

0101010  
0100101  
1101010

# UD 3.1- ESTRUCTURES DE CONTROL DE FLUX: DECISIONS

0485 Programació  
1er DAW/DAM

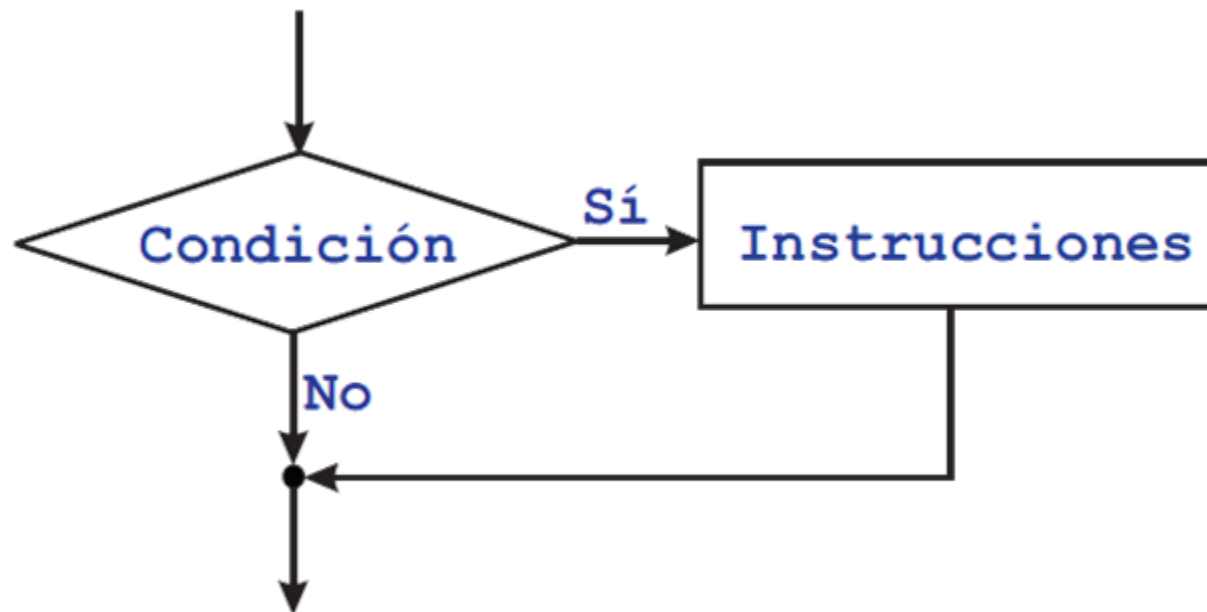
# 0. CONTINGUTS

---

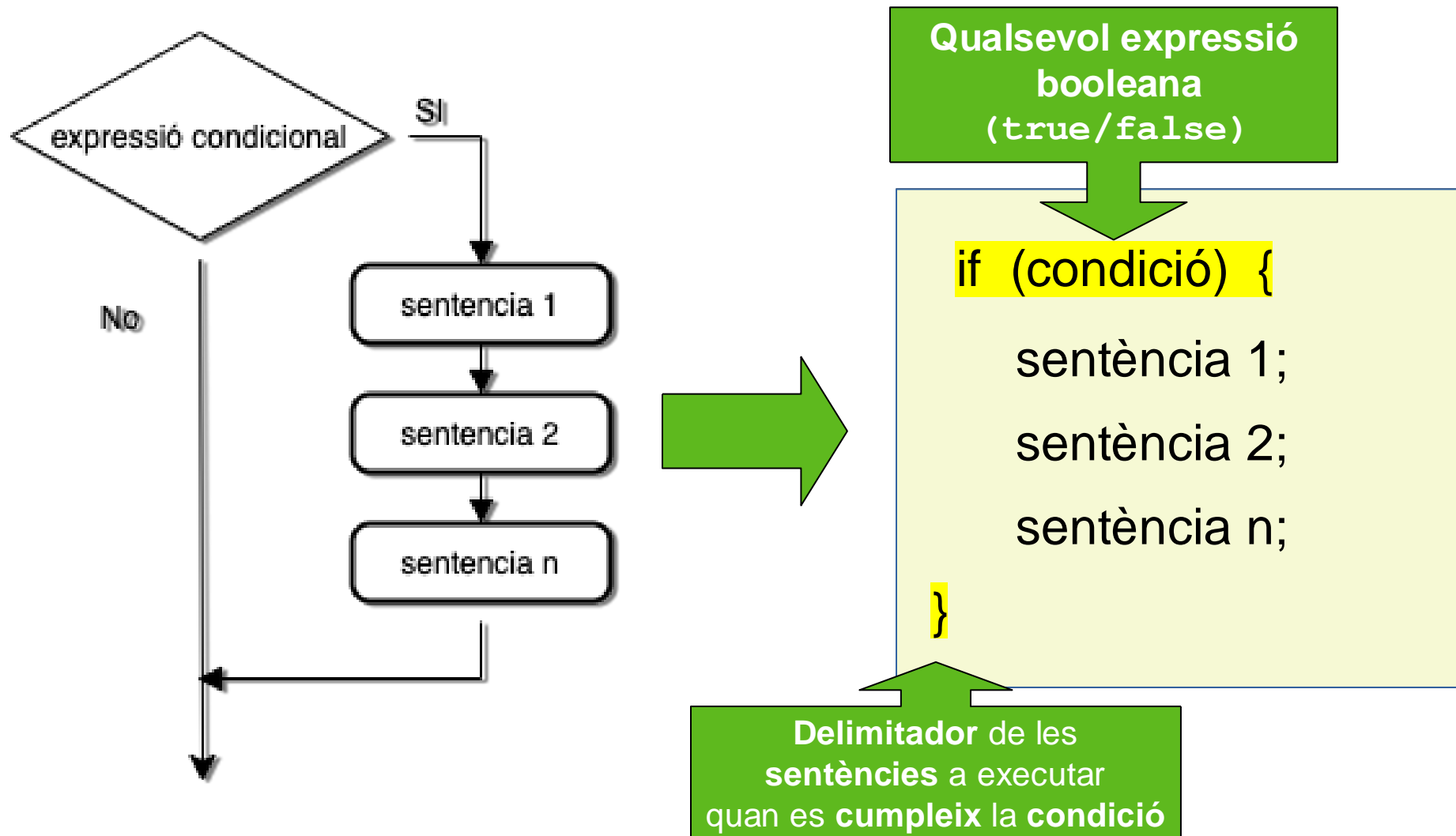
- EXPRESSIONS CONDICIONALS
- ESTRUCTURA `if...`
- ESTRUCTURA `if...else`
- ESTRUCTURA `if...else if`
- ESTRUCTURA `switch...case`
- OPERADOR TERNARI `?:`

# 1. EXPRESSIONS CONDITIONALS

- Permeten **executar una instrucció o conjunt d'instruccions** en funció de la **avaluació d'un predicat**.



# 1.1 ESTRUCTURA `if...`



# 1.1 ESTRUCTURA `if . . .`

- Si (condició és **true**): **s'executen les instruccions** dins de la `if`.
- Si (condició és **false**): les instruccions dins de `if` **no s'executen**.

```
public static void main(String[] args) {  
  
    char character= 'A';  
    char character2='B';  
  
    if (character == 'A') {  
        System.out.println ("El caràcter és A.");  
    }  
  
    if (character2=='B') {  
        System.out.println ("El caràcter és B.");  
    }  
  
}
```

## 1.1 ESTRUCTURA `if . . .`

- Podem **utilitzar** una **variable booleana** sense necessitat de fer cap comparació

No necessitem fer una **comparació** (la **variable** es **booleana**)

```
public static final double QUOTA_SOCI_ANUAL = 23.5;

public static void main(String[] args) {

    boolean esJubilat = true;
    boolean eraSoci = false;
    double quantitatAPagar = QUOTA_SOCI_ANUAL;

    if (esJubilat) {
        quantitatAPagar -= quantitatAPagar * 0.15;
    }
    if (eraSoci) {
        quantitatAPagar -= 5;
    }
    System.out.println(quantitatAPagar);
}
```

