarem un exemple:

float preu1, preu2, preu3;    ≡    float preu1;  
  float preu2;  
  float preu3;

Ara inicialitzem-lo:

float preu1 = 7.0;  
float preu2 = 7.25;  
float preu3 = 0.5;



## 5. DECLARACIÓ DE VARIABLES

### INICIALIZAR VARIABLES

```
tipus identificador [ = valor ][ , identificador [= valor] ... ] ;
```

OBLIGATORI

Opcional  
INICIALIZAR

Opcional  
DECLARAR MÉS  
VARIABLES

; per acabar TOTES les instruccions

Si una **variable no ha sigut inicialitzada**, Java li assigna un valor per defecte:

- Per a les variables de tipus **numèric**, el valor per defecte és zero ( **0** )
- Les variables de tipus **char**, el valor **'\u0000'**
- Les variables de tipus **booleà**, el valor **false**
- Per a les variables de tipus referencial (**objectes**), el valor **null**

## 5. DECLARACIÓ DE VARIABLES

### INICIALITZAR VARIABLES



## 5. DECLARACIÓ DE VARIABLES

### PARAULES CLAU I RESERVADES

#### Paraules clau

Les següents són paraules clau que **NO es poden utilitzar com a identificadors** ja que Java les utilitza per a altres coses:

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
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

#### Paraules reservades

A més, el llenguatge es reserva **unes quantes paraules més**, però que fins ara no tenen una finalitat especificada. Són:

cast	uture	generic	inner
operator	outer	rest	var


## 5. DECLARACIÓ DE VARIABLES

### 5.1 Àmbit d'una variable

*L'àmbit d'una variable és la porció del programa on aquesta es pot utilitzar*

L'àmbit d'una variable **depén del lloc del programa on és declarada**

4 categories diferents:

1. Variable local     *En aquesta unitat*
2. Atribut
3. Paràmetre d'un mètode
4. Paràmetre d'un tractador d'excepcions

## 5. DECLARACIÓ DE VARIABLES

### 5.2 Variables locals


*Una **variable local** es declara dins del cos d'un mètode d'una classe i és visible únicament dins d'aquest mètode.*

Es pot declarar en qualsevol lloc del cos, fins i tot després d'instruccions executables, encara que és un bon costum declarar-les just al principi.

També poden declarar-se variables dins d'un bloc amb claus {...}. En aqueix cas, **només** seran “visibles” dins d'aquest bloc.

Per exemple:

```
14 public static void main(String[] args) {  
15  
16     int i;  
17  
18     for (i=0;i<10;i++)  
19         System.out.println(i);  
20 }
```



*En aquest exemple existeix una variable local: **int i**; únicament pot utilitzar-se dins del bloc **main** on es va crear.*

## 5. DECLARACIÓ DE VARIABLES

### 5.3 Constants (final)

*Per a definir una constant a Java haurem de **precedir** la declaració d'una variable **de la paraula reservada final***

Per exemple, **creem variable constant** tipus **int** anomenada **x** amb valor **18**:

***final int x = 18;***

Per exemple, **creem variable constant** tipus **float** anomenada **pi** amb valor **3.14**:

***final float pi = 3.14;***

**Si posteriorment intentem modificar els seus valors es produirà un error i Java ens avisarà que no és possible.**

***x = 20;*** // no permés, produeix error

***pi = 7;*** // no permés, produeix error

DIY++

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA



1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
- 6. Operadors**
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

## 6. OPERADORS

### DEFINICIÓ

Ens permeten ***realitzar càlculs matemàtics i lògics.***

El tipus d'operadors són:

- ?????



## 6. OPERADORS

### DEFINICIÓ

*Ens permeten **realitzar càlculs matemàtics i lògics.***

El tipus d'operadors són:

- **Aritmètics**: sumes, restes, etc.
- **Relacionals**: menor, menor o igual, major, major o igual, etc.
- **Lògics**: AND, OR, NOT, etc.
- **Bits**: pràcticament no els utilitzarem en aquest curs.
- **Assignació**: =, +=, -= etc.

## 6. OPERADORS

### 6.1 Aritmètics




Operador	Format	Descripció
+	op1 + op2	Suma aritmètica de dos operands
-	op1 - op2 -op1	Resta aritmètica de dos operands. Canvi de signe
*	op1 * op2	Multiplicació de dos operands
/ %	op1 / op2 op1 % op2	Divisió sencera de dos operands. Resto de la divisió d'enters (o mòdul)
++	++op1 op1++	Increment unitari $x++$ equival a $x = x + 1$ $x--$ equival a $x = x - 1$
--	--op1 op1--	Decrement unitari

## 6. OPERADORS

### 6.1 Aritmètics

Els operadors ++ i -- admeten notació POSTfixa i PREfixa:

- **op1++**: Primer s'executa la instrucció en la qual està immers i després s'incrementa op1
- **op1--**: Primer s'executa la instrucció en la qual està immers i després es decrementa op1
- **++op1**: Primer s'incrementa op1 i després executa la instrucció en la qual està immers
- **--op1**: Primer se decrementa op1 i després executa la instrucció en la qual està immers

$i = i + 2;$	
$i += 2;$	
$++i++;$	

## 6. OPERADORS

### 6.1 Aritmètics



GENERALITAT  
VALENCIANA



Prova el següent;

```
int x=5;  
  
System.out.println(x++);  
System.out.println(x);
```

Prova el següent;

```
int x=5;  
  
System.out.println(++x);  
System.out.println(x);
```

Prova el següent;

```
int x=5;  
  
boolean resultat = x++>=6;  
System.out.println(resultat);  
System.out.println(x);
```

Prova el següent;

```
int x=5;  
  
boolean resultat = ++x>=6;  
System.out.println(resultat);  
System.out.println(x);
```

## 6. OPERADORS

### 6.2 Relacionals

Operador	Format	Descripció
>	op1 > op2	Retorna true (cert) si op1 és major que op2
<	op1 < op2	Retorna true (cert) si op1 és menor que op2
>=	op1 >= op2	Retorna true (cert) si op1 és major o igual que op2
<=	op1 <= op2	Retorna true (cert) si op1 és menor o igual que op2
==	op1 == op2	Retorna true (cert) si op1 és igual a op2
!=	op1 != op2	Retorna true (cert) si op1 és diferente a op2

Actuen sobre valors sencers, reals i caràcters (char); i **retornen un valor del tipus booleà**



SEMPRE !!!!

