Programació

UT6. Classes: Interacció i organització. Paquets

Introducció

- Fins ara hem treballat sempre en un únic fitxer, el qual conté tota la lògica desenvolupada (**programa monolític**).
- El concepte de **programació modular** definia la divisió del problema en subproblemes.
- L'estructura del programa també pot ser dividida en subestructures (carpetes i fitxers).

Programa modular

- És aquell que hem desenvolupat dividint-lo en components fàcils de modificar i intercanviar, anomenats **mòduls.** Cada mòdul pot agrupar altres mòduls amb una sèrie de funcions o mètodes que realitzen tasques relacionades.

Programes amb múltiples classes

- La manera més directa és establir sempre **un mòdul per fitxer** implicat a el programa.
- La majoria de llenguatges de programació permeten definir esta estructura.
- En Java, **cada mòdul s'anomenarà classe**.

Què és realment una classe?

- És un concepte directament relacionat amb la **metodologia orientada a objectes** que estudiarem en la propera unitat.
- Ara bé, hem usat este concepte per referir-nos a:
  - **Un programa en Java.** Tots els fitxers que fins ara contenen el mètode „main“, i tots els mètodes que s'han definit, estan declarats amb la paraula reservada „**class**“.
  - **Un repositori de mètodes.** S'ha usat este terme per a referir-nos a una biblioteca de mètodes, que actuen com extensions a les instruccions per defecte del llenguatge. Per exemple, la classe Scanner (mètodes: nextLine(), nextInt(), etc…), Math…
  - **Un tipus compost.** S'ha usat com sinònim del que coneixem com a tipus compost. Per exemple, la classe String i els seus mètodes per la gestió/manipulació de cadenes de caràcters.

Què és realment una classe?

Tots tenen en comú que disposen d'una sèrie de mètodes que és possible invocar. I és que els tres casos, tot i les seues diferències, **són codi font dins d'un fitxer** anomenat "`NomClasse.java`", amb la declaració `public class NomClasse`.... i una sèrie de mètodes declarats dins d'eixe àmbit.

