



Unió Europea

Fons Social Europeu

L'FSE inverteix en el teu futur



**GENERALITAT
VALENCIANA**



UF07. - FUNCIONS

- Teoria -

**PROGRAMACIÓ
CFGs DAM**

Autor:

José Manuel Martí Fenollosa

Revisat per:

Àngel Olmos Giner

a.olmosginer@edu.gva.es

2022/2023

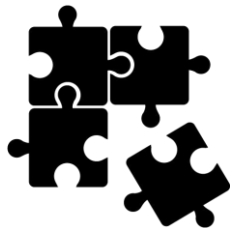
FUNCIONS

ÍNDEX DE CONTINGUT

- 1. INTRODUCCIÓ**
- 2. DECLARACIÓ D'UNA FUNCIO**
- 3. ANOMENA A UNA FUNCIO**
- 4. ÀMBIT DE LES VARIABLES**
- 5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA**
- 6. DEVOLUCIÓ D'UN VALOR**

1. INTRODUCCIÓ

INTRODUCCIÓ



La millor manera de crear i mantindre un **programa** gran és construir-lo **a partir de peces més xicotetes o mòduls**. Cadascun dels quals és més **manejable** que el programa íntegrament

Les **FUNCIONS** (o **MÈTODES**) són utilitzades per a **evitar la repetició de codi en un programa** en poder executar-les des de diversos punts d'un programa amb només invocar-les

Permeten la **descomposició funcional i la diferenciació de tasques**

1. INTRODUCCIÓ

INTRODUCCIÓ

Per a què les utilitzem?

- Agrupar codi que **forma una entitat pròpia** o una idea concreta
- Agrupar codi que es necessitarà diverses vegades en un programa, amb al missió de **no repetir codi**
- **Dividir** el codi d'un **programa gran** un **subprogrames (funcions)**, cadascun d'ells especialitzats a resoldre una part del problema

Característiques

- Es defineixen mitjançant un **nom únic** que representa el bloc de codi
- Poden ser **anomenades (executades)** des de qualsevol part del codi
- **Se'ls pot passar valors** perquè els processen d'alguna forma
- **Poden retornar un resultat** per a ser usat des d'on se'ls haja anomenat

1. INTRODUCCIÓN

INTRODUCCIÓN

C

O

D

E

Etapa 20: Funciones en Minecraft

10

MÁS

Iniciar Sesión

?

≡

Vídeo: Hora del Código de Minecraft - Funciones

C

O

D

E

Minecraft Hour of Code - Functions

Copiar en...



The video player displays a Minecraft Education Edition scene. In the foreground, a grey robot character with a red chest and wooden arms stands on a stone block. The background shows a Minecraft landscape with green grass, a body of water, and a small wooden structure. A red YouTube play button is overlaid in the center of the video. The video title 'Minecraft Hour of Code - Functions' is visible in the top left corner of the video frame, along with a small video feed of a person.

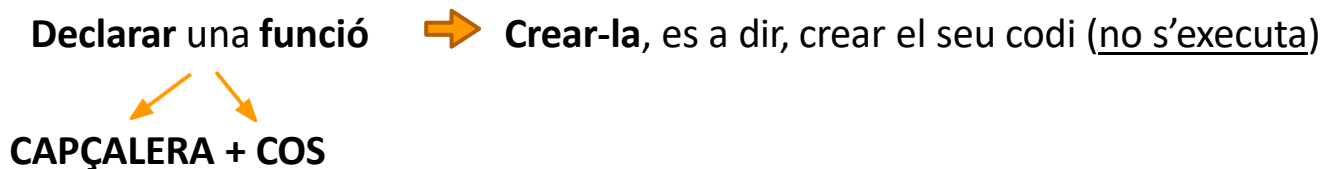
FUNCIONS

ÍNDEX DE CONTINGUT

- 1. INTRODUCCIÓ**
- 2. DECLARACIÓ D'UNA FUNCIO**
- 3. ANOMENA A UNA FUNCIO**
- 4. ÀMBIT DE LES VARIABLES**
- 5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA**
- 6. DEVOLUCIÓ D'UN VALOR**

2. DECLARACIÓ D'UNA FUNCIO

DEFINICIÓ



La **capçalera** es declara en una sola línia i es compon de:

