DOC1: Introducció a la programació estructurada

1. Variables

Les **variables** són una part essencial de la programació, ja que permeten emmagatzemar informació que es pot utilitzar i modificar al llarg del programa. A JAVA, cada variable té un **tipus de dada** que determina quins valors pot emmagatzemar i quines operacions es poden realitzar amb ella. Les **variables** s'han de declarar abans d'usar-les i, en molts casos, inicialitzar-les amb un valor.

Tipus de dades:

```
int edat = 20;
double pes = 70.5;
char sexe = 'M';
boolean esEstudiant = true;
String nom = "Joan";
```

Tipus de dades comuns:

`int`: És el tipus més comú per a emmagatzemar nombres enters, com per exemple el número de persones, l'edat, etc.

```
int edat = 20;
```

`double`: S'utilitza per a emmagatzemar nombres amb decimals, com per exemple el pes, la temperatura, el temps, etc.

```
double pes = 70.5;
```

`char`: Representa un sol caràcter, com per exemple una lletra o un símbol.

```
char sexe = 'M';
char lletra = 'a';
char acceptar ='n';
```

`boolean`: S'utilitza per a valors lògics, com per exemple saber si una condició és certa o falsa.

```
boolean esEstudiant = true;
boolean acceptarCondicions = true;
```

`String`: És un tipus especial per emmagatzemar cadenes de text, com noms, frases, etc. Recorda que `String` comença amb majúscula, ja que és una classe especial en JAVA.

```
String nom = "Joan";
String saludo = "Hola com estas, tot bé?";
```

2. Expressions

Una **expressió** és una combinació de variables, constants, operadors i/o funcions que avaluen les variables donant un únic resultat. Les expressions són molt útils perquè permeten calcular nous valors dins del codi.

Operadors:

Els operadors més comuns que podem utilitzar en les expressions són:

- **Aritmètics**: per a sumar (`+`), restar (`-`), multiplicar (`*`), dividir (`/`), o obtenir el residu d'una divisió (`%`).
- **Relacionales**: per comparar valors (`==`, `!=`, `<`, `>`, `<=`, `>=`).
  - a == b Compara si els dos valors son iguals.
  - a != b Compara si els dos valors son diferents.
  - a < b Compara si el valor a és més petit que el valor b
  - a > b Compara si el valor a és més gran que el valor b
  - a <= b Compara si el valor a és més petit o igual que el valor b.
  - a >= b Compara si el valor a és més gran o igual que el valor b.
- **Lògics**: per combinar condicions (`&&` per a "i", `||` per a "o", `!` per a "no").
  - Si (a==1 && b==1) Comproba que per a que sigui cert tant el valor a i b han de ser 1. Retorna true o false.
  - Si (a==1 || b==1) Estableix que per a que sigui cert el valor a o b han de ser 1.
  - Ex: Retorna el contrari variable booleana es el contrari.

```
boolean resultat1 = suma > 10;

boolean resultat3 = !resultat1;
```

Resultat de resultat3 es false. Ja que resultat1 es true pero al ficar ! davant es transforma amb false.

Exemple de combinació d'expressions:

```
int x = 10;
int y = 5;
int suma = x + y;  // suma de dos variables
boolean resultat1 = suma > 10;
boolean resultat2 = suma == 10;// compara el resultat
boolean resultat3 = !resultat1;
System.out.println(resultat1);
System.out.println(resultat2);
System.out.println(resultat3);
```

3. Condicionals (IF i SWITCH)

Els **condicionals** permeten que el programa prengui decisions en funció de les condicions que es compleixin. Permeten que un bloc de codi s'executi només si es compleix una condició determinada.

IF

L'`if` és l'instrument més senzill per a la selecció de codi basat en una condició. Si la condició és **true**, s'executa el bloc de codi dins de l'`if`; si és **false**, es passa al següent bloc (si existeix). Aquest condicional funciona molt bé per avaluar condicions, true, false, and, or etc..

Exemple de l'ús de `if`:

```
int edat = 18;
if (edat >= 18) {
    System.out.println("Ets adult");
} else {
    System.out.println("Ets menor d'edat");
}
```

En aquest cas, el programa imprimeix si la persona és adulta o no, segons l'edat.

L'`if` també es pot utilitzar amb l'`else` per especificar què passa si la condició és falsa. També podem utilitzar l'`else if` per a més condicions.

SWITCH

El `switch` és una estructura més eficient quan hi ha múltiples opcions possibles per a una mateixa variable. Aquesta estructura compara el valor de la variable amb diferents **casos** (`case`). Va molt bé per a quan tenim menús, diferents opcions a seleccionar etc..

Exemple de l'ús de `switch`:

```
int dia = 3;
switch (dia) {
   case 1:
      System.out.println("Dilluns");
      break;
   case 2:
      System.out.println("Dimarts");
      break;
   case 3:
```

```
            System.out.println("Dimecres");
            break;
        default:
            System.out.println("Dia desconegut");
    }
```

El `switch` fa una comparació de la variable amb els valors dels casos. Si la variable coincideix amb algun dels casos, s'executa el codi corresponent. El `break` s'utilitza per sortir de l'estructura `switch` després d'executar un cas. Si cap cas coincideix, es passa al `default`, que és opcional.

