Conceptes bàsics

Introducció a la programació modular

Primer de tot...

- El concepte de la programació modular és simple, consisteix en **dividir el programa en mòduls**.

- En realitat, es tracta de dividir el problema en **subproblemes** més **senzills** i clarament diferenciats.

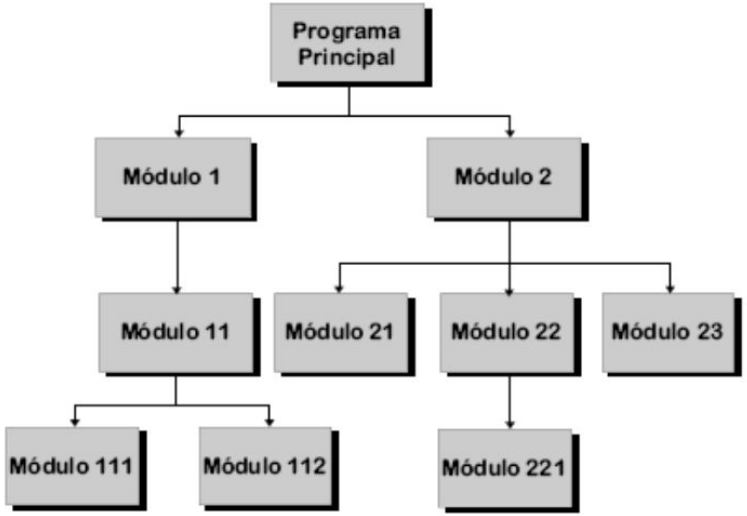
- Cal crear programes **clars, intel·ligibles i breus** perquè puguen ser **llegits, entesos i modificats fàcilment**.

Primer de tot...

La programació estructurada i modular solen ser **complementàries**. Es fan servir criteris de programació modular per **descompondre** el problema en parts independents i després s'utilitza la programació estructurada per **desenvolupar cada mòdul**.

Què és un mòdul?

Un mòdul és un **conjunt d'instruccions contigües**, les quals es poden **referenciar** mitjançant un nom i poden ser **cridats** en diferents punts del programa. Un mòdul pot ser un ***programa, funció o procediment***.



Avantatges i inconvenients

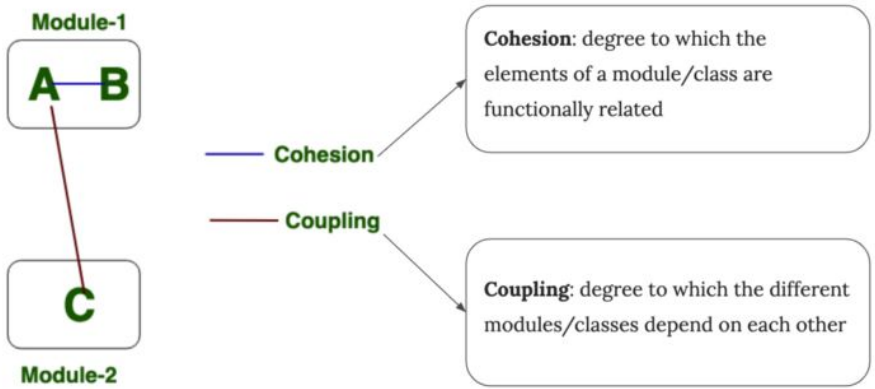
Descompondre el programa en mòduls suposa:

- AVANTATGES: Programes més fàcils d'escriure i depurar, més fàcils d'entendre, modificar i mantenir, reutilització de codi, treball en equip...

- INCONVENIENTS: Increment d'interfícies entre mòduls, més memòria i temps d'execució.