- **Modificadors de funció:** Existeixen molts però els veurem en futures unitats. (Per ara només utilitzarem public static)
- **Tipus retornat:** El tipus de dada que retornarà la funció, com per exemple int, double, char, boolean, String, etc. Si la funció no retorna res s'indica mitjançant void
- **Nom de la funció:** Identificador únic per a cridar a la funció
- **Llista de paràmetres:** Indica els tipus i noms de les dades que se li passaran a la funció quan siga anomenada. Poden ser varis o cap

El **cos** és un bloc de codi entre claus { ... } que s'executarà quan des d'una altra part del codi cridem la funció

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static void imprimeixHolaMon() {  
    System.out.println("Hola mon");  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static void imprimeixHolaMon() {  
    System.out.println("Hola mon");  
}
```

Exemple 1: Aquest és un exemple molt senzill d'una funció anomenada 'imprimeixHolaMon', que no té paràmetres d'entrada (no hi ha res entre els parèntesis) i no retorna cap valor (indicat per *void*). Quan la cridem l'única cosa que farà serà escriure per pantalla el missatge "Hola mon".

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static void imprimeixHolaNom(String nom) {  
    System.out.println("Hola " + nom);  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static void imprimeixHolaNom(String nom) {  
    System.out.println("Hola " + nom);  
}
```

Exemple 2: Aquesta funció es diu 'imprimeixHolaNom', té com a paràmetre d'entrada una dada *String* anomenat 'nom' i no retorna res. Quan la cridem ens imprimirà per pantalla el text "Hola " seguit del *String* nom que li'l passarem com a paràmetre

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int doble(int a) {  
    int resultat = a * 2;  
    return resultat;  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int doble(int a) {  
    int resultat = a * 2;  
    return resultat;  
}
```

Exemple 3: Aquesta funció es diu 'doble', té com a paràmetre d'entrada una dada int anomenada 'a' i retorna una dada de tipus int. Quan la cridem calcularà el doble de 'a' i el retornarà (amb el return)

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int multiplica(int a, int b) {  
    int resultat = a * b;  
    return resultat;  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int multiplica(int a, int b) {  
    int resultat = a * b;  
    return resultat;  
}
```

Exemple 4: Aquesta funció es diu 'multiplica', té dos paràmetres d'entrada de tipus int anomenats 'a' i 'b' i retorna una dada de tipus int. Quan la cridem calcularà $a*b$ i ho retornarà (amb el return).

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static double maxim(double valor1, double valor2) {  
    double max;  
    if (valor1 > valor2)  
        max = valor1;  
    else  
        max = valor2;  
    return max;  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static double maxim(double valor1, double valor2) {  
    double max;  
    if (valor1 > valor2)  
        max = valor1;  
    else  
        max = valor2;  
    return max;  
}
```

Exemple 5: Aquesta funció es diu 'maxim, té dos paràmetres d'entrada de tipus double anomenats 'valor1' i 'valor2' i retorna una dada de tipus double. Quan la cridem calcularà el màxim entre 'valor1' i 'valor2' i ho retornarà

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int sumaVector(int v[]) {  
    int suma = 0;  
    for (int i = 0; i < v.length; i++)  
        suma += v[i];  
    return suma;  
}
```

Paràmetres d'entrada?

Què retorna?

Què fa la funció?

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int sumaVector(int v[]) {  
    int suma = 0;  
    for (int i = 0; i < v.length; i++)  
        suma += v[i];  
    return suma;  
}
```

Exemple 6: Aquesta funció es diu 'sumaVector', té un paràmetre d'entrada tipus int[] (un vector de int) anomenat 'v' i retorna una dada tipus int. Quan la cridem recorrerà el vector 'v', calcularà la suma de tots els seus elements i la retornarà

2. DECLARACIÓ D'UNA FUNCIO

EXEMPLES

```
public static int sumaVector(int v[]) {  
    int suma = 0;  
    for (int i = 0; i < v.length; i++)  
        suma += v[i];  
    return suma;  
}
```



For-each

https://www.w3schools.com/java/java_foreach_loop.asp

Exemple 6: Aquesta funció es diu 'sumaVector', té un paràmetre d'entrada tipus int[] (un vector de int) anomenat 'v' i retorna una dada tipus int. Quan la cridem recorrerà el vector 'v', calcularà la suma de tots els seus elements i la retornarà

2. DECLARACIÓ D'UNA FUNCIO

IMPORTANT

És important saber que les funcions es declaren dins de 'class' **PERÒ** fora del 'main'

