DOC2: Programació estructurada avançada - Funcions i mètodes.

1. Funcions

Una funció (o mètode) és un bloc de codi que realitza una tasca específica. Permet reutilitzar codi, millorar l'organització del programa i facilitar-ne el manteniment.

En Java, totes les funcions es defineixen dins d'una classe. Per tant, també es coneixen com mètodes.

Una funció en Java es compon de:

1. Modificadors d'accés (opcional): Determinen la visibilitat de la funció. Exemples: public, private, protected.
2. Tipus de retorn: Indica el tipus de dades que retorna la funció (o void si no retorna res).
3. Nom de la funció: Ha de ser descriptiu i seguir les convencions de nomenclatura (camelCase).
4. Paràmetres (opcional): Entrades que la funció rep per treballar.
5. Cos de la funció: Conté les instruccions que executa.

Sintaxi general:

```
[modificador d'accés] tipus_de_retorn nomDeLaFunció (paràmetres) {

    // Cos de la funció

    instruccions;

    [retorn (si no és void)];

}
```

Exemple bàsic

Funció que suma dos números:

```
public class Exemple {
    public static int suma(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {
        int resultat = suma(5, 10);
        System.out.println("El resultat és: " + resultat);
    }
}
```

Paraules clau importants de les funcions:

return: Utilitzada per retornar un valor des de la funció.

void: Indica que la funció no retorna cap valor.

static: Fa que la funció sigui accessible sense crear una instància de la classe.

Tipus de funcions

1. **Sense retorn ni paràmetres** Només executen codi intern i no tenen entrades ni sortides.

```
public static void mostraMissatge() {
    System.out.println("Hola, món!");
}
```

2. **Amb retorn però sense paràmetres** Retornen un valor però no requereixen dades d'entrada.

```
public static int obtenirNumero() {
    return 42;
}
```

3. **Amb paràmetres però sense retorn** Accepten dades d'entrada però no retornen cap valor.

```
public static void mostraText(String text) {
    System.out.println(text);
}
```

4. **Amb retorn i paràmetres** Accepten dades d'entrada i retornen un resultat.

```
public static double calculaArea(double base, double altura) {
        area =  base * altura / 2;
        return area;
}
```

Sobrecàrrega de funcions

Permet definir diverses funcions amb el mateix nom però amb paràmetres diferents.

```
public class Sobrecarga {
    public static int suma(int a, int b) {
        return a + b;
    }

    public static double suma(double a, double b) {
        return a + b;
    }
}
```

Bones pràctiques

1. Escriu funcions curtes i amb un propòsit clar.
2. Utilitza noms de funcions descriptius.
3. Si una funció es fa massa complexa, divideix-la en funcions més petites.

4. Sempre documenta les funcions amb comentaris per facilitar-ne l'entesa.

Avantatges d'utilitzar les funcions

- **Funció reutilitzada:** La funció calculaAreaRectangle es defineix una vegada i es crida dues vegades amb diferents valors d'entrada.
- **Eficàcia:** Amb aquesta estructura, evitem haver d'escriure el codi per calcular l'àrea dues vegades.
- **Modificació fàcil:** Si en el futur volem canviar la lògica de càlcul de l'àrea (per exemple, afegir decimals), només cal modificar la funció i no totes les crides.

Exemple funciona on s'ahorra codi:

```
public class ExempleReutilitzacio {
    // Funció per calcular l'àrea d'un rectangle
    public static int calculaAreaRectangle(int amplada, int alçada) {
        return amplada * alçada;
    }

    public static void main(String[] args) {
        // Dades del primer rectangle
        int amplada1 = 5;
        int alçada1 = 10;

        // Dades del segon rectangle
        int amplada2 = 8;
        int alçada2 = 4;

        // Utilitzem la funció per calcular l'àrea dels dos rectangles
```

```
        int area1 = calculaAreaRectangle(amplada1, alçada1);
        int area2 = calculaAreaRectangle(amplada2, alçada2);

        // Mostrem els resultats
        System.out.println("L'àrea del primer rectangle és: " + area1);
        System.out.println("L'àrea del segon rectangle és: " + area2);
    }
}
```