Programació Orientada a Objectes

Que és?

Es un paradigma o model de programació.

Estructurar i organitzar les nostres aplicacions amb diferents objectes.

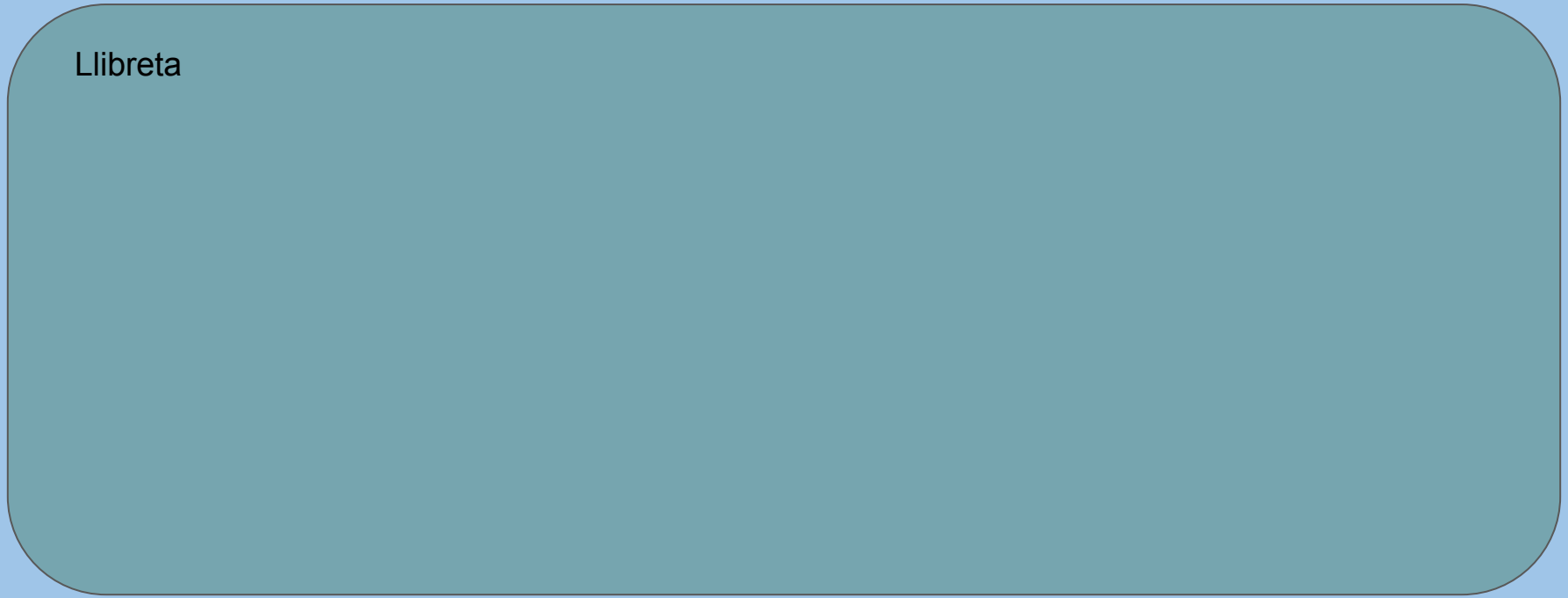
Associar els objectes amb conceptes del món real.

No és un llenguatge o una tecnología específica.

