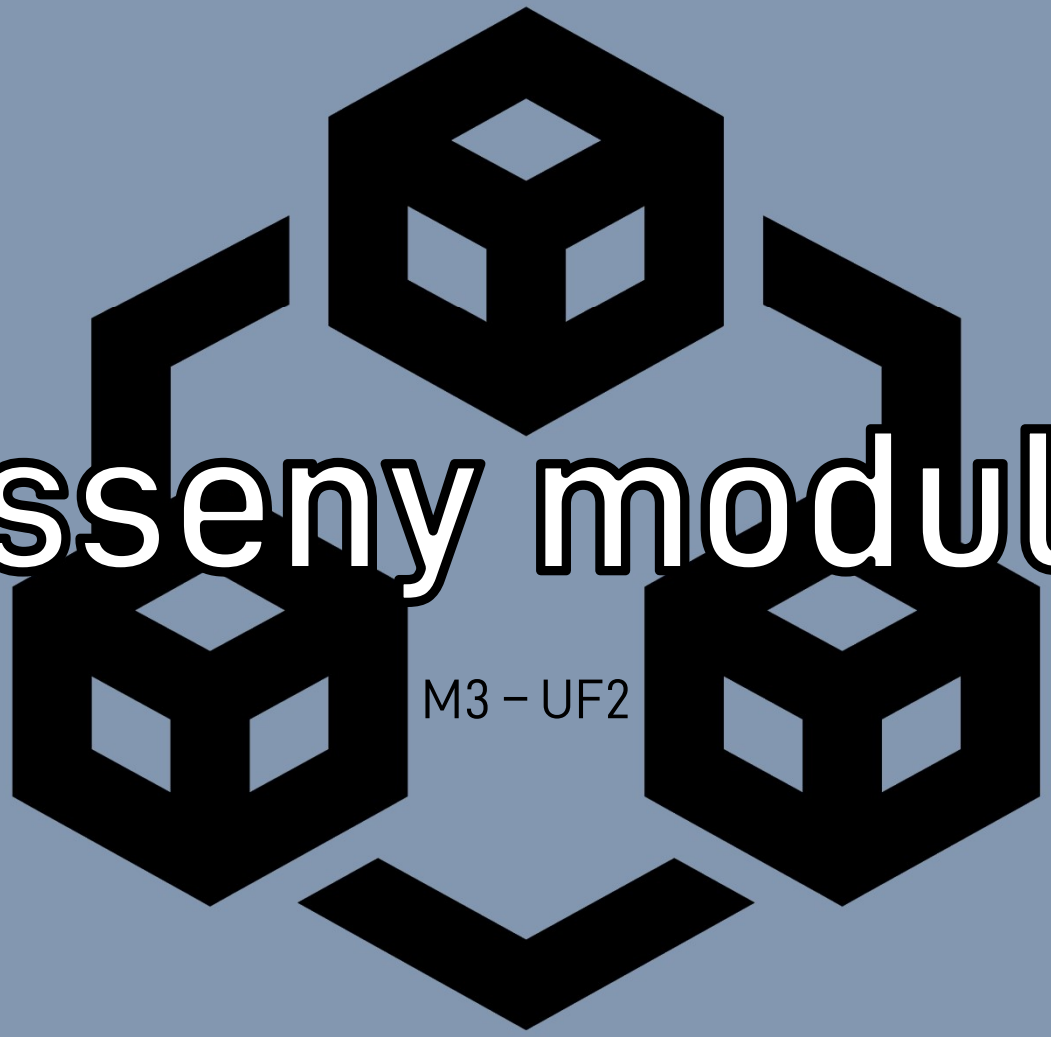
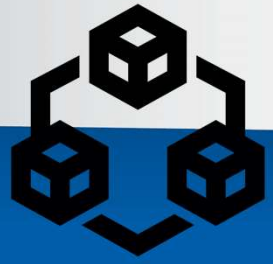