# 6. OPERADORS

## 6.2 Relacionals

Per exemple:

```
15 public static void main(String[] args){
16
17     double op1,op2;
18     char op3,op4;
19
20     op1=1.34;
21     op2=1.35;
22     op3='a';
23     op4='b';
24
25     System.out.println("op1=" + op1 + " op2=" + op2);
26     System.out.println("op1>op2 = " + (op1 > op2));
27     System.out.println("op1<op2 = " + (op1 < op2));
28     System.out.println("op1==op2 = " + (op1 == op2));
29     System.out.println("op1!=op2 = " + (op1 != op2));
30     System.out.println("'a'>'b' = " + (op3 > op4));
31
32 }
```

Eixida???



GENERALITAT  
VALENCIANA



J.E.S. Maria Enriqueta  
García

## 6. OPERADORS

### 6.2 Relacionals

Per exemple:

```
15 public static void main(String[] args){
16
17     double op1,op2;
18     char op3,op4;
19
20     op1=1.34;
21     op2=1.35;
22     op3='a';
23     op4='b';
24
25     System.out.println("op1=" + op1 + " op2=" + op2);
26     System.out.println("op1>op2 = " + (op1 > op2));
27     System.out.println("op1<op2 = " + (op1 < op2));
28     System.out.println("op1==op2 = " + (op1 == op2));
29     System.out.println("op1!=op2 = " + (op1 != op2));
30     System.out.println("'a'>'b' = " + (op3 > op4));
31
32 }
```

Eixida

```
run:
op1=1.34 op2=1.35
op1>op2 = false
op1<op2 = true
op1==op2 = false
op1!=op2 = true
'a'>'b' = false
BUILD SUCCESSFUL (total
```



GENERALITAT  
VALENCIANA



J.E.S. Maria Enríquez  
Gandia

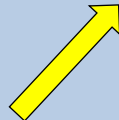
## 6. OPERADORS

### 6.3 Lògics

Operador	Format	Descripció
<b>&amp;&amp;</b>	op1 && op2	AND lògic: Retorna True (cert) si són certs op1 i op2
<b>  </b>	op1    op2	OR lògic: Retorna True (cert) si són certs op1 o op2
<b>!</b>	!op1	NOT lògica: Retorna True (cert) si op1 es Fals

**Actuen sobre operadors o expressions lògiques**, és a dir, aquells que s'avaluen a True / False

**SEMPRE !!!!**





## 6. OPERADORS

### 6.3 Lògics



GENERALITAT  
VALENCIANA



Per exemple:

```
15 public static void main(String[] args){
16
17     boolean a, b, c, d;
18
19     a=true;
20     b=true;
21     c=false;
22     d=false;
23
24     System.out.println("true Y true = " + (a && b) );
25     System.out.println("true Y false = " + (a && c) );
26     System.out.println("false Y false = " + (c && d) );
27     System.out.println("true O true = " + (a || b) );
28     System.out.println("true O false = " + (a || c) );
29     System.out.println("false O false = " + (c || d) );
30     System.out.println("NO true = " + !a);
31     System.out.println("NO false = " + !c);
32     System.out.println("(3 > 4) Y true = " + ((3 >4) && a) );
33
34 }
35 }
```

Eixida???



# 6. OPERADORS

## 6.3 Lògics

Per exemple:


```
15 public static void main(String[] args){
16
17     boolean a, b, c, d;
18
19     a=true;
20     b=true;
21     c=false;
22     d=false;
23
24     System.out.println("true Y true = " + (a && b) );
25     System.out.println("true Y false = " + (a && c) );
26     System.out.println("false Y false = " + (c && d) );
27     System.out.println("true O true = " + (a || b) );
28     System.out.println("true O false = " + (a || c) );
29     System.out.println("false O false = " + (c || d) );
30     System.out.println("NO true = " + !a);
31     System.out.println("NO false = " + !c);
32     System.out.println("(3 > 4) Y true = " + ((3 >4) && a) );
33
34 }
35 }
```

Eixida

```
run:
true Y true = true
true Y false = false
false Y false = false
true O true = true
true O false = true
false O false = false
NO true = false
NO false = true
(3 > 4) Y true = false
BUILD SUCCESSFUL (total time:
```

## 6. OPERADORS

### 6.4 D'assignació

***variable = expressió***  Assigna a la variable el resultat d'avaluar l'expressió de la dreta

Operador	Format	Equivalència
<b>+=</b>	op1 += op2	op1 = op1 + op2
<b>-=</b>	op1 -= op2	op1 = op1 - op2
<b>*=</b>	op1 *= op2	op1 = op1 * op2
<b>/=</b>	op1 /= op2	op1 = op1 / op2
<b>%=</b>	op1 %= op2	op1 = op1 % op2
<b>&amp;=</b>	op1 &= op2	op1 = op1 & op2
<b> =</b>	op1  = op2	op1 = op1   op2
<b>^=</b>	op1 ^= op2	op1 = op1 ^ op2
<b>&gt;&gt;=</b>	op1 >>= op2	op1 = op1 >> op2
<b>&lt;&lt;=</b>	op1 <<= op2	op1 = op1 << op2
<b>&gt;&gt;&gt;=</b>	op1 >>>= op2	op1 = op1 >>> op2

[https://es.wikibooks.org/wiki/Programaci%C3%B3n\\_en\\_Java/Operadores\\_de\\_bits](https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Operadores_de_bits)

És possible combinar l'operador d'assignació amb altres operadors per a, de forma abreujada, realitzar un càlcul i assignar-lo a una variable.