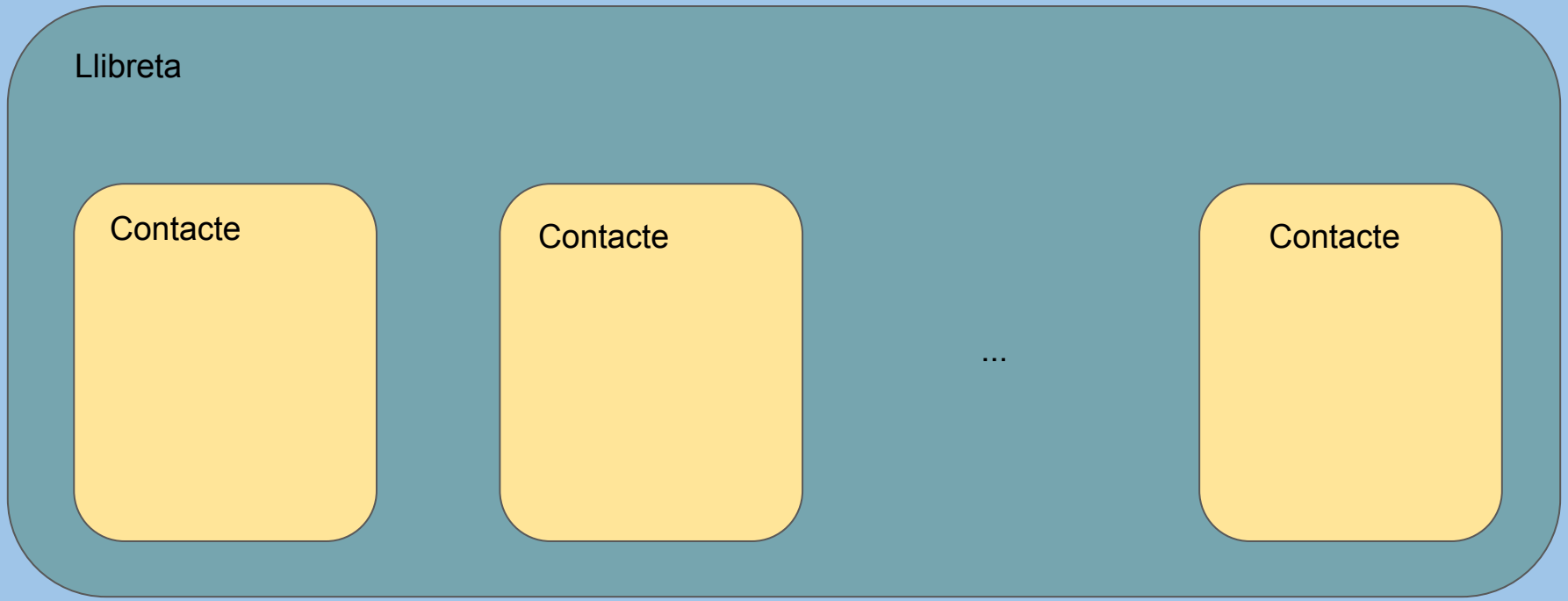
Avantatges

- Divisió de l’aplicació amb mòduls més petits.
- Més fàcil de mantenir.
- Flexible als canvis.
- Reutilització de codi.

Exemple conceptual



Exemple conceptual



Exemple conceptual

Contacte

Nom
Teléfon
Correu electrònic

Adreça física (Carrer, número, pis, porta, codi postal...)

