UD 3.2- ESTRUCTURES DE CONTROL DE FLUX: REPETICIÓ

0485 Programació

1er DAW/DAM

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

CPFP BATOI

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

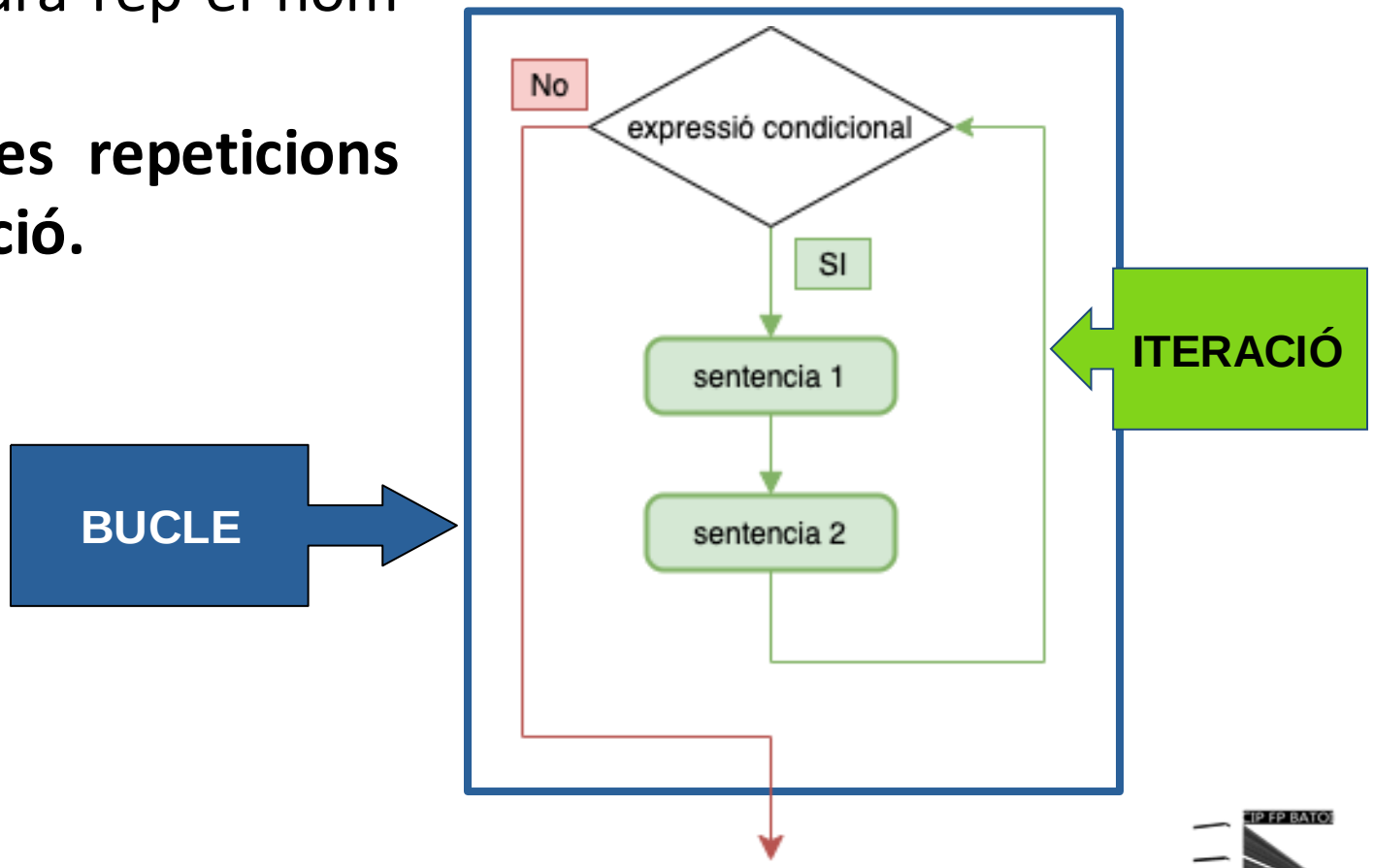
UNIÓ EUROPEA

0. CONTINGUTS

- **ESTRUCTURES DE REPETICIÓ** (bucles)
  - `while ...`
  - `do ... while`
  - `for ...`
- **SENTÈNCIES** `break` i `continue`

