Conceptes bàsics

- Evitar repetir estructures de codi
- Conjunt d'instruccions agrupades sota un nom i amb un objectiu comú
- Àmbit de les variables
- Notació a Java

```
static void numFunció() {
    …
    instruccions
    …
}

main() {
    nomFunció();
}
```

Pas d'informació

```
public static void numFunció(tipus nomVariable) {
     …
     instruccions usant nomVariable
     …
}
```

Exercicis:

1. Dissenyar una aplicació amb la funció `eco()`. L'aplicació demana un número `n` i repeteix el text "Eco…" n-vegades.

2. Fer una aplicació que demani dos sencers i, a través d'una funció, mostri tots els nombres que hi ha entre ells de major a menor.

3. Fes una funció que mostri per pantalla l'àrea O el volum d'un cilindre. Per distingir un cas de l'altre s'inclou com a paràmetre un número: si val 1 calcula l'àrea, si val 2, calcula el volum. A més, a la funció s'ha de passar el radi de la base i l'altura.

$$\text{área} = 2\pi \cdot \text{radio} \cdot (\text{altura} + \text{radio})$$

$$\text{volumen} = \pi \cdot \text{radio}^2 \cdot \text{altura}$$

Retorn d'un valor

```
static tipus numFunció( … ) {
    …
    instruccions
    …
    return variable; // Del tipus correcte
}
```

Exercicis:

1. Dissenyar una funció que rebi dos valors sencers i retorni el valor màxim d'ambdós.

2. Fer una aplicació que demani el nombre d'hores, minuts i segons. Una funció ha de comprovar si els valors introduïts són correctes. És a dir, que les hores i els minuts no tinguin més de 60 minuts o segons respectivament.

Sobrecàrrega de funcions

```
static int suma(int a, int b) {
    int suma = a + b;
    return suma;
}

static double suma(int a, float pesA, int b, float pesB) {
    double suma = a*pesA + b*pesB;
    return suma;
}

public static void main() {
    int resultat1 = suma(4, 5);
    double resultat2 = suma(4, 0.25, 3, 0.75);
    System.out.println(resultat1);
    System.out.println(resultat2);
}
```

Exercicis:

1. Fes una aplicació que determini el màxim entre 2 o 3 números. Ha de tenir dues funcions amb sobrecàrrega a les que puguem passar 2 o 3 xifres.

Recursivitat

```
static int funcioRecursiva() {
    …
    funcioRecursiva();
    …
}
```

Exemple: resoldre n! = n · (n-1) · (n-2) · … · 2 · 1

```
long factorial(int n) {
    long resultat;
    if (n==1) {
        resultat = 1;
    } else {
        resultat = n * factorial(n-1);
    }
    return resultat;
}
```

Cas base: Si n és 1
el resultat és 1

Exercicis:

1. Fes una funció recursiva per a calcular el terme *n* de la sèrie de Fibonacci. El n-terme de la sèrie de Fibonacci es calcula obtenint la suma dels dos termes anteriors, és a dir:
   fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
   fibonacci(0) = 0;
   fibonacci(1) = 1;

Variables globals

```
public class JavaApplication {

    static tipus variable;

    public static void main(String[] args) {
        …
        instruccions
        …
    }

    static tipus funcio() {
        …
        instruccions
        …
    }
}
```