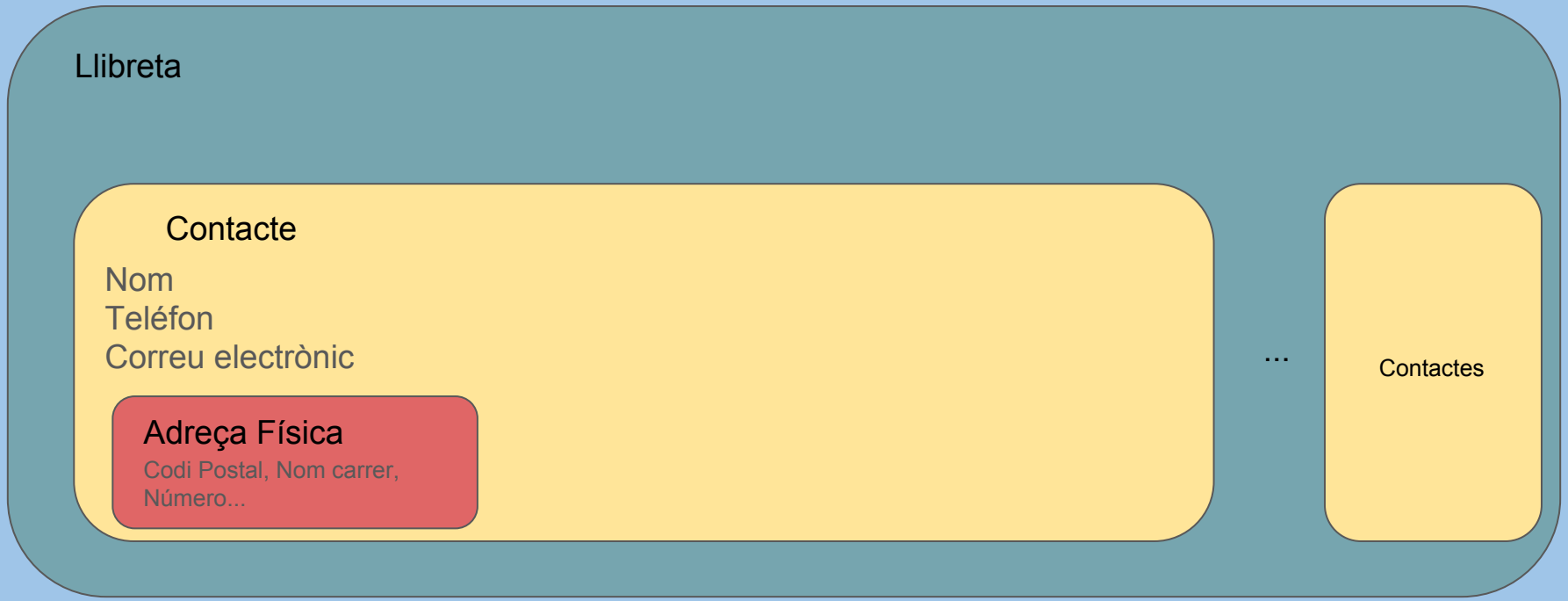
Exemple conceptual

Exemple conceptual

Adreça Física

Codi Postal
Nom carrer
Número
Pis
Porta

Exemple conceptual



Conceptes POO - Classe

Plantilla o esquema que representa un determinat concepte.

Conté les propietats i mètodes relacionats amb el concepte.

```
public class Contacte {

}
```

Conceptes POO - Classe

Plantilla o esquema que representa un determinat concepte.

Conté les propietats i mètodes relacionats amb el concepte.

```java
public class Contacte {

    //  Propietats
    String nom;
    String telefon;
    String correuElectronic;

}
```

Conceptes POO - Classe

Plantilla o esquema que representa un determinat concepte.

Conté les propietats i mètodes relacionats amb el concepte.

```java
public class Contacte {

    //  Propietats
    String nom;
    String telefon;
    String correuElectronic;

    //  Cosntructors

    //  Mètodes
    public void imprimirContacte() {
        System.out.println( "Contacte: " + nom + ", Telèfon: " + telefon + ", Correu: " + correuElectronic );
    }
}
```

Crear classe - Direcció física

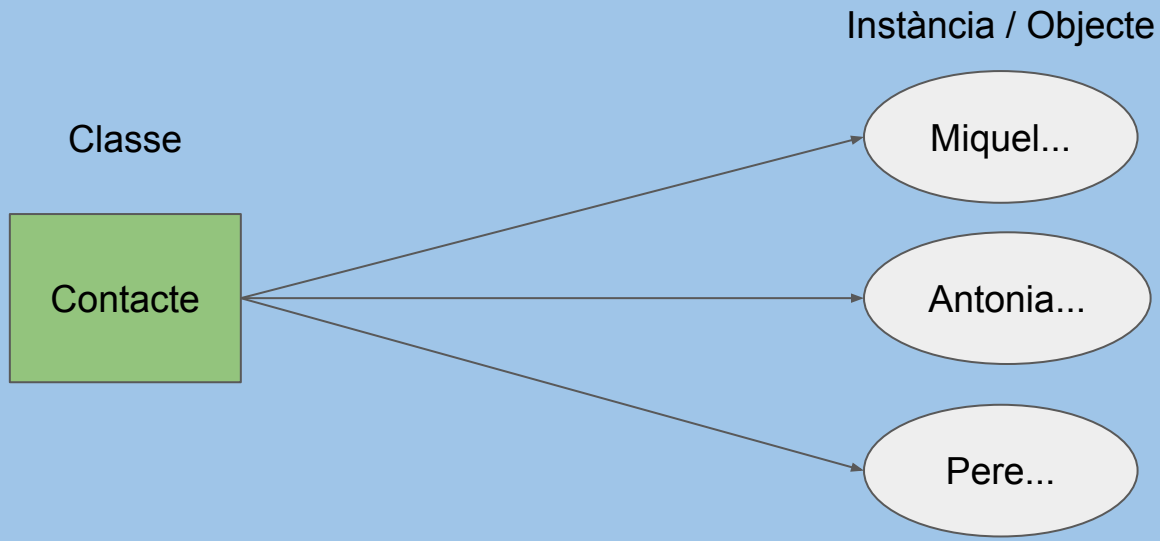
Crear classe - Direcció física

```java
public class DireccioFisica {

    //  Propietats
    String codiPostal;
    String nomCarrer;
    int numero;
    int pis;
    String porta;

    //  Constructor

    //  Mètodes
    public void imprimirDireccio() {
        System.out.println("C/" + nomCarrer + ", " + numero + ", CP: " + codiPostal + ", " + pis + porta);
    }
}
```