1. ESTRUCTURES DE REPETICIÓ

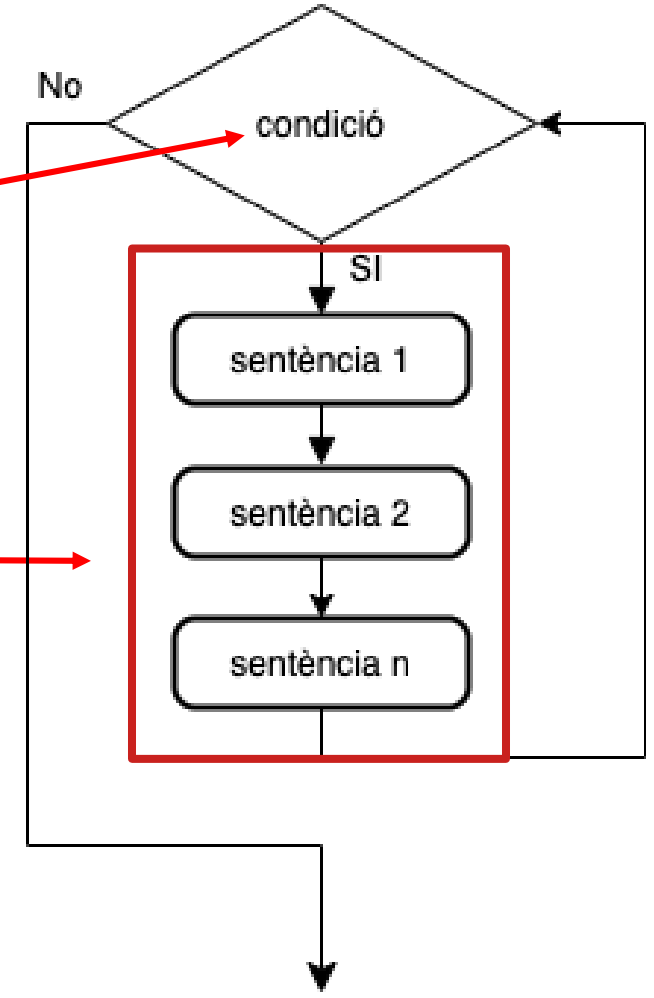
- Permeten **executar 1 o diverses instruccions**, de forma repetida, **mentre** es **complisca** una **condició.**
  - Aquesta estructura rep el nom de **bucle**
  - **Cadascuna de les repeticions** s'anomena **iteració.**



1.1 ESTRUCTURA `while` (I)

Tipus de dades expressió booleana

```
while (condició) {
    sentència 1;
    sentència 2;
    ...
    sentència n;
}
```



1.1 ESTRUCTURA `while` (II)

- **El codi del bucle** sols s'executarà si la **condició s'avalua a `true`** i **es repeteix fins** que la **expressió** siga **`false`**.
  - La condició de finalització es comprova al **principi de cada iteració**.

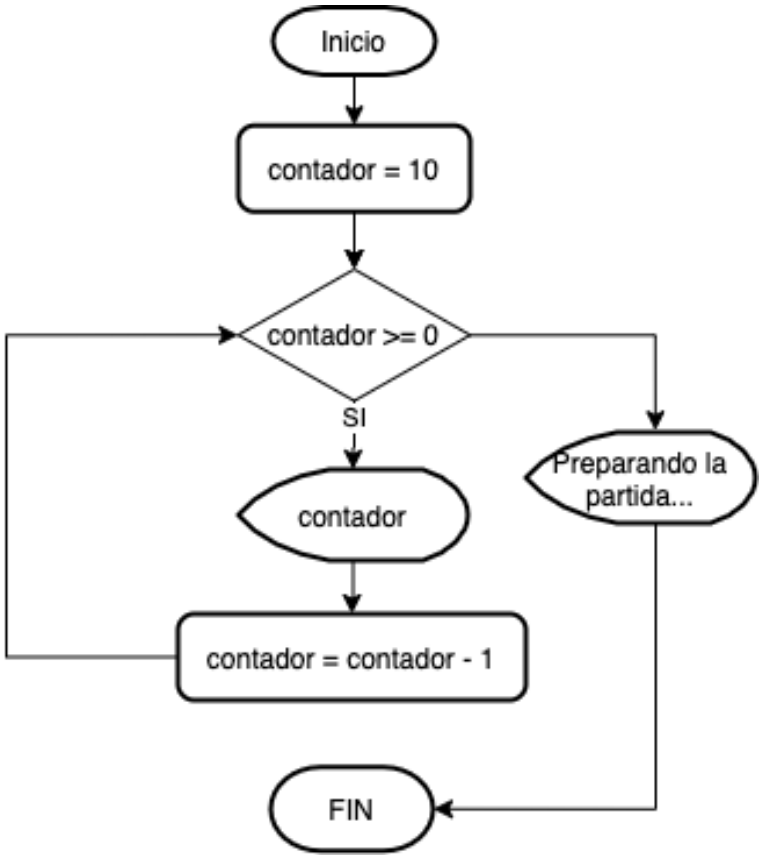
```
public static void main(String[] args) {
    int comptador = 0;
    while (comptador < 10) {
        System.out.println (comptador);
        comptador++;
    }
}
```