## 6. OPERADORS

### 6.5 Precedència d'operadors

El llenguatge Java avalua les expressions aplicant els operadors un a un seguint un ordre específic.

És important conèixer almenys els més utilitzats: matemàtics, relacionals, lògics i d'assignació.



GENERALITAT  
VALENCIANA



1. Operadors postfixos: `expr++`, `expr--`, `()`, `.`, `[]`, `{}`
2. Operadors unaris: `++expr`, `--expr`, `-expr`, `~`, `!`
3. Creació o conversió de tipus: `new (tipus)expr`
4. Multiplicació i divisió: `*`, `/`, `%`
5. Suma i resta: `+`, `-`
6. Desplaçament de bits: `<<`, `>>`, `>>>`
7. Relacionals: `<`, `>`, `<=`, `>=`
8. Igualtat i desigualtat: `==`, `!=`
9. AND a nivell de bits: `&`
10. AND lògic: `&&`
11. XOR a nivell de bits: `^`
12. OR a nivell de bits: `|`
13. OR lògic: `||`
14. Operador condicional: `?:`
15. Assignació: `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `^=`, `&=`, `|=`, `>>=`, `<<=`

## 6. OPERADORS

### 6.6 La classe Math

Una classe especial anomenada **Math** dins del paquet **java.lang** inclou operadors matemàtics més complexos i potents (*càlcul de potències, arrels quadrades, valors absoluts, sinus, cosinus, etc.*)

Per exemple:


















```
double x = Math.pow(3,3);    // Potència 3 ^ 3  
double i = Math.sqrt(9);     // Arrel quadrada de 9
```

També posseeix constants com:

```
double PI = Math.PI --> El número  $\Pi$ (3,1416...)  
double E = Math.E --> El número e (2, 7182...)
```

DIY++

Alguns exemples d'altres mètodes:

 <b>E</b>	double	▲
 <b>PI</b>	double	
 IEEEremainder(double f1, double f2)	double	
 abs(double a)	double	
 abs(float a)	float	
 abs(int a)	int	
 abs(long a)	long	
 acos(double a)	double	
 addExact(int x, int y)	int	
 addExact(long x, long y)	long	
 asin(double a)	double	
 atan(double a)	double	
 atan2(double y, double x)	double	
 cbrt(double a)	double	
 ceil(double a)	double	
 copySign(double magnitude, double sign)	double	
 copySign(float magnitude, float sign)	float	▼

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA



1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
- 7. Literals**
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

## 7. LITERAL

### DEFINICIÓ

A l'hora de **tractar amb valors dels tipus de dades simples (i Strings)** s'utilitza el que es denomina “literals”. Els literals són elements que **serveixen per a representar un valor en el codi font del programa**.

A Java existeixen literals per als següents tipus de dades:

1. Lògics (**boolean**)
2. Caràcter (**char**)
3. Enters (**byte, short, int i long**)
4. Reals (**double i float**)
5. Cadenes de caràcters (**String**)

## 7. LITERAL

### 7.1 Literals lògics

Són únicament dos, les paraules reservades ***true*** i ***false***.

Exemple:

*boolean activat = false;*



## 7. LITERAL

### 7.2 Literals enters

Els literals de tipus enters (nombres enters): ***byte, short, int i long***

Poden expressar-se en:

- Decimal: base 10 --> 0 a 9
- Octal: base 8 --> 0 a 7
- Hexadecimal: base 16 --> ??

## 7. LITERAL

### 7.2 Literals enters

Els literals de tipus enters (nombres enters): **byte**, **short**, **int** i **long**

Poden expressar-se en:

- Decimal: base 10 --> 0 a 9
- Octal: base 8 --> 0 a 7
- Hexadecimal: base 16 --> 0

```
int nDec = 24;  
int nOct = 024;  
int nHex = 0x24;  
System.out.println("decimal= " + nDec + ", octal= " + nOct + ", hex = " + nHex);
```

La lletra **L** al final d'un literal de tipus enter pot aplicar-se a qualsevol sistema de numeració i indica que el nombre decimal siga tractat com un enter llarg (de 64 bits).

Exemple:

`long max1 = 9223372036854775807L; //aquest és el valor màxim per a un enter llarg`

## 7. LITERAL

### 7.3 Literals reals

Els literals de tipus real serveixen per a indicar valors *float* o *double*.

A diferència dels literals de tipus enter, **no poden expressar-se en Octal o Hexadecimal**.

Existeixen dos formats de representació:

- Mitjançant la seua part sencera, el punt decimal ( . ) i la part fraccionària
- Mitjançant [notació exponencial o científica](#)

Exemples equivalents:

3.1415  
0.31415e1

.31415e1  
0.031415E+2

.031415e2  
314.15e-2

31415E-4

Exemples equivalents:

3.1415f  
0.31415e1f

.31415e1f  
0.031415E+2f

.031415e2f  
314.15e-2f

31415E-4f

## 7. LITERAL

### 7.4 Literals caràcter

Els literals de tipus caràcter es representen sempre entre cometes simples.