Conceptes POO - Objecte

Un objecte és una instància d'una classe.

A partir d'una **classe** podem crear tantes instances (**objectes**) com desitgem.



Conceptes POO - Constructor

Un constructor es el primer que s'executa quan cream una instància.

Es defineix dins la pròpia **Classe**

Molt parescut a un mètode, però:

- Sempre comença amb public

- No s'especifica el retorn

- Ha de tenir exactament el mateix nom que la classe.

```
public NomClasse() {

}
```

Conceptes POO - Constructor

```java
public class Contacte {

    //  Propietats
    String nom;
    String telefon;
    String correuElectronic;

    //  Cosntructors
    public Contacte() {
    }

    //  Mètodes
    public void imprimirContacte() {
        System.out.println( "Contacte: " + nom + ", Telèfon: " + telefon + ", Correu: " + correuElectronic );
    }
}
```

Crear constructor - Direcció física

Crear constructor - Direcció física

```java
public class DireccioFisica {

    //  Propietats
    String codiPostal;
    String nomCarrer;
    int numero;
    int pis;
    String porta;

    //  Constructor
    public DireccioFisica() {

    }

    //  Mètodes
    public void imprimirDireccio() {
        System.out.println("C/" + nomCarrer + ", " + numero + ", CP: " + codiPostal + ", " + pis + porta);
    }
}
```

Conceptes POO - Creació d'una instancia

NomClasse nomVariable = new NomConstructor();

```
public class POO {

    public static void main(String[] args) {

        Contacte cont1 = new Contacte();

    }

}
```

```
public class POO {

    public static void main(String[] args) {

        Contacte cont1;

        // ...

        cont1 = new Contacte();

    }

}
```

Paràmetres als constructors

```java
public class Contacte {

    //  Propietats
    String nom;
    String telefon;
    String correuElectronic;

    //  Cosntructors
    public Contacte() {

    }

    public Contacte(String nomPassat) {
        nom = nomPassat;
    }

    //  Mètodes
    public void imprimirContacte() {
        System.out.println( "Contacte: " + nom + ", Telèfon: " + telefon + ", Correu: " + correuElectronic );
    }
}
```

Propietats i mètodes - Ús

- Assignació

  nomVariable.propietat = valor;

- Cridar un mètode

  nomVariable.metode();

Àmbit - public i private

private: La propietat o el mètode només serà visible (llegir o modificar) dins la pròpia classe.

públic: La propietat o el mètode será accessible des de qualsevol lloc.

→ Donar accés i control total a la variable.

→ No podem garantitzar el valor desitjat.

→ El funcionament desitjat de la classe pot variar.

Àmbit - Getters i Setters

Crear un mètode per a recuperar el valor i un mètode per a assignar un valor.

SI ES NECESSARI.

Setter → Mètode que rep per paràmetre el valor de la variable que volem assignar.

Getter → Mètode que retorna el valor de la variable que hem fet privada.

Concepte - this

El terme “this” ens serveix per referencia qualsevol propietat o funció de la nostra classe.