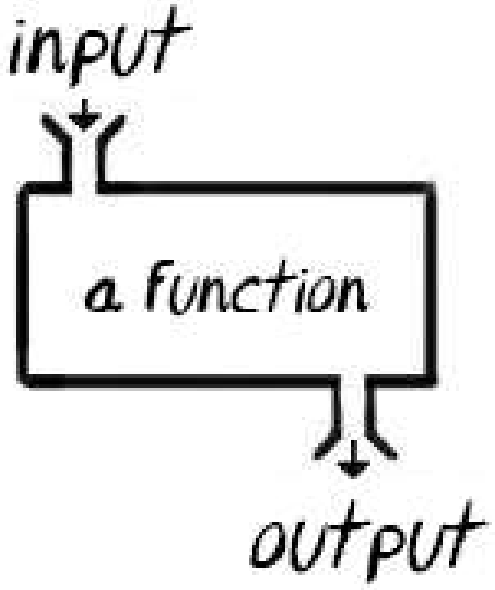
Disseny modular

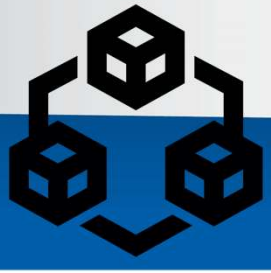
M3 – UF2



Què és un mètode (o funció)?

- És un **bloc de codi** que es defineix fora del codi principal
- S'executa solament **quan es crida** *(excepte* `main`*)*
- S'usa per **executar certes accions**
- Permet definir un codi una vegada i **reutilitzar-lo** tantes com vulguem
- Se li poden passar dades anomenades **paràmetres**
- Pot **retornar un cert valor**, o no
- **Reaprofita, millora i simplifica** l'estructura del codi





Format d'un mètode

Ja coneixem un mètode, el que es crida quan s'executa una classe:

```
public static void main(String[] args){
    ...<cos_del_mètode>...
}
```

TIPUS D'ACCÉS
(opcional)
Pot ser `public`, `private` o `protected`

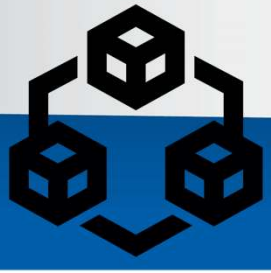
	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	
no modifier	+	+	+		
private	+				

MODIFICADOR
(opcional)
Indica que el mètode pertany a la classe i no a l'objecte

TIPUS RETORNAT
Tipus de dades (primitiu o no) que retorna el mètode (`void` si no retorna res)

NOM
Nom del mètode amb el qual l'identificarem i el cridarem.
El mètode amb nom `main` s'executa automàticament a una classe pública.

PARÀMETRES
Paràmetres que li passem al mètode (`tipus nom`).
També pot estar buit o tenir-ne més d'un (separats per comes)



Creació d'un mètode

Per a crear un mètode o funció, haurem de fer-ho dintre d'una classe determinada.

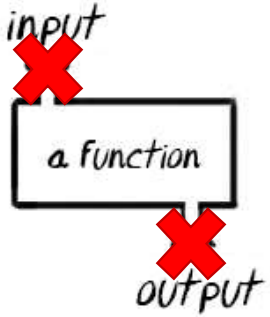
Crearem una funció que simplement imprimeix una salutació per la consola:

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

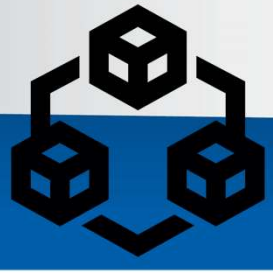
Cridar a un mètode

Per cridar a un mètode o funció, simplement haurem de posar el seu nom al codi principal seguit de parèntesis (buits si no té paràmetres)

```
saludar();
```



Mostrar alguna cosa per consola no es considera que sigui una sortida d'un mètode o funció



Mètode amb paràmetres

Per afegir paràmetres a un mètode o funció, els haurem de definir dintre dels parèntesi, posant el tipus de dades i el nom amb què s'anomenarà aquest paràmetre dintre del mètode:

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

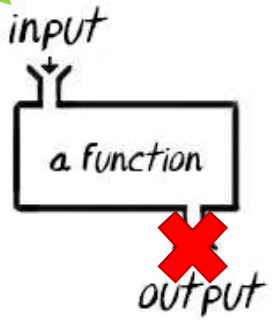
Cridar a un mètode amb paràmetres

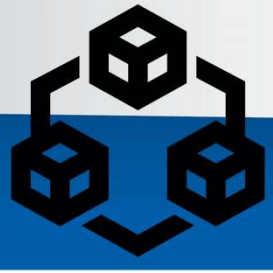
Per cridar a un mètode o funció amb paràmetres, haurem de posar el seu nom al codi principal seguit de parèntesis i el valor del paràmetre dintre dels parèntesis:

```
saludarPersona("Eduard");
```

o també:

```
String pers = "Anton";
saludarPersona(pers);
```

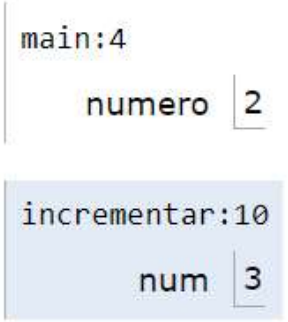




Consideracions al passar paràmetres (I)

- Si el paràmetre d'un mètode és de tipus primitiu o String, el paràmetre fora del mètode no es veurà afectat, ja que es fa una còpia diferent de la variable dintre del mètode*.

```
public class VariablePrimitiva {
  public static void main(String[] args) {
    int numero = 2;
    incrementar(numero);
  }
  static void incrementar(int num) {
    num++;
  }
}
```

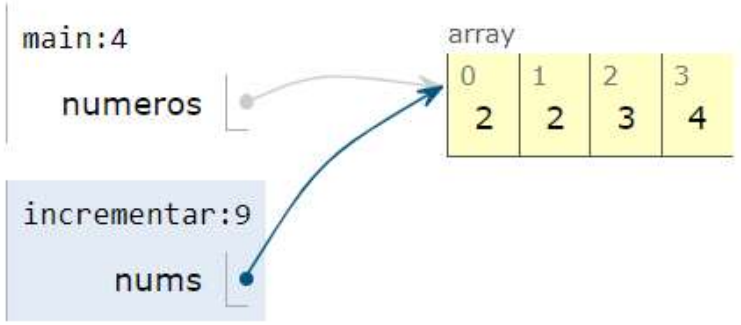


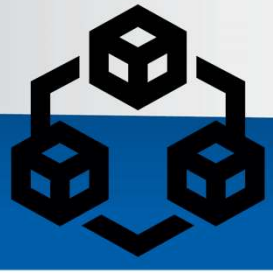
**Les variables que es creen dintre d'un mètode s'esborren una vegada el mètode acaba.*

Consideracions al passar paràmetres (II)

- Si el paràmetre és de tipus complex (un objecte, un array, una matriu,...) aquest sí que es veurà afectat fora del mètode, ja que no es fa una còpia del paràmetre, sinó que es crea una referència al mateix.

```
public class VariableObjecte {
  public static void main(String[] args) {
    int[] numeros = {1,2,3,4};
    incrementar(numeros);
  }
  static void incrementar(int[] nums) {
      nums[0]++;
  }
}
```





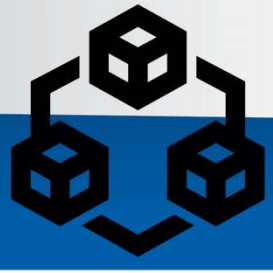
Mètodes que retornen dades (I)

Quan el nostre mètode o funció retorna alguna dada, haurem de tenir en compte dues coses:

- Posar el **tipus de dada que retorna** quan definim el mètode
- Posar la paraula **return** seguit del valor a retornar

```
public static boolean esParell(int num) {

      boolean resultat;
      if (num%2 == 0){
            resultat = true;
      } else {
            resultat = false;
      }
      return resultat;
}
```



Mètodes que retornen dades (II)