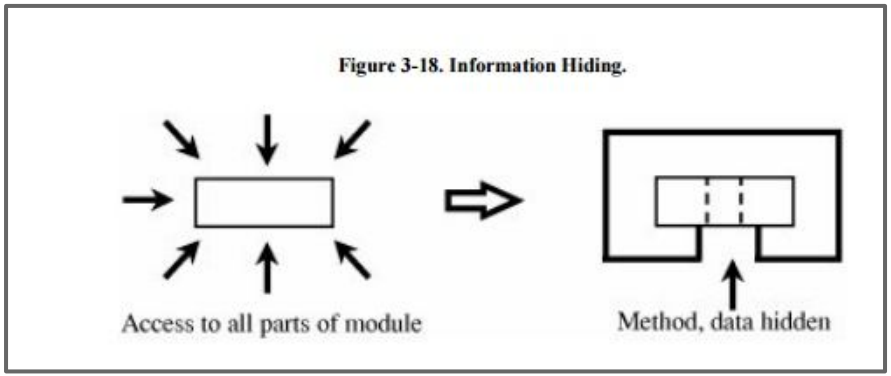
INDEPENDÈNCIA FUNCIONAL

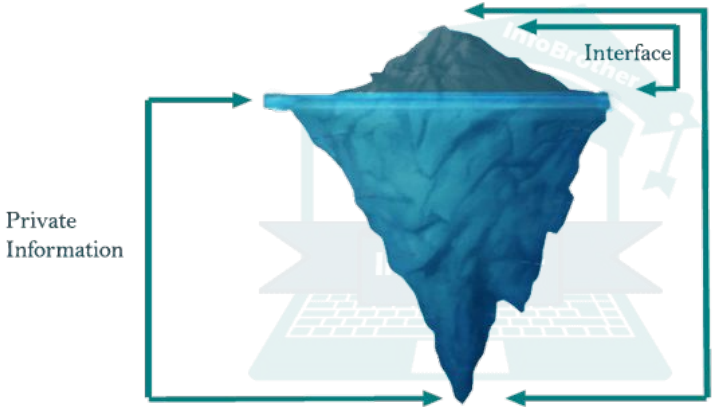
- El nostre objectiu serà **especialitzar** cada mòdul.
- La independència funcional implica que s'han de dissenyar mòduls **altament cohesionats** (relacionats amb la resolució d'una única tasca) i **poc acoblats** (poc relacionats amb elements d'altres mòduls).



Ocultació de la informació

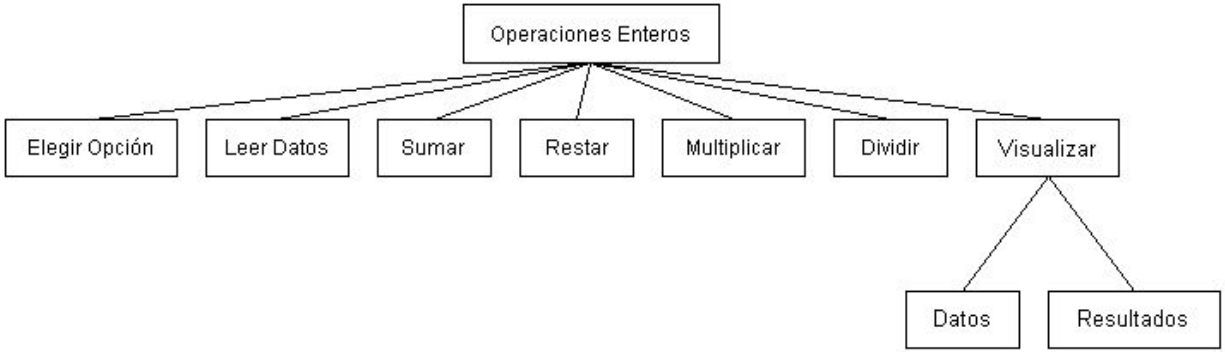
La informació continguda dins d'un mòdul **ha de ser inaccessible per a altres mòduls que no necessiten eixa informació** (si cada programador es dedica a programar una part del programa pot fer ús de mòduls ja creats, però sense interessar-se per com duen a terme la seua tasca).





Metodologia descendent o top-down

- Ja sabem que abans d'implementar un programa s'ha de dissenyar.
- Per al disseny d'un programa modular seguirem una **metodologia descendent** amb **refinament successiu**, és a dir, partim del problema genèric, es **descompon en problemes més concrets** i estos alhora en **tasques més senzilles** fins a arribar al nivell de tasca més bàsic.



Elements bàsics d'un subprograma

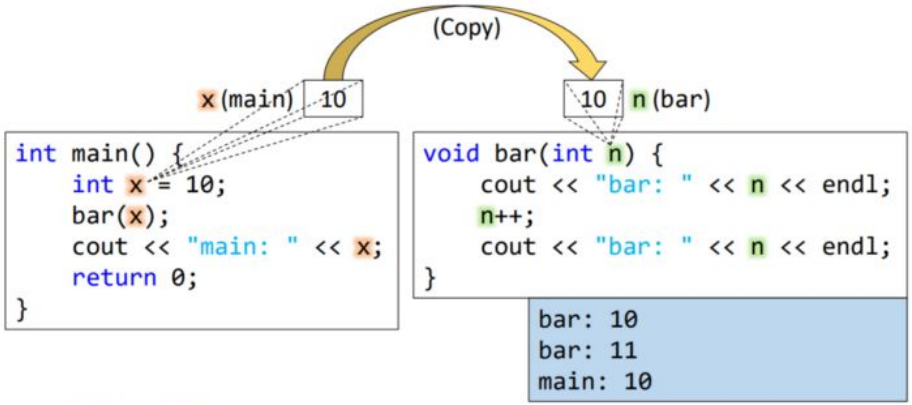
- Identificador
- Llista de paràmet (variables de comunicació)
  - Formals
  - Actuals o arguments
- Cos
- Entorn

```java
public class Prova{
    private static final int FACTOR = 3;

    public static int operar(int operand1, int operand2){
        int resultat = operand1 * operand2 * FACTOR;
        return resultat;
    }

    public static void main(String[] args){
        int valor1 = 32;
        int valor2 = 3;
        int primerResultat = operar(valor1,valor2);
    }
}
```