Per exemple:

this.nomPropietat

this.funcio()

Concepte - static

Atribut que es pot associar als mètodes i a les propietats

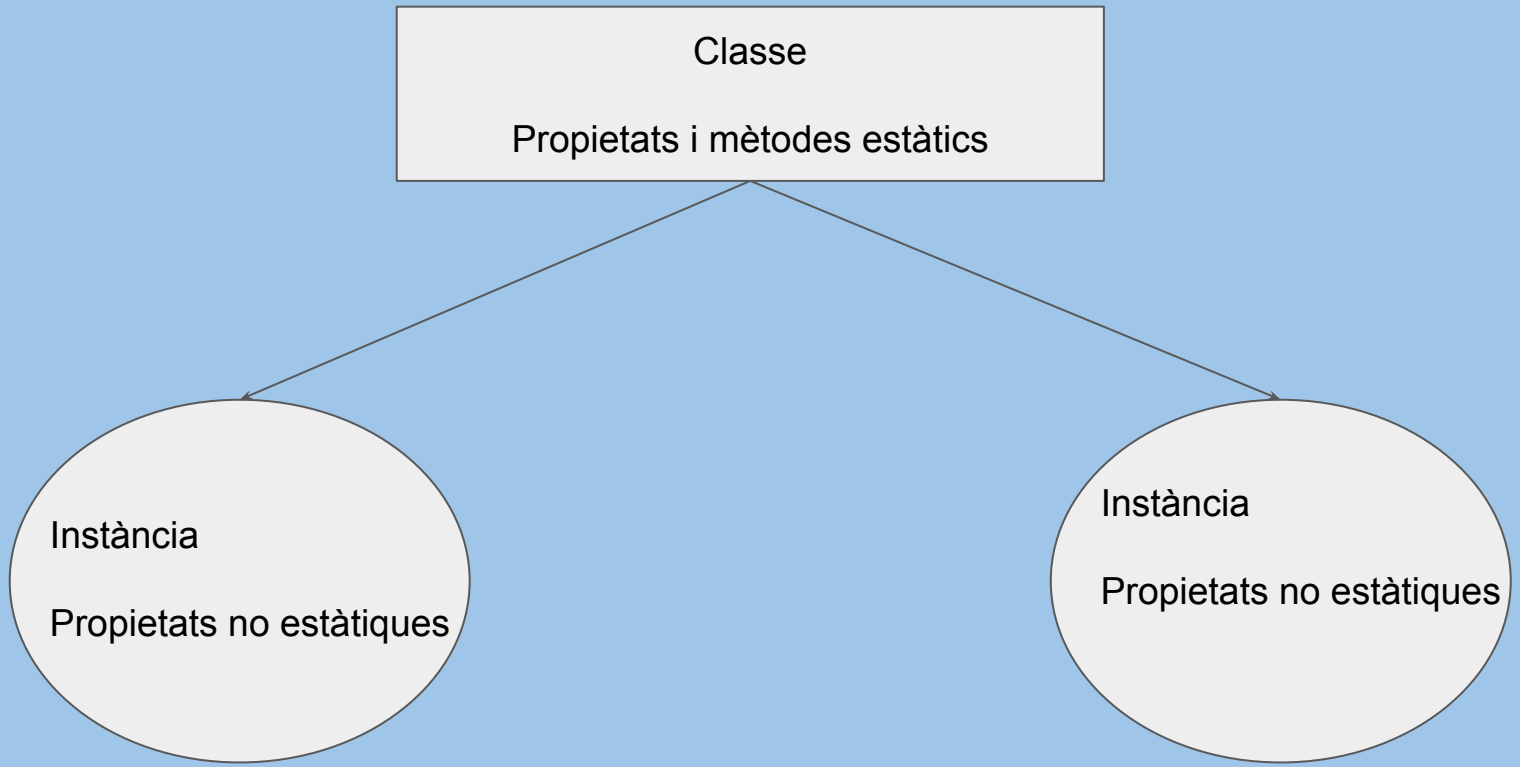
Quan una propietat o mètode és “static” es defineix a nivell de **Classe** i **NO** a la **instància**

→ Pot ser emprada sense necessitat de crear una instància.