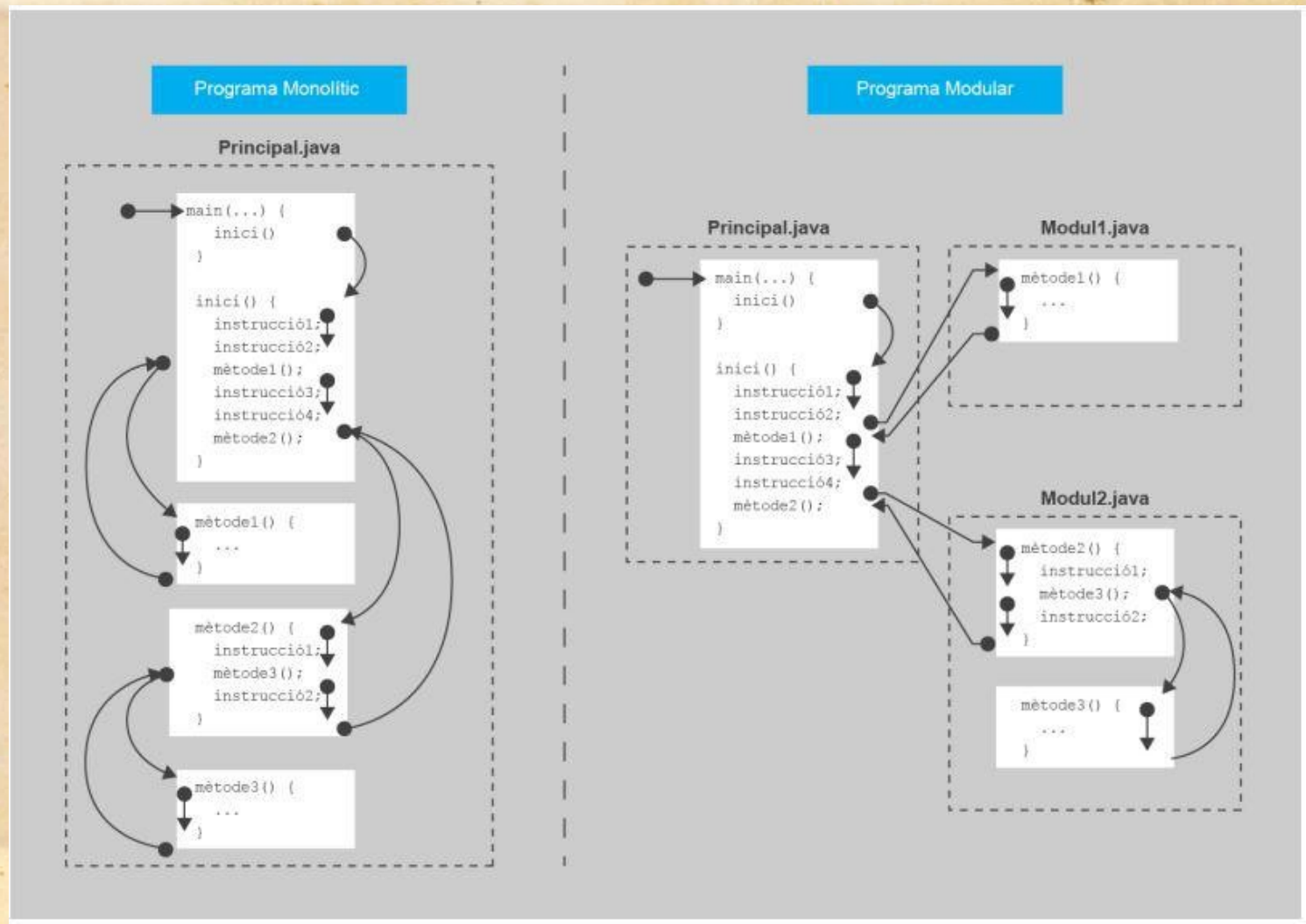
Descomposició modular

- La descomposició en subproblemes no és arbitrària, sinó que es planteja com un objectiu parcial, amb entitat pròpia, per resoldre part del problema de nivell superior.

Pràctica 1

- Busca en Internet el codi font de les classes `Scanner` i `String` i verifica el que s'ha comentat anteriorment.

Programa monolític
vs Programa modular



Programa monolític RegistreNotes (I)

```
public class RegistreNotes {
  public static void main(String[] args) {
    RegistreNotes programa = new RegistreNotes();
    programa.inici();
  }
  public void inici() {
    double[] notes = {2.0, 5.5, 7.25, 3.0, 9.5, 8.25, 7.0, 7.5};
    double max = calcularMaxim(notes);
    double min = calcularMinim(notes);
    double mitjana = calcularMitjana(notes);
    System.out.println("La nota màxima és " + max + ".");
    System.out.println("La nota mínima és " + min + ".");
    System.out.println("La mitjana de les notes és " + mitjana + ".");
  }
  public double calcularMaxim(double[] array) {
    double max = array[0];
    for (int i = 1; i < array.length; i++) {
      if (max < array[i]) {
        max = array[i];
      }
    }
    return max;
  }
```

Continua a la següent

Programa monolític RegistreNotes (II)

```
  public double calcularMinim(double[] array) {
    double min = array[0];
    for (int i = 1; i < array.length; i++) {
      if (min > array[i]) {
        min = array[i];
      }
    }
    return min;
  }
  public double calcularMitjana(double[] array) {
    double suma = 0;
    for (int i = 0; i < array.length; i++) {
      suma = suma + array[i];
    }
    return suma/array.length;
  }
}
```

Pràctica 2

- Crea una classe anomenada "`CalculsArrayReals`", que contindrà els mètodes "`calcularMinim`", "`calcularMaxim`" i "`calcularMitja`" de les diapositives anteriors.
- La nova classe s'ha de crear en la mateixa ubicació on es troba la classe amb el mètode "main" (RegistreNotes.java)

Tingues la precaució de que esta nova classe NO continga també el mètode main.