Si la **CONDICIÓ NO ES COMPLEIX**, El codi **NO S'EXECUTARÀ MAI**

Ens hem d'assegurar
Que hi haga una **CONDICIÓ** de **FINALITZACIÓ**

ACTIVITAT PRÈVIA

- **Activitat 14.** Implementa un programa que mostre per pantalla un comptador descendent que vaja des del 10 al 0. Una vegada acabe ha de mostrar el text **"Preparant la partida ..."**



ACTIVITATS PRÈVIES

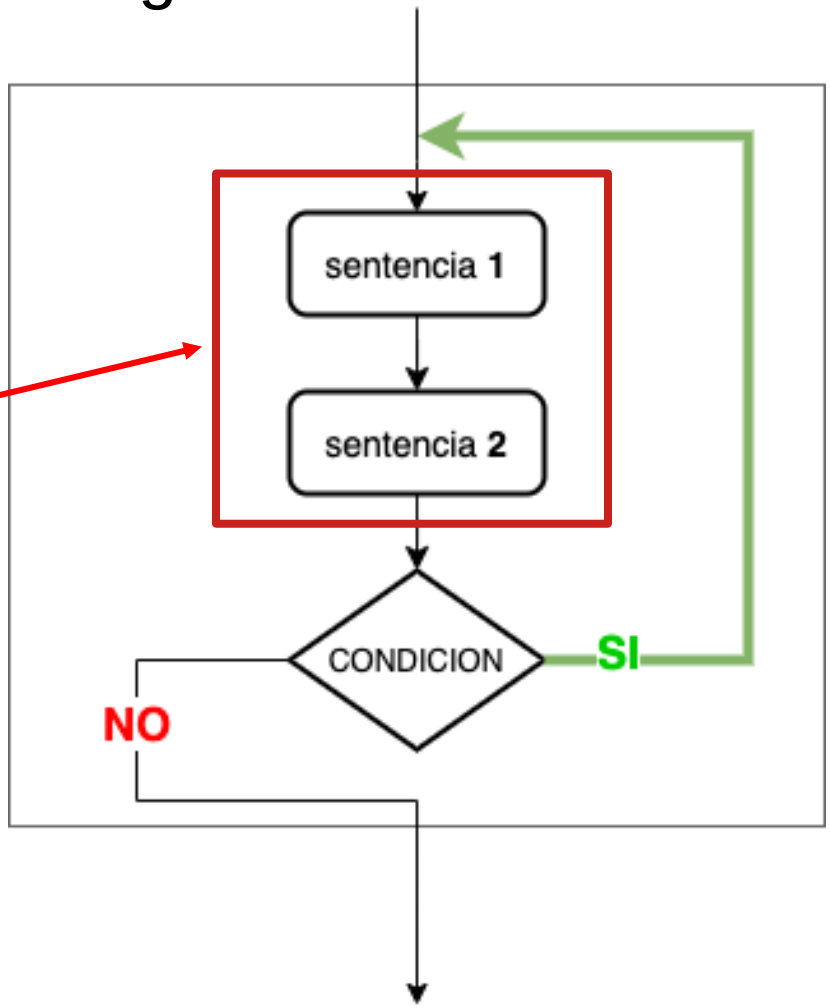
- **Activitat 15.** Implementa un programa que, utilitzant una **estructura while** mostre la següent sèrie de números **48, 50, 52, ... 100**

- **Activitat 16.** Implementa un programa que mostre el resultat de **sumar** els números compresos **entre 1 i 1000.** (*1 + 2 + 3 + 4 + 5 + 6 + ... + 1000)*

1.2 ESTRUCTURA `do...while` (I)

- **Executa el codi del bucle** i **després avalua la condició**. **Repeteix fins** que la **condició** sigui **`false`**.

```
do {
    sentència 1;
    sentència 2;
    ...
} while (Condició);
```