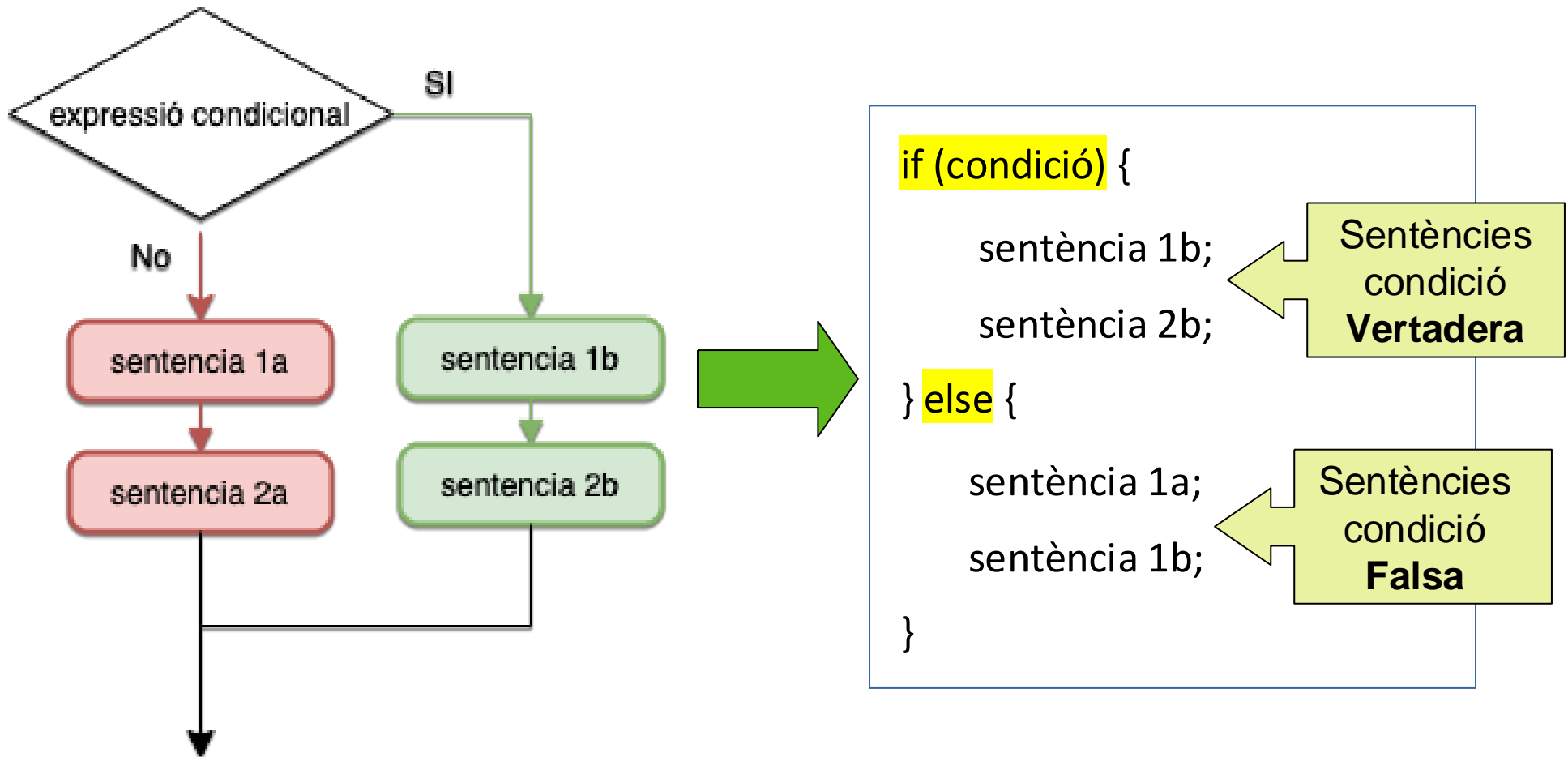
## 1.1 ESTRUCTURA `if...`

- També podem fer ús dels **operadors lògics** per poder manejar condicions més complexes:

### Expressió operador Expressió

```
public static void main(String[] args) {  
    boolean haPagatLaQuota = true;  
    boolean haPagatEntradaPuntual = false;  
    if (haPagatLaQuota || haPagatEntradaPuntual) {  
        System.out.println("Benvingut al centre esportiu");  
    }  
}
```

# 1.1 ESTRUCTURA `if...else`





# 1.1 ESTRUCTURA `if...else`

---

```
if (condició) {
```

**Bloc de codi a executar si la condició és certa**

```
} else {
```

**Bloc de codi a executar si la condició és falsa**

```
}
```

- La part **else** és opcional.
- Les **{ }** S'utilitzen obligatòriament per més d'una instrucció
  - És recomanable **posar-les sempre.**