```
1 package unidad7;
2
3 public class programadeprueba {
4
5     public static void imprimeHolaMundo() {
6         System.out.println("Hola mundo");
7     }
8
9     public static int doble(int a) {
10         int resultado = a * 2;
11         return resultado;
12     }
13
14     public static int multiplica(int a, int b) {
15         int resultado = a * b;
16         return resultado;
17     }
18
19     public static void main(String[] args) {
20
21         // Un programa siempre empieza ejecutándose por la función main.
22         // Aquí va el código principal de nuestro programa.
23     }
```

En aquest programa tenim 4 funcions: *imprimeHolaMundo*, *doble*, *multiplica* i *main*.

El 'main' on sempre has programant fins ara és en efecte una funció, però un poc especial: 'main' es la funció principal, el punt d'inici d'un programa

2. DECLARACIÓ D'UNA FUNCIO

IMPORTANT

És important saber que les funcions es declaren dins de 'class' **PERÒ** fora del 'main'

```
1 package unidad7;
2
3 public class programadeprueba {
4
5     public static void imprimeHolaMundo() {
6         System.out.println("Hola mundo");
7     }
8
9     public static int doble(int a) {
10         int resultado = a * 2;
11         return resultado;
12     }
13
14     public static int multiplica(int a, int b) {
15         int resultado = a * b;
16         return resultado;
17     }
18
19     public static void main(String[] args) {
20
21         // Un programa siempre empieza ejecutándose por la función main.
22         // Aquí va el código principal de nuestro programa.
23     }
```

És obligatori que tot programa Java tinga una funció *main*. Si et fixes, és una funció que rep com a paràmetre un *String[]* (vector de String) i no retorna res (encara que podria retornar un *int*). El per què d'això ho veurem més endavant

Les 3 funcions que hem declarat a dalt del main per si soles no fan res, simplement estan ahí esperant a ser anomenades (*utilitzades*), normalment des del propi *main*

FUNCIONS

ÍNDEX DE CONTINGUT

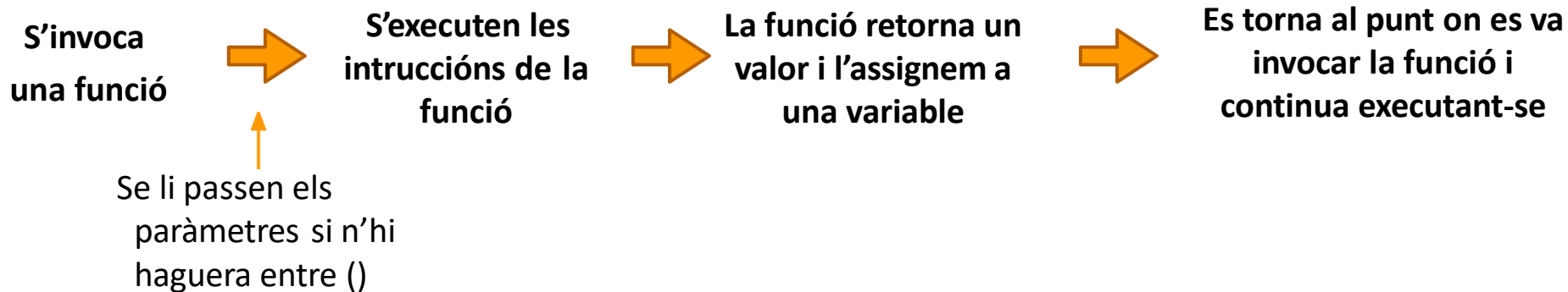
1. INTRODUCCIÓ
2. DECLARACIÓ D'UNA FUNCIO
3. **ANOMENA A UNA FUNCIO**
4. ÀMBIT DE LES VARIABLES
5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA
6. DEVOLUCIÓ D'UN VALOR

3. ANOMENAR A UNA FUNCIO

DEFINICIO

Les **funcions poden ser invocades** o anomenades **des de qualsevol altra funció, inclosa ella mateixa**
(*Ara com ara anomenarem funcions només des de la funció principal 'main'*)

Flux d'execució:



3. ANOMENAR A UNA FUNCIO

EXEMPLES

Exemples: utilitzant les funcions de l'apartat anterior