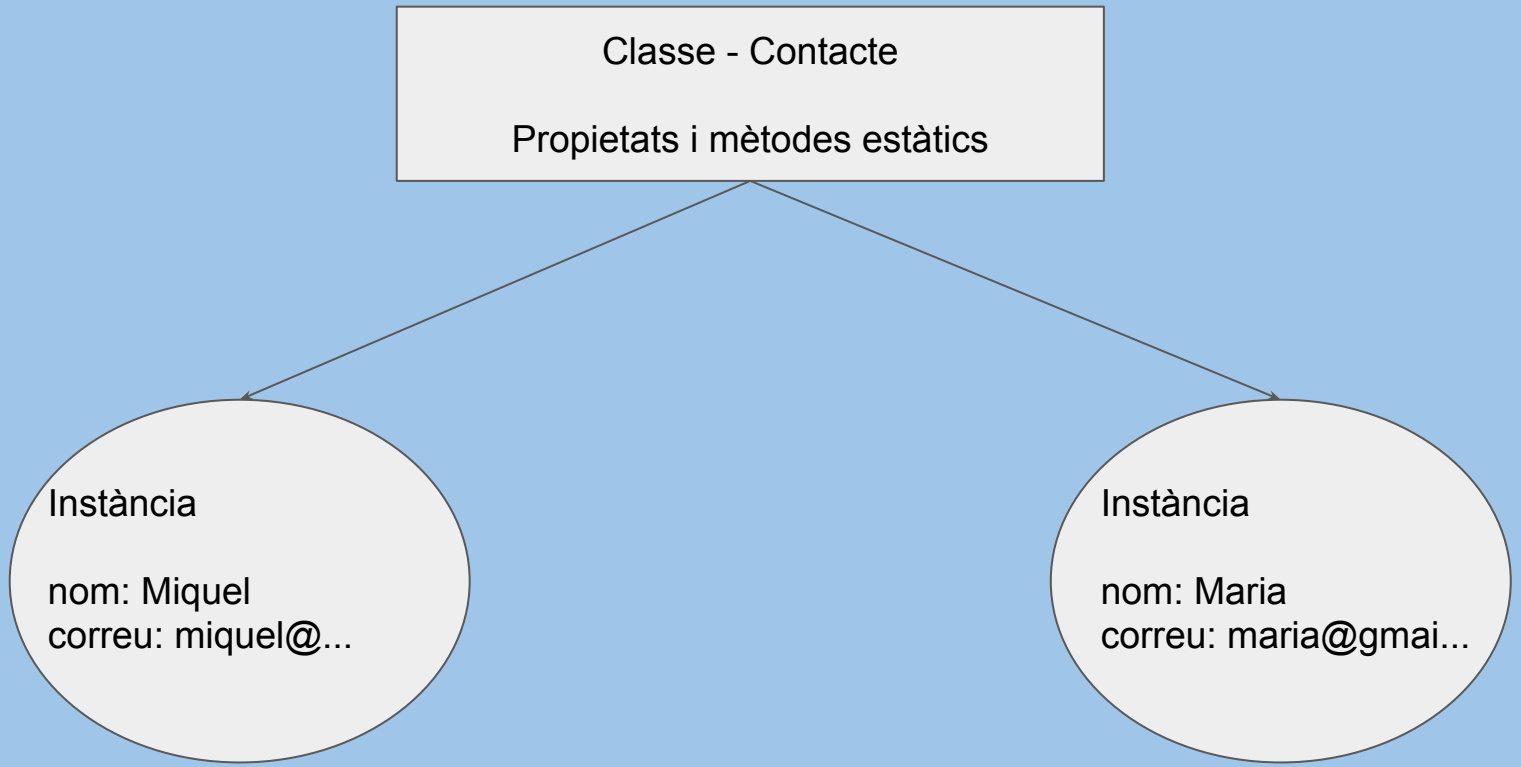
→ NomClasse.metode()

→ NomClasse.propietat

Concepte - static



Concepte - static



Exemple

Validació de dades - Constructor

- Si les dades no són vàlides hem d'aturar la creació de l'instància

- Excepcions → IllegalArgumentException