Entre les cometes simples pot aparéixer:

- Un **símbol**: lletra associada a un codi [Unicode](#)  
Exemples: 'a', 'B', '{', 'ñ', 'á'
- Una **“seqüència de fuga”**: són combinacions del símbol contrabarra \ seguit d'una lletra, i serveixen per a representar caràcters que no tenen una equivalència en forma de símbol

Les possibles seqüències de fuga són:

- `\n` -----> Nova Línia
- `\t` -----> Tabulador
- `\r` -----> Reculada de Carro
- `\f` -----> Començament de Pàgina
- `\b` -----> Esborrat a l'Esquerra
- `\\` -----> El caràcter barra inversa ( \ )
- `\'` -----> El caràcter preval simple ( ' )
- `\"` -----> El caràcter preval doble o bi-prima ( " )

Per exemple :

Diagonal inversa s'utilitza: `\'`

Cometes dobles en un String s'utilitza: `\"`

## 7. LITERAL

### 7.5 Literals cadenes (String)

Un literal de tipus String **va tancat entre cometes dobles ( " )** i ha d'estar inclòs completament **en una sola línia del programa font** (no pot dividir-se en diverses línies).

#### **NOTA 1**

Si el String és massa llarg i ha de dividir-se en diverses línies en el fitxer font, pot utilitzar-se l'operador de concatenació de Strings (+) de la següent forma:

```
"Aquest String és massa llarg per a estar en una línia del " +  
"fitxer font i s'ha dividit en dues."
```

Entre les cometes dobles **pot incloure's qualsevol caràcter del codi Unicode** (o el seu codi precedit del caràcter \ ) **a més de les seqüències de fuga** vistes anteriorment.

#### **NOTA 2**

La manera d'incloure els caràcters cometes dobles ( " ) i contrabarra ( \ ) és mitjançant les seqüències de fuga \" i \\ respectivament (o mitjançant el seu codi [Unicode](#) precedit de \ )

## 7. LITERAL

### 7.5 Literals cadenes (String)

#### Exemple:

Per a incloure un canvi de línia dins d'un literal de tipus string haurà de fer-se mitjançant la seqüència de fuita `\n` :

```
System.out.println("Primera línia\nSegona línia del string");
```

```
System.out.print("Hola");
```

Com serà l'eixida per pantalla?

## 7. LITERAL

### 7.5 Literals cadenes (String)

#### Exemple:

Per a incloure un canvi de línia dins d'un literal de tipus string haurà de fer-se mitjançant la seqüència de fuita `\n` :

```
System.out.println("Primera línia\nSegona línia del string");
```

```
System.out.print("Hola");
```

La visualització del *string* anterior mitjançant `println()` produiria la següent eixida per pantalla:

**Primera línia**

**Segona línia del string**

**Hola**

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



GENERALITAT  
VALENCIANA



1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples



## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.1 Eixida estàndard

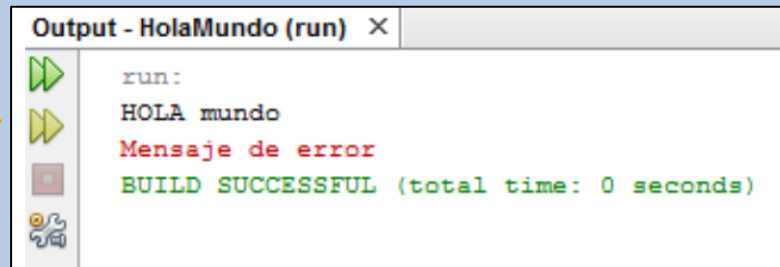
Ja hem vist l'ús de **System.out** per a mostrar informació per pantalla:

- `print("...")` imprimeix text per pantalla
- `println("...")` imprimeix text per pantalla i **introdueix un salt de línia**

La utilització de **System.err** seria totalment anàloga a `System.out`, però per a enviar els missatges produïts per **errors en l'execució**.

```
14 public static void main(String[] args) {  
15  
16     System.out.print("HOLA ");  
17     System.out.println("mundo");  
18     System.err.println("Mensaje de error");  
19 }
```

Eixida



Output - HolaMundo (run) X

```
run:  
HOLA mundo  
Mensaje de error  
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.1 Eixida estàndard

També podem imprimir variables de qualsevol tipus, així com **combinacions de text i variables concatenades** amb l'operador +

Per exemple:

```
14 public static void main(String[] args) {  
15     String nombre = "Pepito";  
16     int edad = 25;  
17     System.out.println(nombre);  
18     System.out.println(edad);  
19     System.out.println(nombre + " tiene " + edad + " años");  
20 }
```

Eixida ??



## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.1 Eixida estàndard

També podem imprimir variables de qualsevol tipus, així com **combinacions de text i variables concatenades** amb l'operador +

Per exemple:

```
14 public static void main(String[] args) {  
15     String nombre = "Pepito";  
16     int edad = 25;  
17     System.out.println(nombre);  
18     System.out.println(edad);  
19     System.out.println(nombre + " tiene " + edad + " años");  
20 }
```

Eixida

```
Pepito  
25  
Pepito tiene 25 años
```

## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.2 Entrada estàndard



GENERALITAT  
VALENCIANA



Hi ha diverses maneres de **llegir informació del teclat escrita per l'usuari**, però la més senzilla és utilitzar la **classe Scanner**.

- **Primer** declarar un **objecte Scanner** que llija de l'entrada estandar *System.in*:

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

**NOTA:** En aquest exemple hem creat un **objecte Scanner** anomenat "reader" però podríem posar-li qualsevol nom.

- Ara podrem utilitzar *reader* tantes vegades com vulguem per a llegir informació del teclat:

```
String texto = reader.nextLine();
```

El mètode ***reader.nextLine()*** recollirà el text que l'usuari escriga per teclat (fins a pressionar la tecla Intro) i ho guardarà en **text** (de tipus *String*).

## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.2 Entrada estàndard

Existeixen molt altres mètodes segons la mena de dada que es vulga llegir:

- **nextByte()**: obté un nombre enter tipus byte
- **nextShort()**: obté un nombre enter tipus short
- **nextInt()**: obté un nombre enter tipus int
- **nextLong()**: obté un nombre enter tipus long
- **nextFloat()**: obté un nombre real float
- **nextDouble()**: obté un nombre real double
- **next()**: obté el següent *\*token* (lleg text fins a un espai)
- **nextline()**: obté cadena de text (fins a pressionar la tecla Intro)

No existeixen mètodes de la classe Scanner per a obtindre directament booleans ni per a obtindre un sol caràcter

*\*Un Scanner divideix la seua entrada en tokens usant un patró delimitador, que per defecte coincideix amb l'espai en blanc*

# 8. EIXIDA I ENTRADA ESTÀNDARD

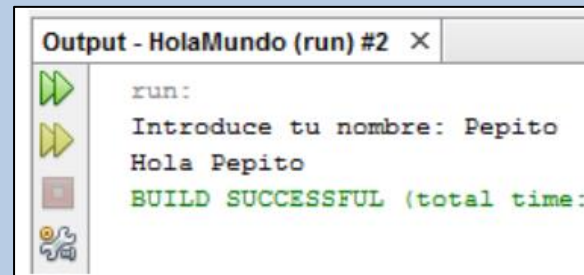
## 8.2 Entrada estàndard

Exemple: Llegim una cadena de text i la mostrem per pantalla.

```
12 import java.util.Scanner;
13
14 public class EjemploScanner {
15
16     public static void main(String[] args){
17
18         String nombre;
19
20         Scanner entrada = new Scanner(System.in);
21
22         System.out.print("Introduce tu nombre: ");
23
24         nombre = entrada.nextLine();
25
26         System.out.println("Hola " + nombre);
27     }
28 }
29
```

**DIY**

Eixida



**IMPORTANT:** Per a poder utilitzar la classe *Scanner* és necessari importar-la des del paquet *java.util* de Java: **import java.util.Scanner;**

## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.2 Entrada estàndard

Exemple: llegim un valor tipus double. El programa demana a l'usuari que introduïsca el radi d'un cercle, després calcula la seua àrea i circumferència, finalment el mostra per pantalla. Utilitza els mètodes i constants que proporciona la classe *Math*.

DIY

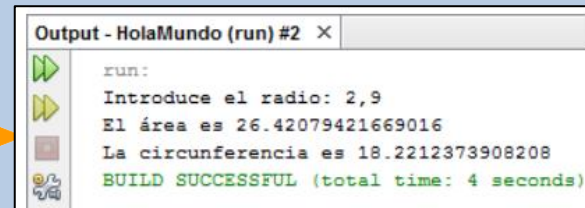
## 8. EIXIDA I ENTRADA ESTÀNDARD

### 8.2 Entrada estàndard

Exemple: llegim un valor tipus double. El programa demana a l'usuari que introduïsca el radi d'un cercle, després calcula la seua àrea i circumferència, finalment el mostra per pantalla. Utilitza els mètodes i constants que proporciona la classe *Math*.

```
12 import java.util.Scanner;
13
14 public class EjemploScanner {
15
16     public static void main(String[] args){
17
18         double radio, area, circumferencia;
19
20         Scanner entrada = new Scanner(System.in);
21
22         System.out.print("Introduce el radio: ");
23
24         radio = entrada.nextDouble();
25
26         // Se hace uso de la libreria Math para usar PI y la portencia(pow)
27         area = Math.PI * Math.pow(radio, 2);
28
29         circumferencia = 2 * Math.PI * radio;
30
31         System.out.println("El área es " + area);
32
33         System.out.println("La circumferencia es " + circumferencia);
34     }
35 }
```

Eixida



```
run:
Introduce el radio: 2,9
El área es 26.42079421669016
La circumferencia es 18.2212373908208
BUILD SUCCESSFUL (total time: 4 seconds)
```



# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT

1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
- 9. Estructures alternatives**
10. Exemples

## 9. ESTRUCTURES ALTERNATIVES

### DEFINICIÓ

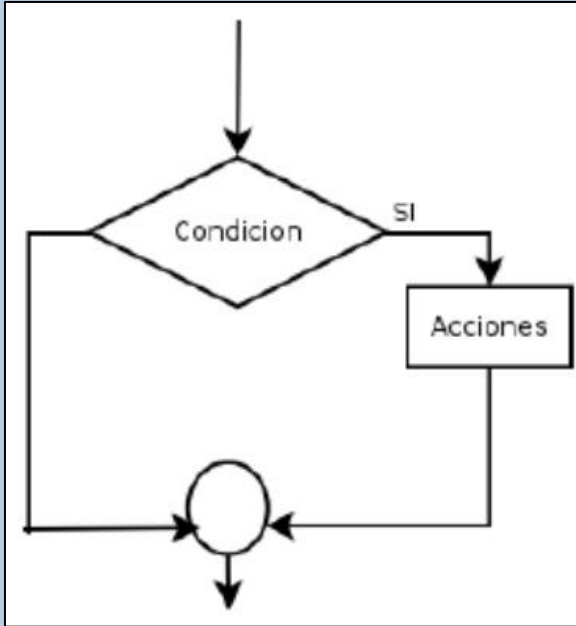
Recordem que les estructures alternatives són construccions que **permeten alterar el flux seqüencial d'un programa** de manera que **en funció d'una condició o el valor d'una expressió**, el mateix pugui ser desviat en l'una o l'altra alternativa de codi.

Les estructures alternatives disponibles a Java són:

- Alternativa Simple (**if**)
- Alternativa Doble (**if-else**)
- Alternativa Múltiple (**switch**)

## 9. ESTRUCTURES ALTERNATIVES

### 9.1 Estructura Alternativa Simple (if)