1.2 ESTRUCTURA do...while (II)

- Aquesta **estructura** ens **asegura** que les **instruccions del bucle s'executaran almenys una vegada**

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

    Scanner teclat = new Scanner (System.in);
    // demanar una nota fins que sigui major que 5
    double nota;
    do {
        System.out.println ("Introdueix una nota");
        nota = teclat.nextDouble();
    } while (nota < 5.0);

}
```

1.3 ESTRUCTURA while vs do...while

- Fent les adaptacions oportunes, qualsevol bucle `while` pot ser **sempre** representat com a un bucle `do-while` i viceversa.
- Depenent del programa a implementar pot resultar més adequat fer servir un que l'altre.

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

    Scanner teclat = new Scanner (System.in);

    double nota;
    System.out.println ("Introdueix una nota");
    nota = teclat.nextDouble();
    while (nota < 5.0) {
        System.out.println ("Introdueix una nota");
        nota = teclat.nextDouble();
    }
}
```

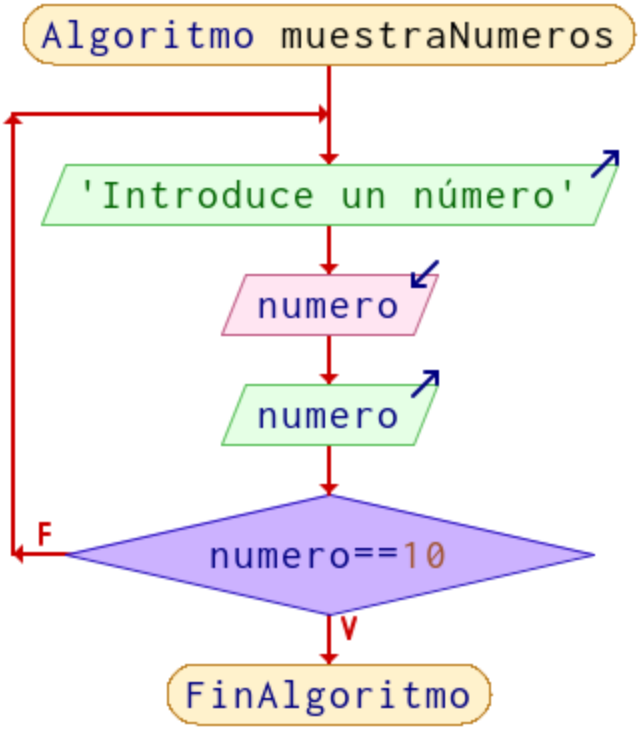
```
public static void main(String[] args) {

    Scanner teclat = new Scanner (System.in);

    double nota;
    do {
        System.out.println ("Introdueix una
nota");
        nota = teclat.nextDouble();
    } while (nota < 5.0);
}
```

ACTIVITAT PRÈVIA

- **Activitat 17.** Implementa un programa que demane números a l'usuari de forma indefinida i els mostre per pantalla mentre el número introduït siga diferent de 10.



ACTIVITAT PRÈVIA

Activitat 18. Implementa un programa que demane a l'usuari **un número** enter **entre 1 i 10**. Seguidament ha de mostrar per pantalla la seua taula de multiplicar. Utilitza l'**estructura do-while**.

```
Escriu un enter: 6
Taula del 6
-----------
6 * 1 = 6
6 * 2 = 12
6 * 3 = 6
    .
    .
6 * 10 = 60
```

1.4 Característiques comuns

- Les **estructures iteratives** disposen de:
  - Una **variable de control** que permet definir quan acabarà l'execució del bucle.
    - **Comptador:** `while (comptador < 10) {...}`
    - **Semàfor:** `while (demanarMes) {...}`
  - Una **operació d'actualització** sobre el valor de la variable.
    - **comptador:** `comptador++`; *(pot ser més d'una unitat de increment o decrement)*
    - **semàfor:** `demanarMes = false`;

1.4 Característiques comuns

```java
public static void main(String[] args) {
    int comptador = 0;
    while (comptador < 10) {
        System.out.println (comptador);
        comptador++;
    }
}
```