```
public static void main(String[] args) {
```

```
    // No té paràmetres ni retorna valor. Simplemente imprimeix "Hola Mon"
```

```
    imprimeixHolaMon();
```

```
    // És habitual cridar a una funció i guardar el valor retornat en una variable
```

```
    int a = doble(10); // a = 20 (10*2)
```

```
    int b = multiplica(3, 5); // b = 15 (3*5)
```

3. ANOMENAR A UNA FUNCIÓN

EXEMPLES

Exemples: utilitzant les funcions de l'apartat anterior

```
// Poden passar-se variables com a paràmetres
```

```
int c = doble(a); // c = 40 (20*2)
```

```
int d = multiplica(a, b); // d = 300 (20*15)
```

```
// Poden combinar-se funcions i expressions
```

```
int e = doble(4) + multiplica(2,10); // e = 8 + 20
```

```
System.out.println("El doble de 35 és " + doble(35) ); // "El doble de 35 és 70"
```

```
System.out.println("12 per 12 és " + multiplica(12,12) ); // "12 per 12 és 144"
```

3. ANOMENAR A UNA FUNCIO

EXEMPLES

Exemples: Programa amb una funció que suma dos nombres

DIY

3. ANOMENAR A UNA FUNCIÓN

EXEMPLES

Exemples: Programa amb una funció que suma dos nombres

```
4 public class Suma {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         int num1, num2, suma;
9
10        System.out.print("Introduce un número: ");
11        num1 = sc.nextInt();
12
13        System.out.print("Introduce otro número: ");
14        num2 = sc.nextInt();
15
16        suma = suma(num1, num2);
17
18        System.out.println("La suma es: " + suma);
19    }
20
21    public static int suma(int n1, int n2) {
22        int suma;
23
24        suma = n1 + n2;
25
26        return suma;
27    }
28 }
29
30
31 }
```

3. ANOMENAR A UNA FUNCIO

EXEMPLES

Exemples: Programa amb una funció que determina si un nombre és parell o imparell



DIY

3. ANOMENAR A UNA FUNCIÓN

EXEMPLES

```
14 public class ParImpar {
15
16     public static void main(String[] args) {
17         Scanner in = new Scanner(System.in);
18         int num;
19
20         System.out.print("Introduce un número: ");
21         num = in.nextInt();
22
23         if(par(num) == true) // Llamada a la función desde la expresión
24             System.out.println(num + " es par.");
25         else
26             System.out.println(num + " es impar.");
27     }
28
29     public static boolean par(int numero)
30     {
31         boolean par = false;
32
33         if(numero % 2 == 0) // Si el resto es 0 par será 'true' sino 'false'
34             par = true;
35
36         return par;
37     }
38 }
```

3. ANOMENAR A UNA FUNCIÓ *DEBUGGING*



<https://en.wikipedia.org/wiki/Debugging>

3. ANOMENAR A UNA FUNCIÓN DEBUGGING