```
if (condició)
{
    // Accions;
}
```

El bloc d'**Accions** s'executa si la condició és vertadera

```
if (cont == 0)
{
    System.out.println("cont és 0");
    // més instruccions...
}
```

**Si** dins del if **només hi ha una instrucció**, no és necessari posar les claus

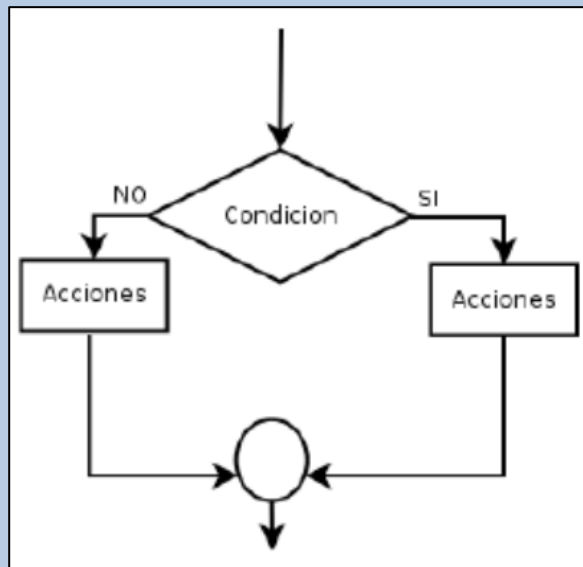
```
if (cont == 0) System.out.println("cont és 0");
```

## 9. ESTRUCTURES ALTERNATIVES

### 9.2 Estructura Alternativa Doble (if-else)

El bloc del IF s'executa si la condició és vertadera.

En cas contrari, s'executa el bloc de l'ELSE.



```
if (cont == 0)
{
    System.out.println("cont és 0");
    // més instruccions...
}
else
{
    System.out.println("cont no és 0");
    // més instruccions...
}
```

**Si dins del if o el else només hi ha una instrucció, no és necessari posar les claus.**

```
if (cont == 0) System.out.println("cont és 0");
else System.out.println("cont no és 0");
```

## 9. ESTRUCTURES ALTERNATIVES

### 9.2 Estructura Alternativa Doble (if-else)



En moltes **ocasions**, es nien **estructures alternatives if-else**, de manera que es pregunte per una condició si anteriorment no s'ha complit una altra successivament.

Per exemple: suposem que realitzem un programa que mostra la nota d'un alumne en la forma (insuficient, suficient, bé, notable o excel·lent) en funció de la seua nota numèrica. (fesho utilitzant if-else niats)

DIY



GENERALITAT  
VALENCIANA



# 9. ESTRUCTURES ALTERNATIVES

## 9.2 Estructura Alternativa Doble (if-else)



GENERALITAT  
VALENCIANA



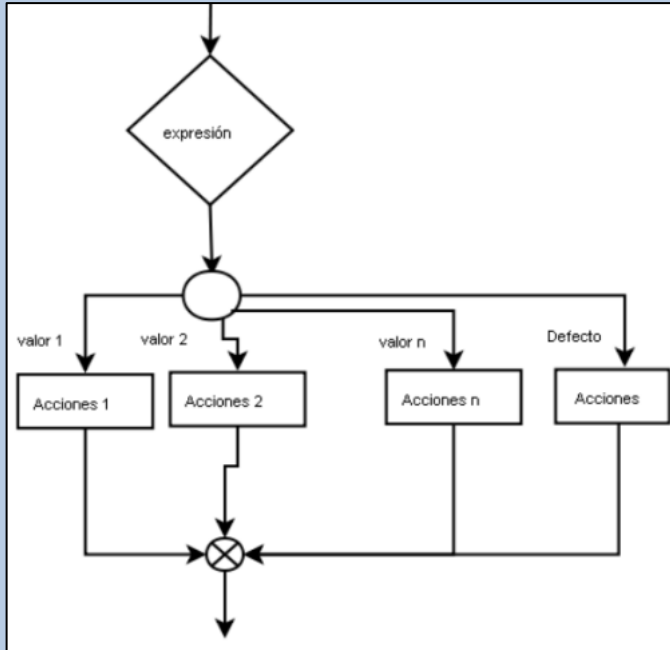
```
2 import java.util.Scanner;
3
4 public class Nota {
5
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         int nota;
9         //Suponemos que el usuario introduce el número correctamente.
10        //No hacemos comprobación
11        System.out.println("Dame un número entre 0 y 10");
12
13        nota = entrada.nextInt();
14
15        if (nota < 5) {
16            System.out.println("Insuficiente");
17        } else if (nota < 6) {
18            System.out.println("Suficiente");
19        } else if (nota < 7) {
20            System.out.println("Bien");
21        } else if (nota < 9) {
22            System.out.println("Notable");
23        } else {
24            System.out.println("Sobresaliente");
25        }
26    }
27 }
28 }
```

Eixida

```
run:
Dame un número entre 0 y 10
8
Notable
BUILD SUCCESSFUL (total time: 11 seconds)
```

## 9. ESTRUCTURES ALTERNATIVES

### 9.3 Estructura Alternativa Múltiple (switch)



```
switch (expressió)
{
    case valor1:
        // Accions1;
        break;
    case valor2:
        // Accions2;
        break;
    case valorN:
        // AccionsN;
        break;
    default:
        // Accions per defecte;
}
```

És molt important entendre que en el switch s'avalua una expressió (un valor concret com 0, 5, 1...) no una condició (vertadera o falsa) com en el if i el if-else

## 9. ESTRUCTURES ALTERNATIVES

### 9.3 Estructura Alternativa Múltiple (switch)