4. Bucles (WHILE i FOR)

Els **bucles** són essencials per repetir un conjunt d'instruccions fins que es compleixi una condició determinada. Són útils per a tasques repetitives i per a processos on el nombre d'iteracions no es coneix exactament des del principi.

WHILE

El bucle `while` repeteix un conjunt d'instruccions mentre una condició sigui **true**. Si la condició és **false** des del principi, el bloc de codi dins del `while` no s'executarà. En aquest cas, el bucle es repetirà fins que `i` sigui igual a 5. **Important:** sempre cal tenir una forma de sortir del bucle, com incrementar el valor de `i`.o avaluar una condició.

Exemple de `while`:

```
int i = 0;
      while (i < 5) {
          System.out.println("Iteració " + i);
          i++;  // augmenta i per evitar un bucle infinit.
      }
```

DO WHILE

El bucle **`do while`** és similar al bucle **`while`**, però amb una diferència important: mentre que el bucle `while` comprova la condició abans d'executar el codi, el **`do while`** sempre executa el conjunt d'instruccions almenys una vegada, independentment de si la condició és **true** o **false** en la primera iteració. Després de l'execució, es comprova si la condició continua sent **true** per seguir executant el codi.

Exemple:

```
int i = 0;
do {
   System.out.println("Volta " + i);
   i++;  // Incrementa i
} while (i < 5);
```

En aquest exemple, el bucle `do while` imprimeix "volta 0", "volta 1", fins a "volta 4" perquè, tot i que la condició `i < 5` es comprova després de cada iteració, el codi s'executarà almenys una vegada abans de comprovar si `i` és menor que 5.

Les diferències entre el do while i el while son les següents:

- En un `while`, si la condició és falsa des del principi, el codi **no es mostrarà mai**.
- En un `do while`, el codi **s'executa almenys una vegada**, independentment de la condició, ja que la condició es comprova després de l'execució.

Exemple real d'un do while:

El programa demana a l'usuari que introdueixi una opció i segueix demanant fins que l'usuari selecciona l'opció `0` per sortir.

```
Scanner scanner = new Scanner(System.in);
int opcio;
do {
      System.out.println("Selecciona una opció (1 per
      continuar, 0 per sortir):");
    opcio = scanner.nextInt();
} while (opcio != 0);
System.out.println("Has sortit del programa.");
```

FOR

El bucle `for` és ideal quan sabem exactament quantes vegades volem que s'executi un conjunt d'instruccions. Es defineixen tres parts: inicialització, condició i actualització. Aquí, la variable `i` s'inicialitza a 0, el bucle s'executa mentre `i` sigui menor que 5, i després de cada iteració, `i` s'incrementa en 1 (`i++`).

Exemple de `for`:

```
for (int i = 0; i < 5; i++) {
   System.out.println("Valor de la volta " + i);
```

```
}
```

Exemple de `for amb valor d'una variable`:

```
int anys = 18

for (int i = 0; i < anys; i++) {
    System.out.println("Faig tantes voltes fins arribar als anys que tens " + i);
}
```

SCANNER

`Scanner` s'utilitza per capturar dades introduïdes per l'usuari a través de la consola.

La classe `Scanner` està disponible al paquet `java.util`. Per utilitzar-la, cal importar-la al començament del programa:

Per llegir dades de la consola, cal crear un objecte de tipus `Scanner`. Aquest objecte es vincula amb el sistema d'entrada estàndard (teclat) mitjançant `System.in`:

L'objecte `Scanner` ofereix diferents mètodes per capturar valors de diversos tipus. Alguns dels més comuns són:

Tipus de dada	Mètode `Scanner`	Exemple d'ús
Enter	nextInt()	int edat = scanner.nextInt();
Decimal	nextDouble()	double pes = scanner.nextDouble();
Text	nextLine() (linia completa)	String nom = scanner.nextLine();
Una paraula	next()	String paraula = scanner.next();
Booleà	nextBoolean()	boolean resposta = scanner.nextBoolean();

Exemple per a demanar a l'usuari un nom, edat i pes.:

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

    // Capturar una cadena de text
    System.out.println("Introdueix el teu nom:");
    String nom = scanner.nextLine();  // Llegir text

    // Capturar un enter
    System.out.println("Introdueix la teva edat:");
    int edat = scanner.nextInt();  // Llegir un número enter

    // Capturar un decimal
    System.out.println("Introdueix el teu pes (en kg):");
    double pes = scanner.nextDouble();  // Llegir un número decimal

    // Mostrar la informació capturada
    System.out.println("Hola, " + nom + "! Tens " + edat + " anys i peses " + pes + " kg.");

    // Tancar l'objecte Scanner
    scanner.close();
```