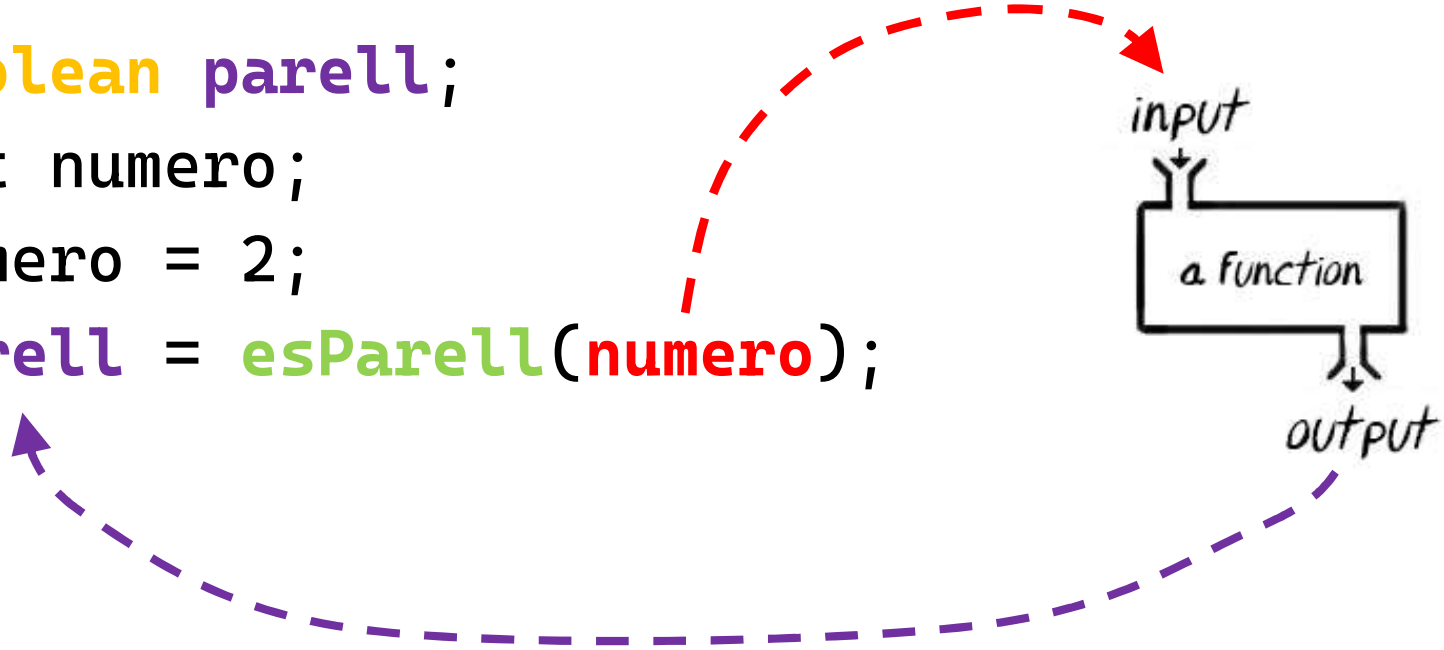
Tingueu en compte que una vegada fem un return, el mètode finalitzarà, encara que hi hagi més codi per executar dintre d'aquest:

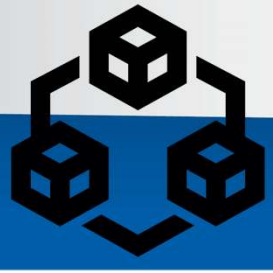
```
public static boolean esParell(int num) {
     if (num%2 == 0){
          return true;
     } else {
          return false;
     }
}
```

Crida a mètodes que retornen dades

Quan cridem a un mètode que retorna un tipus de dades, haurem de fer l'assignació a una variable del tipus corresponent al fer la crida:

```
boolean parell;
int numero;
numero = 2;
parell = esParell(numero);
```



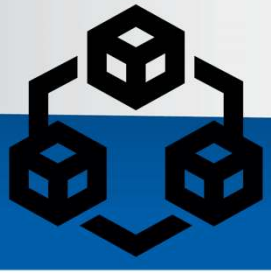


Sobrecàrrega de mètodes

Diversos mètodes poden tenir el mateix nom si tenen diferent tipus i/o número de paràmetres (tenen diferent ***signature****)

```
int    elMeuMetode(int x)
float   elMeuMetode(float x)
int    elMeuMetode(int x, int y)
```

* *signature = nom + llista de paràmetres*



Ús de mètodes ja creats (llibreries)

Hi ha mètodes ja creats i que es poden usar directament important les seves llibreries (fitxers `*.jar`).

Alguns d'aquests mètodes poden ser:

```
public void println (String x) //Sobrecarregat
public static double random ()
public static double sqrt (double a)
public char charAt (int index)
```