Actualització de variable comptadora

Variable comptadora

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

    Scanner teclat = new Scanner(System.in);
    boolean dadaCorrecta = false;
    do{
        int numero = teclat.nextInt();
        if (numero > 0) {
            dadaCorrecta = true;
        }
    }while(!dadaCorrecta);
}
```

Variable semàfor

Actualització de la variable semàfor

1.5 ESTRUCTURA for (I)

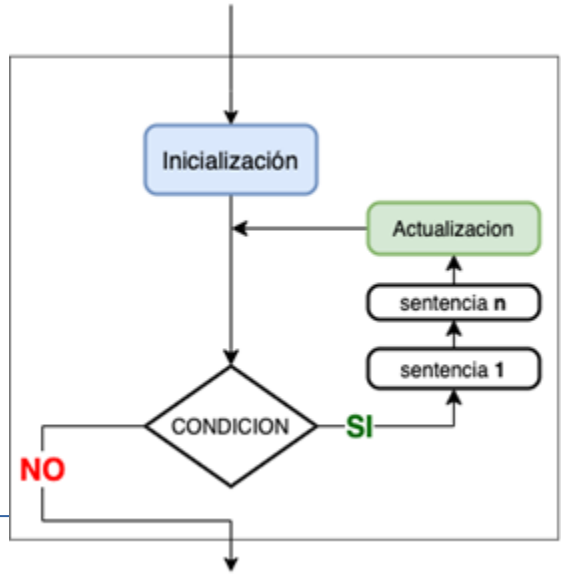
- S'utilitza quan **coneixem per endavant** el **nombre de vegades què volem** que **es repetisca l'execució** d'un **bloc de sentències.**

1. s'executa 1 vegada al **iniciar el bucle**

2. S'executa al **començament** de **cada iteració**

3. S'executa al **finalitzar cada iteració**

```
for (inicialització; condició; actualització) {
        sentència 1;
        ...
        sentència n;
}
```

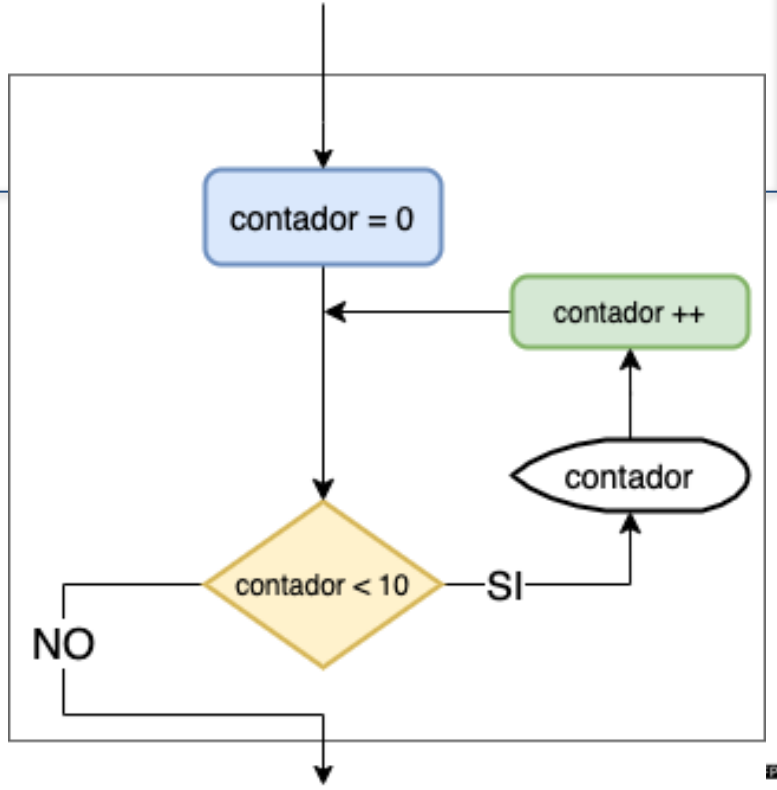


1.5 ESTRUCTURA `for` (II)

- **Inicialització**: Inicialitza una (o més d'una) **variable de tipus sencer** què servirà com a comptador.
- **Condició:** Expressió booleana que indica si s'ha de fer una nova iteració o no.
- **Actualització**: Aquesta instrucció (o instruccions) s'executen **a la fi de cada iteració**. El més habitual és que aquesta instrucció incremente o decremente el comptador (o comptadors).

1.4 ESTRUCTURA for (III)

```
public static void main(String[] args) {
    for (int comptador = 0; comptador <10 ; comptador ++) {
        System.out.println (comptador);
    }
}
```



1.4 ESTRUCTURA `for` (IV)

- Habitualment els bucles "**for**" utilitzen un **comptador** de **àmbit local** al cos del bucle. En aquest cas:
  - **No** es pot emprar **fora** del **bucle**.
  - **Els noms** de **variables comuns** per als **comptadors** són: ***i, j, k, l, m, n***
  - Això no vol dir que no siga possible **nombrar-les diferent en cassos puntuals**