## 1.1 ESTRUCTURA `if...else`

- Si (**condició és true**): S'executen les instruccions dins de l' **if**.
- Si (**condició és false**): S'executen les de l'**else**.

```
public static void main(String[] args) {  
    int edat = 15;  
    if (edat < 18) {  
        // codi a realitzar si la condició es compleix  
        System.out.println ("No pots sortir a l'hora de pati");  
    } else {  
        // codi a realitzar si la condició no es compleix  
        System.out.println ("Pots sortir");  
    }  
}
```

## 1.2 ACTIVITATS PRÈVIES

---

- **Activitat 1.-** Escriu un programa que sol·licite l'edat d'una persona i mostre un missatge per pantalla indicant **si és major d'edat o no**.
- **Activitat 2.-** Escriu un programa que demane un número enter a l'usuari i visualitze un missatge per pantalla indicant si es tracta d'un nombre **parell** o **senar** i **positiu** o **negatiu**. En cas de ser 0 s'indicarà a banda.

## 1.3 ESTRUCTURA `if...else if...else...`

- S'utilitzen per **niar condicions**.

Podem ficar tants **else if** com necessitem sempre i quan les **condicions** no es solapen.

```
public class Exemple {  
    public static void main(String[] args) {  
        int mes = 5;  
        if (mes == 1) {  
            System.out.print ("Gener");  
        } else if (mes == 2) {  
            System.out.print ("Febrer");  
        } else if (mes == 3) {  
            System.out.print ("Març");  
        } else {  
            System.out.print ("No sé ...");  
        }  
    }  
}
```

S'executarà quan **no es complixca** cap de les condicions anteriors

## 1.3.1 ACTIVITAT PRÈVIA

---

**Activitat 3.-** Escriu un programa que demane **1 qualificació numèrica** (entre zero i deu) i visualitze el seu equivalent segons la següent taula:

- Des de **0 e inferior a 3** -----> MOLT DEFICIENT
- Des de **3 i menor que 5** -----> INSUFICIENT
- Des de **5 e inferior a 6** -----> SUFICIENT
- Des de **6 i més xicoteta que 7** -----> BÉ
- Des de **7 i menor que 9** -----> NOTABLE
- Des de **9 Fins a 10** -----> EXCEL.LENT

En altre cas haurà de mostrar que la nota introduïda és invàlida.

## 1.4 ESTRUCTURA `switch...case`

- Construcció **sintàctica compacta** per **seleccionar 1 bloc de codi** a executar **dependent d'un valor**.
- S'utilitza com a **alternativa** a instruccions **if ... else enllaçades**.
- Únicament **sobre sencers, caràcters o strings**
- Les **condicions** han de ser **excloents**.

```
int mes = 2;
switch (mes) {
    case 1:
        System.out.println ("Gener");
        break;
    case 2:
        System.out.println ("Febrer");
        break;
    case 3:
        System.out.println ("Març");
        break;
    default:
        System.out.println ("No sé ...");
        break;
}
```