```
14 public class ParImpar {
15
16     public static void main(String[] args) {
17         Scanner in = new Scanner(System.in);
18         int num;
19
20         System.out.print("Introduce un número: ");
21         num = in.nextInt();
22
23         if(par(num) == true) // Llamada a la función desde la expresión
24             System.out.println(num + " es par.");
25         else
26             System.out.println(num + " es impar.");
27     }
28
29     public static boolean par(int numero)
30     {
31         boolean par = false;
32
33         if(numero % 2 == 0) // Si el resto es 0 par será 'true' sino 'false'
34             par = true;
35
36         return par;
37     }
38 }
```

Fem *debugging* de
l'exemple anterior

FUNCIONS

ÍNDEX DE CONTINGUT

- 1. INTRODUCCIÓ**
- 2. DECLARACIÓ D'UNA FUNCIO**
- 3. ANOMENA A UNA FUNCIO**
- 4. ÀMBIT DE LES VARIABLES**
- 5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA**
- 6. DEVOLUCIÓ D'UN VALOR**

4. ÀMBIT DE LES VARIABLES

DEFINICIÓ



ENCAPSULACIÓ

Una funció només pot utilitzar les variables d'àmbit local, és a dir, les seues pròpies variables (els paràmetres de la capçalera i les variables creades dins de la funció). Quan una funció s'executa es creen les seues variables, s'utilitzen i quan la funció acaba es destrueixen les variables

Per tot això **una funció no pot utilitzar variables que estiguen fora d'ella**, i fora d'una funció no és possible utilitzar variables de la pròpia funció. A aquesta característica se'n diu **encapsulació** i permet que les funcions siguin independents entre si, facilitant el disseny de programes grans i complexos

Tècnicament sí que és possible que una funció utilitze variables que estan fora d'ella, però això ho veurem en futures unitats quan aprenguem Programació Orientada a Objectes

FUNCIONS

ÍNDEX DE CONTINGUT

- 1. INTRODUCCIÓ**
- 2. DECLARACIÓ D'UNA FUNCIO**
- 3. ANOMENA A UNA FUNCIO**
- 4. ÀMBIT DE LES VARIABLES**
- 5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA**
- 6. DEVOLUCIÓ D'UN VALOR**

5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

2 tipus de paràmetres



- **Paràmetres de tipus simple** (*pas per valor*)
- **Paràmetres de tipus objecte** (*pas per referència*)

5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

- **Paràmetres de tipus simple (pas per valor):** Com *int*, *double*, *boolean*, *char*, etc. En aquest cas **es passen per valor**. És a dir, **el valor es copia al paràmetre** i per tant si es modifica dins de la funció això no afectarà el valor fora d'ella perquè **són variables diferents**

```
public static void main(String[] args) {  
  
    int a = 10;  
    System.out.println("Valor inicial de a: " + a);  
    imprimeix_doble(a);  
    System.out.println("Valor final de a: " + a);  
}
```

EIXIDA ??

```
public static void imprimeix_doble(int a) {  
    a = 2 * a;  
    System.out.println("Valor de a en la funció: " + a);  
}
```

5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

- **Paràmetres de tipus simple (pas per valor):** Com *int*, *double*, *boolean*, *char*, etc. En aquest cas **es passen per valor**. És a dir, **el valor es copia al paràmetre** i per tant si es modifica dins de la funció això no afectarà el valor fora d'ella perquè **són variables diferents**

```
public static void main(String[] args) {  
  
    int a = 10;  
    System.out.println("Valor inicial de a: " + a);  
    imprimeix_doble(a);  
    System.out.println("Valor final de a: " + a);  
}
```

```
run:  
Valor inicial de a: 10  
Valor de a en la funció: 20  
Valor final de a: 10
```

```
public static void imprimeix_doble(int a) {  
    a = 2 * a;  
    System.out.println("Valor de a en la funció: " + a);  
}
```

5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

- **Paràmetres de tipus objecte (pas per referències)** : Com a objectes de tipus *String*, els *Arrays*, etc. En aquest cas no es copia l'objecte sinó que **se li passa a la funció una referència a l'objecte original (un punter)**. Per això **des de la funció s'accedeix directament a l'objecte** que es troba fora. Els canvis que fem dins de la funció afectaran l'objecte

```
// Summa x a tots els elements del vector v
public static void summa_x_al_vector(int v[], int x) {
    for (int i = 0; i < v.length; i++)
        v[i] = v[i] + x;
}
```

EIXIDA ??

```
public static void main(String[] args) {
    int v[] = {0, 1, 2, 3};
    System.out.println("Vector abans: " + Arrays.toString(v));
    summa_x_al_vector(v, 10);
    System.out.println("Vector després: " + Arrays.toString(v));
}
```

5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

- **Paràmetres de tipus objecte (pas per referències)** : Com a objectes de tipus *String*, els *Arrays*, etc. En aquest cas no es copia l'objecte sinó que **se li passa a la funció una referència a l'objecte original (un punter)**. Per això **des de la funció s'accedeix directament a l'objecte** que es troba fora. Els canvis que fem dins de la funció afectaran l'objecte

```
// Summa x a tots els elements del vector v
public static void summa_x_al_vector(int v[], int x) {
    for (int i = 0; i < v.length; i++)
        v[i] = v[i] + x;
}

public static void main(String[] args) {
    int v[] = {0, 1, 2, 3};
    System.out.println("Vector abans: " + Arrays.toString(v));
    summa_x_al_vector(v, 10);
    System.out.println("Vector després: " + Arrays.toString(v));
}
```