```
public static void main(String[] args) {
    for (int i = 0; i <10 ; i++) {
        System.out.println (i);
    }
}
```

```
public static void main(String[] args) {
    for (int j = 0; j <10 ; j ++) {
        System.out.println (j);
    }
}
```

L'àmbit d'aquestes variables vindrà determinada pel propi bucle `for`

1.4 ESTRUCTURA `for` (V)

- No obstant això, les variables comptadores poden ser declarades fora del bucle i emprades dins, encara que **es recomana sempre declarar-la dins (en el nostre cas la recomanació serà una obligació)**.
- En aquest cas:

```
public static void main(String[] args) {
    int i;
    for (i = 0; i <10 ; i++) {
        System.out.print(i);
    }
    i--;
    System.out.print(i);
}
```

L'àmbit d'aquesta variable vindrà determinada, en aquest cas, pel mètode `main`

- Mostrarà per pantalla:

```
01234567899
```

ACTIVITATS PRÈVIES

- **Activitat 19.** Refactoritza la **activitat 14** perquè mostre el llistat de números fent ús de l'**estructura for**
- **Activitat 20.** Refactoritza l'**activitat 18** perquè la taula de multiplicar es mostre fent ús de l'**estructura for.**
- **Activitat 21.** Escriu un programa que mostre tots els números parells que hi ha de l'1 al 100 i a la fi indique quants n'hi ha. Repeteix-ho per als imparells i múltiples de 5

1.4 ESTRUCTURA `for` (VI)

- Molt sovint, es fa necessària l'utilització de **bucles `for` dins d'altres** per a poder solventar un problema. Aquestes estructures reben el nom de ***for* anidats**.

```
for (int i = 0; i < 24 ; i++) {
    for (int j = 0; j < 60 ; j++) {
        for (int k = 0; k < 60; k++) {
            System.out.printf(" %d hores %d minuts %d segons", i, j, k);
            System.out.print("\r");
            Thread.sleep(1000);
        }
    }
}
```

Hores

Minuts

Segons

Aquest bloc de codi **s'executarà 24x60x60** voltes, és a dir, **86400** voltes que són els **segons que té un día**

Mostrarà per pantalla **un rellotge:**

`0 hores 0 minuts 0 segons` → `23 hores 59 minuts 59 segons`

ACTIVITATS PRÈVIES

- **Activitat 22.-** Escriu un programa que mostre les taules de multiplicar de l'1 al 10.

2. SENTÈNCIES *break* i continue

- Quan un **bucle està en execució**, Java disposa de dues sentències **per canviar** el seu ***comportament per defecte***:
  - **`break`**: Provoca l'acabament del bucle.
  - **`continue`**: Provoca el començament d'una nova repetició. Avorta la passada actual.

2. SENTÈNCIES *break* i continue

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

    for (int i = 0; i <100 ; i ++) {
        if (i% 2 ! = 0) {
            continue;
        }
        System.out.println (i);
    }
}
```

mostrarà només els **números parells**

2. SENTÈNCIES *break* i continue

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

    for (int i = 1; i <100 ; i ++) {
        if (i % 2 ! = 0) {
            break;
        }
        System.out.println (i);
    }
}
```

No s'executarà mai

2.1 ACTIVITATS PRÈVIES

- **Activitat 23.** Implementa un programa que **demane a l'usuari 50 nombres enters** i imprimisca per pantalla **si són parells, imparells o múltiples de 5**.

  a) Si l'usuari **introdueix un 100** no es mostrarà res per pantalla i **passarà a la següent iteració** del bucle.

  b) Si l'usuari **introdueix un 0 finalitzarà** l'execució de l' **bucle**.

Nota: has d'utilitzar** les sentències **break** i **continue** per implementar els **requeriments a i b

2.1 ACTIVITATS PRÈVIES

Activitat 24. *Escriu un programa que simule el llançament d'una moneda mitjançant la generació d'un* ***nombre aleatori entre 1 (cara) i 2 (creu):***

El programa ha de fer veure que es llança una moneda un nombre 1.000.000 de vegades. A continuació ha d'imprimir per pantalla el nombre de cares i el nombre de creus que han sortit.

Exemple d'execució

```
Nombre de tirades: 1.000.000
----------------------------
Nombre de cares: 500.627
Nombre de creus: 499.373
```

1.4 ACTIVITAT PRÈVIA

- Això és tot ... de moment :-)