## 1.4 ESTRUCTURA `switch...case`

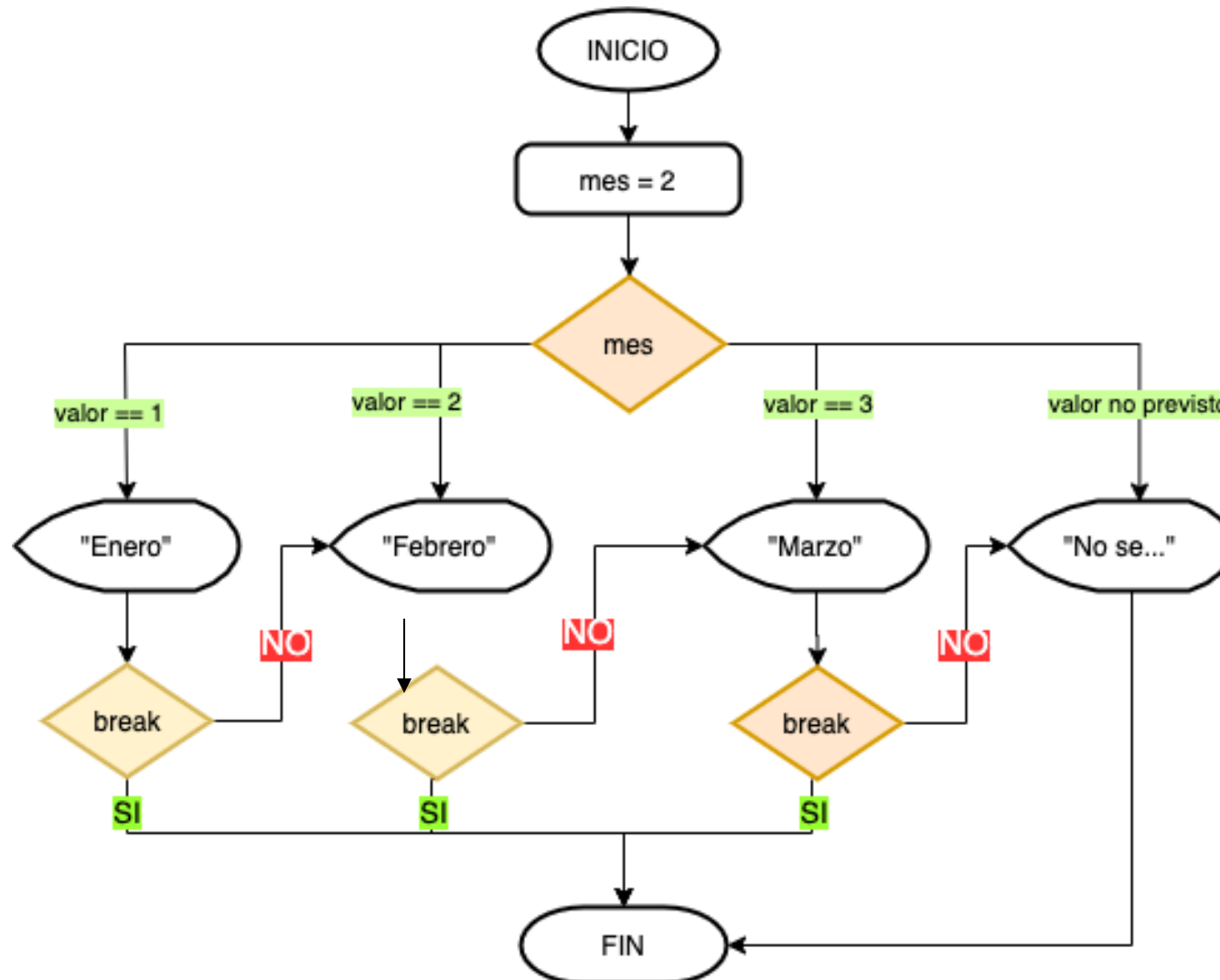
```
int mes = 2;  
switch (mes) {  
    case 1:  
        System.out.println ("Gener");  
        break;  
    case 2:  
        System.out.println ("Febrer");  
        break;  
    case 3:  
        System.out.println ("Març");  
        break;  
    default:  
        System.out.println ("No sé ...");  
}
```

Variable  
**selectora**

Instruccions a executar  
**Quan el mes és 2**

Instruccions a executar  
quan **no coincideix**  
amb **cap dels casos**  
**especificats**

## 1.4 ESTRUCTURA switch...case





## 1.5 ACTIVITATS PRÈVIES

---

**Activitat 4.-** Refactoritza l'activitat 3 per a que utilitze l'estructura `switch...case`. L'usuari sols podrà introduir notes senceres.

## 1.4 ESTRUCTURA switch...case

La sentència **break** provoca la **terminació de la sentència condicional**.

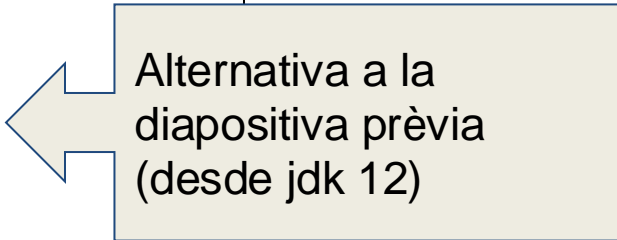
- Si no apareix, el codi següent **continua executant**.
- El **break del default és redundant**.

```
public static void main(string[] args) {  
    int mes = 1;  
    int dies = 0;  
    switch (mes) {  
        case 4:  
        case 6:  
        case 9:  
        case 11:  
            dies = 30;  
            break;  
        case 2:  
            dies = 28;  
            break;  
        default:  
            dies = 31;  
            break;  
    }  
}
```