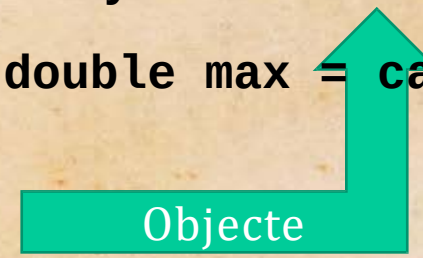
Pràctica 3

- Elimina la definició dels mètodes que ara conté la classe `CalculsArrayReals`, de la classe `RegistreNotes`.
- Invoca els mètodes definits a `CàlculsArrayReals` des de `RegistreNotes`. Utilitza per això la següent instrucció:

```
CalculsArrayReals calculador = new CalculsArrayReals();

double max = calculador.calcularMaxim();

.....
```



Reutilització de mòduls

- Observa com el mòdul CalculsArrayReals ara pot ser utilitzat per altre programa distint, ja que hem independitzat completament la seua funcionalitat.

Pràctica 4.

- Crea una nova classe Java anomenada `RegistreTemperatures`. En ella volem crear un programa que calcule la diferència de la màxima i mínima temperatura de una sèrie de temperatures donades.
- La classe principal només ha de tindre definits els mètodes "`main`" i "`inici`".
- Utilitza la classe `CalculsArrayReals` per obtindre els càlculs, igual que vas fer en la pràctica 3.

Biblioteques de classes

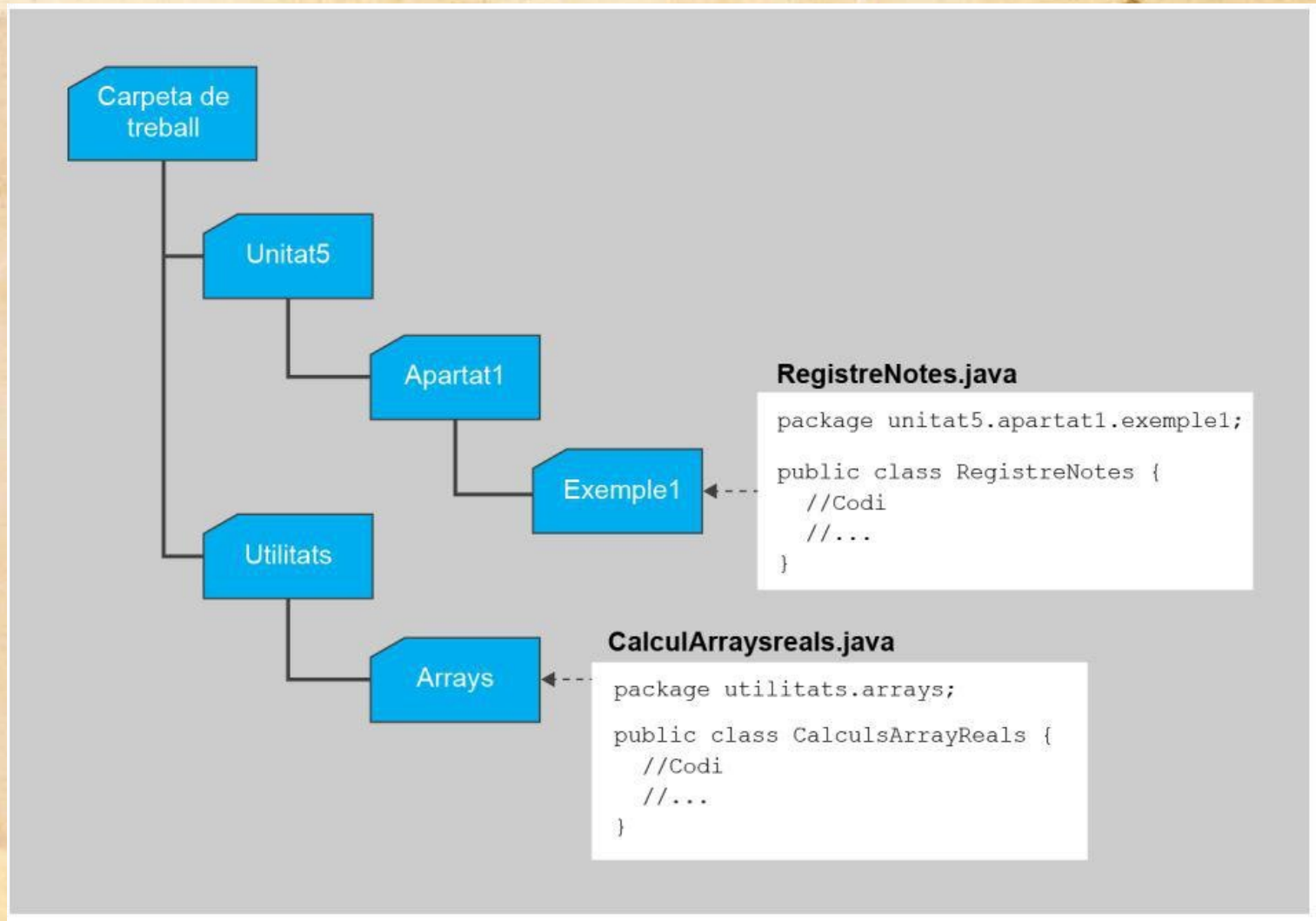
- També anomenats **package,** agrupen un conjunt de classes vinculades entre elles d'acord a algun criteri temàtic o d'organització del seu codi.
- **Donada una classe només pot pertànyer a un package**.
- Donat un package, dins d'ell no poden haver dos classes amb el mateix nom.