```
run:
Vector antes: [0, 1, 2, 3]
Vector después: [10, 11, 12, 13]
```


5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA

DEFINICIÓ

IMPORTANT: Com un paràmetre de tipus objecte és una referència a l'objecte *String* o *Array* que està fora d'ella, si se li assigna un altre objecte es perd la referència i ja no es pot accedir a l'objecte fora de la funció. Encara que Java permet fer-ho, no s'aconsella fer-ho

```
public static void funcion1(int x[]) {  
  
    x = new int[10]; // x apuntarà a un nou vector, l'original queda intacte  
    ...              // el que fem ací amb x no afectarà el vector original  
}                   // S'ha perdut la referència a aquest objecte
```

NO S'ACONSELLA FER ESTE TIPUS DE COSES

FUNCIONS

ÍNDEX DE CONTINGUT

- 1. INTRODUCCIÓ**
- 2. DECLARACIÓ D'UNA FUNCIO**
- 3. ANOMENA A UNA FUNCIO**
- 4. ÀMBIT DE LES VARIABLES**
- 5. PARÀMETRES: PAS PER VALOR I PER REFERÈNCIA**
- 6. DEVOLUCIÓ D'UN VALOR**

6. DEVOLUCIÓ D'UN VALOR

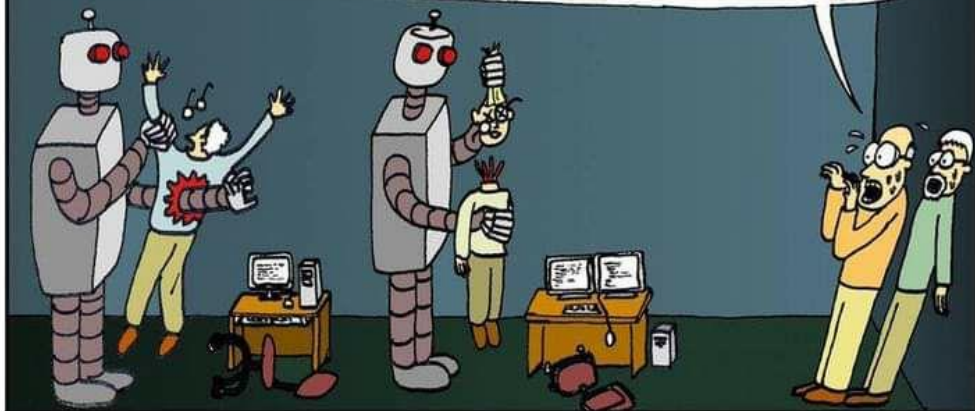
DEFINICIÓ

Comando ***return*** ➡ Devolució d'un valor

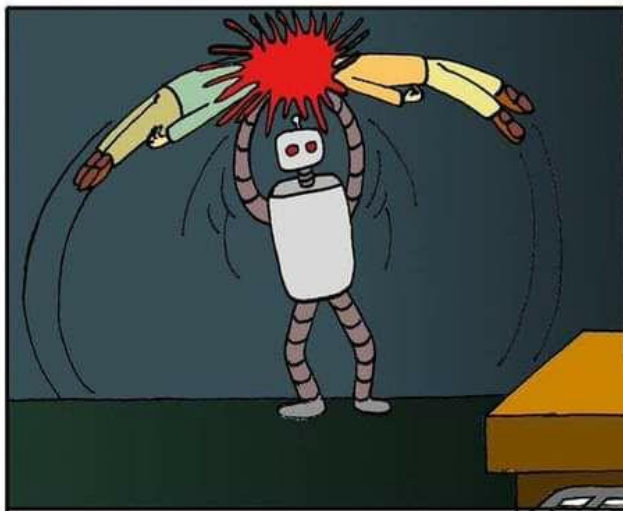
En el cas d'arrays i objectes, retorna una referència a aqueix array o objecte

Els mètodes poden **retornar valors de tipus bàsic o primitiu** (*int, double, boolean, etc.*) i **també de tipus objecte** (*Strings, arrays, etc.*)

OH NO! THE ROBOTS ARE KILLING US!!!



BUT WHY?!? WE NEVER PROGRAMMED THEM TO DO THIS!!!



```
static bool isCrazyMurderingRobot = false;
```

```
void interact_with_humans (void){  
    if(isCrazyMurderingRobot = true)  
        kill(humans);  
    else  
        be_nice_to(humans);  
}
```

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