Els casos 4,6,9 i 11  
executen les **mateixes**  
instruccions

## 1.4 ESTRUCTURA switch...case

```
public static void main(string[] args) {  
    int mes = 1;  
    int dies = 0;  
    switch (mes) {  
        case 4,6,9,11:  
            dies = 30;  
            break;  
        case 2:  
            dies = 28;  
            break;  
        default:  
            dies = 31;  
            break;  
    }  
}
```



Alternativa a la  
diapositiva prèvia  
(desde jdk 12)

## 1.4 ESTRUCTURA `switch...case`

Des de la versió **17 de Java**, existeix la possibilitat de escriure aquesta estructura d'aquesta manera:

```
int mes = 2;
switch (mes) {
    case 1 -> System.out.println ("Gener");
    case 2 -> System.out.println ("Febrer");
    case 3 -> System.out.println ("Març");
    case 4, 5, 6, 7, 8, 9, 10, 11, 12 -> System.out.println ("Altres
mes");
    default -> System.out.println ("No sé ...");
}
```

Amb aquesta opció no hi ha que escriure la sentència **break** al final de cada cas.

## 1.5 ACTIVITATS PRÈVIES

---

**Activitat 5.-** Adapta la estructura switch de l'activitat 4 a aquesta nova estructura. Si disposes d'un jdk anterior al 17 hauràs de descarregar una versió igual o posterior a aquest.

## 1.5 ACTIVITATS PRÈVIES

**Activitat 6.-** Desenvolupa un programa que ens informe del **tipus de contenidor** al que hem de **tirar 1 bossa de rebutjos**. El programa **mostrarà un menú** i demanarà a l'usuari que **inseririsca un nombre enter** que represente el **tipus de rebuig** que **necessita llançar**. el programa **mostrarà** el color del **contenidor** al qual ha de llençar la borsa.

*(Fes 2 versions; 1 fent ús de la estructura **switch** i l'altra de l'estructura **if .. else if ...**)*

opció	sortida
1.- Plàstic	contenidor <b>groc</b>
2.- Orgànic	contenidor <b>gris</b>
3.- Paper	contenidor <b>blau</b>
4.- Cartró	contenidor <b>blau</b>
5.- Un altre	contenidor <b>verd</b>

Qué tipo de desecho necesitas tirar:

1. Plástico
2. Orgánico
3. Papel
4. Cartón
5. Otros

Introduce una opción [1-5]

4

Debes tirarlo al contenedor Azul

## 1.6 QUAN UTILITZAR CADA ESTRUCTURA

---

- Les estructures **if...** s'utilitzen per controlar l'execució d'un únic bloc de codi.
- Les estructures **if...else** s'utilitzen per controlar l'execució de dues seccions de codi mútuament excloents.
- Les estructures **if...else if... else** s'utilitzen per controlar l'execució de 3 o més seccions de codi mútuament excloents.
- Les estructures **case...** s'utilitzen quan es disposa d'una llista de valors possibles finits i ben definits.

## 2. ÀMBIT DE LES VARIABLES

---

- **L'àmbit d'una variable és la part del programa en el qual és coneguda i es pot utilitzar.**
- **Les variables es poden declarar en qualsevol lloc.** No obstant això:
  - El seu **àmbit** vindrà **determinat** pel bloc en el qual han estat declarades {...}
  - Únicament **seran "visibles"** i, per tant, accessibles dins d'aquest bloc.



## 2. ÀMBIT DE LES VARIABLES

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

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

```
    System.out.println ("Introdueix una opció [1-2]");
```

```
    int opcio = teclat.nextInt();
```

```
    if (opcio == 1) {
```

```
        String opcionString = "Seleccionada opció" + opcio; ✓
```

```
        System.out.println (opcionString); ✓
```

```
    }
```

```
    System.out.println (opcionString); ✗
```

```
    System.out.println (opcio); ✓
```

```
    teclat.close();
```

```
}
```

Àmbit 1

Àmbit 2

Aquesta variable  
**sols és visible**  
dins de l'àmbit 2

## 2.1 ACTIVITAT PRÈVIA

**Qüestió 1:** Donats el següents programes. Quines errades tenen? Com les solucionaries?

### Programa 1

```
public static void main(String args[])
{
    {
        int x = 10;
        System.out.println(x);
    }
    System.out.println(x);
}
```