Per al sistema operatiu els "package" son realment carpetes

Pràctica 5

- Crea un projecte amb un package anomenat `unitat6.apartat1.exemple1`. Posa dins d'eixe package el fitxer `RegistreNotes`, executant-lo per a comprovar el seu funcionament.
- Crea un nou paquet `unitat6.apartat1.exemple2`. Posa dins el fitxer `RegistreTemperatures`
- En el mateix projecte, crea un nou paquet que serà germà del paquet unitat6, anomenat `utilitats.arrays` Posa dins el fitxer `CàlculsArrayReals`
- Comprova que tots els programes funcionen.

Estructura de els packages



Pràctica 6

- Modifica ara la classe `CàlculsArrayReals`. Anem a afegir a la definició dels 3 mètodes plantejats la paraula reservada `static` a la dreta de `public`.
- Invoca ara des d'els programes `RegistreNotes` i `RegistreTemperatures` els mètodes ara static.

Recorda que per a invocar els mètodes `static` no es realitza a través d'un objecte sinó amb el nom de la classe

```
NomClasse.metode();
```

exemple: `CalculesArrayReals.calcularMaxim();`

Constants privades

- Ja coneixes per què serveixen les constants, són valors que mai canvien en tota l'execució del programa.
- Per a declarar una constant de classe (fora del “main”) utilitzem la següent nomenclatura:

```
private static final tipus NOM = valor;
```

- La paraula reservada `private` indica que **la constant només podrà ser accedida des de la classe on es va declarar**.

Constants públiques

- Si per el contrari, canviem private per public, **la constant podrà ser accedida des de qualsevol classe que forme part del projecte**.

```
public static final tipus NOM = valor;
```

- Per a accedir a esta constant, donat que és static, accedirem igual que en el cas de els mètodes.

```
NomClasse.NOM;
```

- Exemple: `CalculArrayReals.TAMANY_ARRAY;`

Mètodes privats

- En este punt ja podràs intuir que si bé tots els mètodes definits fins este moment han sigut públics (`public`) també els podem fer privats (`private`).
- Al igual que les constants, un mètode serà privat, si només és utilitzat en l'àmbit de la classe on es defineix i no fora.

Pràctica 7

- Anem a modificar la classe `CàlculsArrayReals`.
- Afegeix una constant privada `TAMANY_ARRAY` i comprova que no pots accedir a esta constant des d'una altra classe del mateix projecte així com sí que ho pots fer des de dins de qualsevol mètode de la pròpia classe.
- De la mateixa manera, modifica el mètode `calcularMaxim` per a que ara siga privat. Apareix algun error en el teu projecte? Per què?