Pas de paràmetres a un subprograma

- **Per valor:** JAVA sempre ho fa així. El paràmetre serà una dada d'ENTRADA per al subprograma. Esta variable **no podrà ser modificada** pel subprograma perquè el que fa és **una còpia del seu valor al paràmetre formal** corresponent per poder utilitzar-lo.

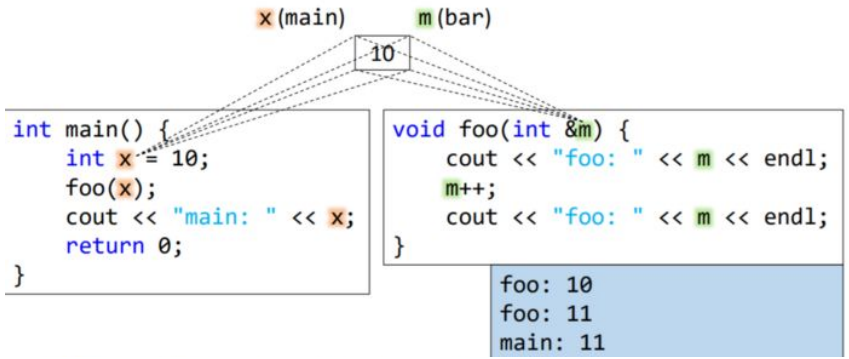
Exemple en llenguatge c++



Pas de paràmetres a un subprograma

- **Per referència:** En alguns llenguatges és possible passar una variable al subprograma i que este puga **col·locar un resultat en esta variable**, que queda a disposició del programa que crida este subprograma. Realment allò que es proporciona al subprograma és l'**adreça de memòria** on es troba la variable (una referència).

Exemple en llenguatge c++



I en Java????

Tipus de subprogramES. Funcions

- Matemàticament és una **operació que pren un o més valors (arguments) i produeix un resultat determinat (només un)**.
- L'execució farà que s'**assigne un valor a esta funció** (és a dir, que la funció seria com una expressió equivalent al resultat).
- Per tant, es pot assignar a una variable, imprimir dins d'un print, o utilitzar-se com qualsevol altre valor en una expressió matemàtica o com a argument d'una altra funció.

```
public static int sumar(int operand1, int operand2){
    int resultat = operand1 + operand2;
    return resultat;
}
```

```
public static void main(String[] args){
    int numero = sumar(3,5);
    int numero2 = sumar(numero, sumar(6,1));

    if(sumar(3,-5) < numero){
        System.out.println("OK");
    }
}
```

Tipus de subprogramES. Procediments, accions o subrutines

- Un procediment és capaç de calcular diversos resultats o cap. **No retorna un resultat** al programa que els invoca, simplement **es limita a executar una sèrie d'instruccions**. Per exemple, un procediment es pot encarregar de rebre un array i ordenar-ho.

- Com que **no té cap valor associat al procediment**, no es pot assignar a una variable com les funcions, o utilitzar-se com a argument d'un altre subprograma, simplement es cridarà en el punt o punts que desitgem.

Tipus de subprogramES. Procediments, accions o subrutines

```java
public static void main(String[] args){
    imprimirMenu();
    // Altres instruccions bla bla bla...
    imprimirMenu();
}

public static void imprimirMenu(){
    System.out.println("-- MENÚ DEL PROGRAMA --");
    System.out.println("-----------------------");
    imprimirOpcio(1, "Sumar");
    imprimirOpcio(2, "Restar");
    imprimirOpcio(3, "Multiplicar");
    imprimirOpcio(4, "Dividir");
    System.out.println();
}

public static void imprimirOpcio(int numero, String opcio){
    System.out.println(numero + ". " + opcio);
}
```