### Programa 2

```
public static void main(String args[])
{
    {
        int x = 5;
        {
            int x = 10;
            System.out.println(x);
        }
    }
}
```



### 3. OPERADOR TERNARI

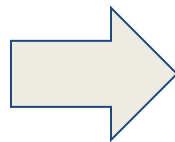
És una forma compacta de decidir entre dos valors:

**resultat = (condició) ? valor1: valor2**

- Si la **condició és certa** → es pren el **valor1**
- Si la **condició és falsa** → es pren el **valor2**

// Forma clàssica

```
if (condició) {  
    variable = valor1;  
} else {  
    variable = valor2;  
}
```



// Forma compacta

```
variable = (condició) ? valor1: valor2;
```

### 3. OPERADOR TERNARI

```
public static void main(String[] args) {  
    Scanner teclat = new Scanner(System.in);  
  
    System.out.println ("Introdueix la teva edat");  
    int edat = teclat.nextInt();  
  
    string missatge;  
    if (edat >= 18) {  
        missatge = "Pots passar";  
    } else {  
        missatge = "No pots passar";  
    }  
    System.out.println(missatge);  
    teclat.close();  
}
```



Equivalents

```
public static void main(String[] args) {  
  
    Scanner teclat = new Scanner(System.in);  
  
    System.out.println ("Introdueix la teva edat");  
    int edat = teclat.nextInt();  
  
    string missatge = (edat >= 18)? "Pots passar" : "No pots passar";  
    System.out.println (missatge);  
    teclat.close();  
}
```

## 3.1 ACTIVITATS PRÈVIES

**Activitat 7.-** Donat el següent programa en pseudocodi, refactoritza'l e **implementa'l en Java** perquè faça ús d'un operador ternari.

```
ALGORISME numeroMayor
VARIABLES
    num1 sencer
    num2 sencer
    missatge cadena
INICI
    ESCRIURE ( 'Introdueix un nombre enter');
    LLEGIR (num1);
    ESCRIURE ( 'Introdueix un altre nombre enter');
    LLEGIR (num2);
    SI (num1 > num2)
        missatge = "Num 1 es major ";
    SI_NO
        missatge = "Num 2 és major o igual a num1 ";
    FIN_SI
    ESCRIURE (missatge);
FI
```

## 3.1 ACTIVITATS PRÈVIES

**Activitat 8.-** Escriu un programa que calcule la quota que s'ha d'abonar en un club de golf. La quota general és de 500€. Tindran un **45% de descompte** les persones **majors de 65 anys** i un **25% els menors** de 18 anys. Pensa bé l'estructura que has de gastar i tracta de fer una única eixida per pantalla per informar del resultat a l'usuari.

*Demana a l'usuari el nom i l'edat i, a continuació, mostra la informació que has de pagar.*

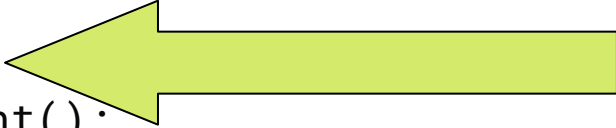
### Exemple d'execució

```
Insereix nom: Amparo  
Insereix edat: 30  
Amparo, has de pagar 500.0 euros.
```

## 4. CONTROL ENTRADA dades per TECLAT

La **classe** `Scanner` presenta una sèrie de **mètodes** que ens **permeten comprovar** el tipus de dades introduït per l'usuari de manera que ens permet **controlar els possibles errors**.

```
public static void main(String[] args) {  
  
    Scanner teclat = new Scanner (System.in);  
    System.out.println ("Introdueix una opció [1-2]");  
  
    if (teclat.hasNextInt ()) {  
        int opcio = teclat.nextInt();  
        System.out.println (opcio);  
    } else {  
        System.out.println ("Hauries d'haver introduït un sencer");  
        return;  
    }  
}
```



## 4.1 ACTIVITAT PRÈVIA

---

**Activitat 9.-** Modifica les activitats 7 i 8 realitzades en aquesta unitat perquè **duguen a terme el control** del tipus de dades introduïdes per teclat.

*Si no s'introdueix el tipus adequat, es presentarà el missatge "Cal introduir un **enter** / **float**" i finalitzarà l'execució del programa.*



## 4.1 ACTIVITAT PRÈVIA

---

Això és tot ... de moment :-)