```
switch (expressió)
{
    case valor1:
        // Accions1;
        break;
    case valor2:
        // Accions2;
        break;
    case valorN:
        // AccionsN;
        break;
    default:
        // Accions per defecte;
}
```

El programa comprova el valor de l'expressió i saltarà al 'case' que corresponga amb aquest valor (valor1 o valor2 o ...) executant el codi de dita 'case' (Accions1 o Accions2 o ...). Si no coincideix cap valor, saltarà al **'default'** i executarà les accions per defecte.

**IMPORTANT:** afegir la sentència **break;** al final de cada 'case', ja que en cas contrari el programa continuarà executant el codi de les altres accions i normalment no voldrem que faça això.



# 9. ESTRUCTURES ALTERNATIVES

## 9.3 Estructura Alternativa Múltiple (switch)

```
2 import java.util.Scanner;
3
4 public class Alternativa_Multiple {
5
6     public static void main(String[] args) {
7         Scanner entrada = new Scanner(System.in);
8         int dia;
9
10        System.out.println("Dame un número entre 1 y 7:");
11
12        dia = entrada.nextInt();
13
14        switch (dia) {
15            case 1:
16                System.out.println("Lunes");
17                break;
18            case 2:
19                System.out.println("Martes");
20                break;
21            case 3:
22                System.out.println("Miércoles");
23                break;
24            case 4:
25                System.out.println("Jueves");
26                break;
27            case 5:
28                System.out.println("Viernes");
29                break;
30            case 6:
31                System.out.println("Sábado");
32                break;
33            case 7:
34                System.out.println("Domingo");
35                break;
36            default:
37                System.out.println("Error el número debe estar entre 0 y 7");
38        }
39    }
40 }
41
42 }
```

**DIY**

**Eixida**

```
run:
Dame un número entre 1 y 7:
4
Jueves
BUILD SUCCESSFUL (total time: 2 seconds)
```

## 9. ESTRUCTURES ALTERNATIVES

### 9.3 Estructura Alternativa Múltiple (switch)

```
public class Alternativa_Multiple{  
  
    public static void main(String[] args)  
    {  
        Scanner entrada = new Scanner(System.in);  
        int dia;  
        System.out.println("Dame un nombre entre 1 i 7:");  
        dia=entrada.nextInt();  
  
        switch (dia)  
        {  
            //multiples cases sin declaraciones break  
  
            case 1:  
            case 2:  
            case 3:  
            case 4:  
            case 5:  
                System.out.println("Dia laborable");  
                break;  
            case 6:  
            case 7:  
                System.out.println("Fin de semana");  
                break;  
  
            default: System.out.println("Error el nombre deu estar entre 1 i 7");  
        }  
    }  
}
```



Què faria aquest?

# INTRODUCCIÓ A JAVA

## ÍNDEX DE CONTINGUT



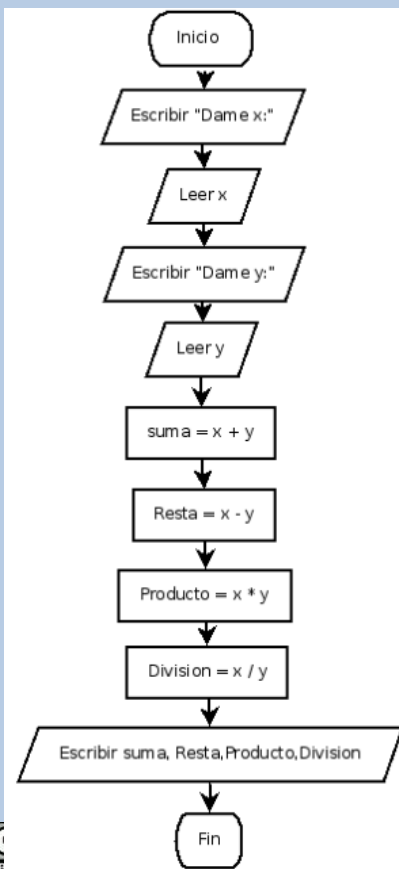
GENERALITAT  
VALENCIANA



1. Introducció
2. Primer exemple
3. Elements bàsics
4. Tipus de dades
5. Declaració de variables
6. Operadors
7. Literals
8. Eixida i entrada estàndard
9. Estructures alternatives
10. Exemples

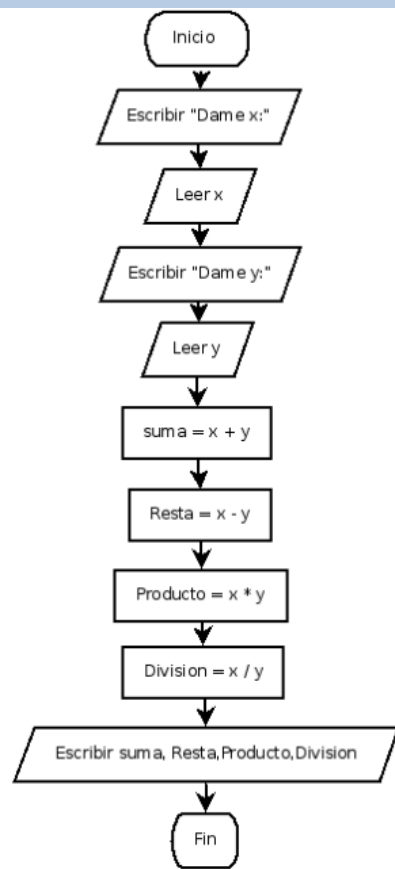
## 10. EJEMPLOS

Exemple 1: Programa que llija dos números, calcule i mostre el valor de les seues suma, resta, producte i divisió.



## 10. EXEMPLES

Exemple 1: Programa que llija dos números, calcule i mostre el valor de les seues suma, resta, producte i divisió.



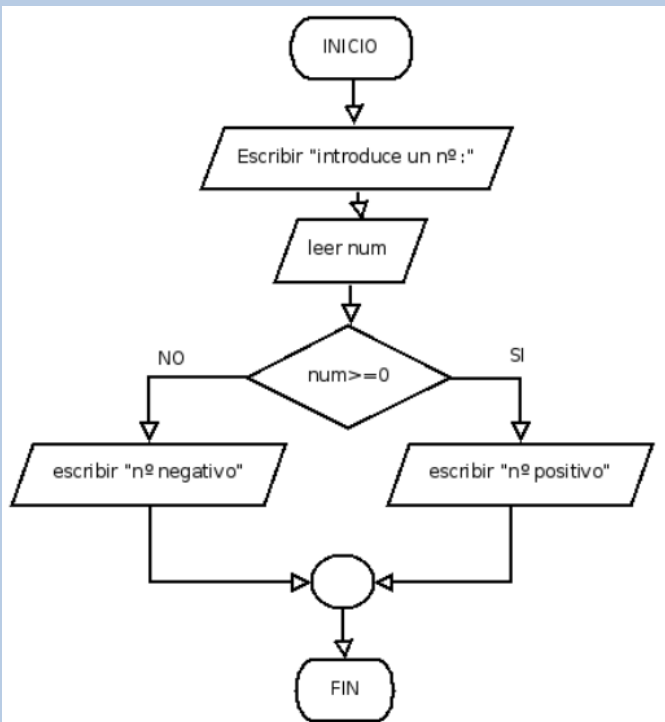
```
1 package ejemplo1;
2
3 import java.util.Scanner; // Importamos la clase Scanner
4
5 public class Ejemplo1 {
6
7     public static void main(String[] args) {
8
9         // Declaramos las variables que vamos a necesitar
10        int x, y, suma, resta, mult, div;
11
12        // Creamos el objeto Scanner para leer por teclado
13        Scanner reader = new Scanner(System.in);
14
15        // Pedimos y leemos x
16        System.out.print("Dame x:");
17        x = reader.nextInt();
18
19        // Pedimos y leemos y
20        System.out.print("Dame y:");
21        y = reader.nextInt();
22
23        // Realizamos los cálculos necesarios
24        suma = x + y;
25        resta = x - y;
26        mult = x * y;
27        div = x / y;
28
29        // Mostramos los cálculos por pantalla
30        System.out.println("Suma: " + suma);
31        System.out.println("Resta: " + resta);
32        System.out.println("Multiplicación: " + mult);
33        System.out.println("División: " + div);
34    }
35 }
```

Eixida

```
run:
Dame x:4
Dame y:2
Suma: 6
Resta: 2
Multiplicación: 8
División: 2
```

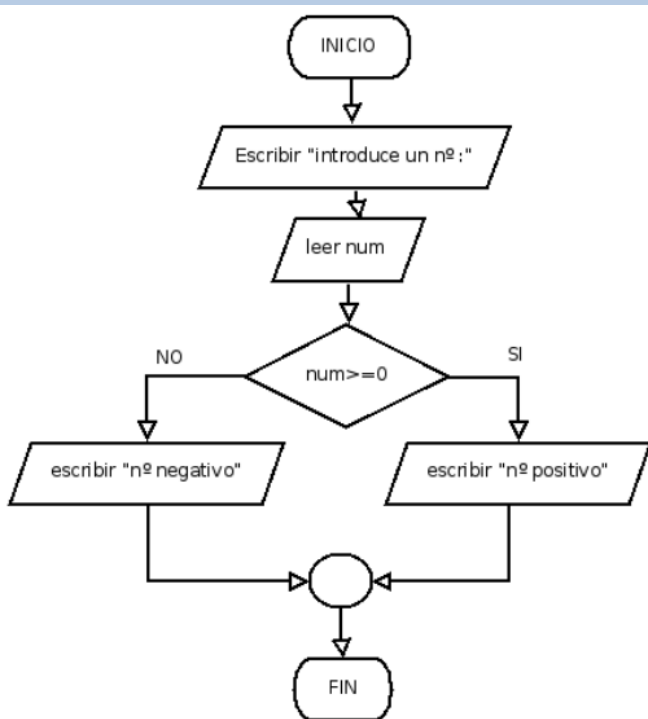
## 10. EXEMPLES

Exemple 2: Programa que llig un número i em diu si és positiu o negatiu. Considerarem el zero com a positiu.



## 10. EXEMPLES

Exemple 2: Programa que llig un número i em diu si és positiu o negatiu. Considerarem el zero com a positiu.



```
1 package ejemplo2;
2
3 import java.util.Scanner; // Importamos la clase Scanner
4
5 public class Ejemplo2 {
6
7     public static void main(String[] args) {
8
9         // Declaramos la variable num
10        int num;
11
12        // Creamos el objeto Scanner para leer por teclado
13        Scanner reader = new Scanner(System.in);
14
15        // Pedimos y leemos x
16        System.out.print("Introduce un nº: ");
17        num = reader.nextInt();
18
19        // Aestructura alternativa doble
20        if (num >= 0)
21            System.out.println("Número positivo");
22        else
23            System.out.println("Número negativo");
24    }
25 }
```

Eixida



```
run:
Introduce un nº: 10
Número positivo
```

# EXERCICIS PROPOSATS



## Autor:

José Manuel Martí Fenollosa

## Revisat per:

Àngel Olmos Giner

## Llicència:



**CC BY-NC-SA 3.0 ES Reconeixement – No Comercial – Compartir Igual (by-nc-sa)**

No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a la que regula l'obra original. Aquesta és una obra derivada de l'obra original de José Manuel Martí Fenollosa