

Classe 17.10.2021: Introducció a Disseny

Anna Puig

Enginyeria Informàtica

Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

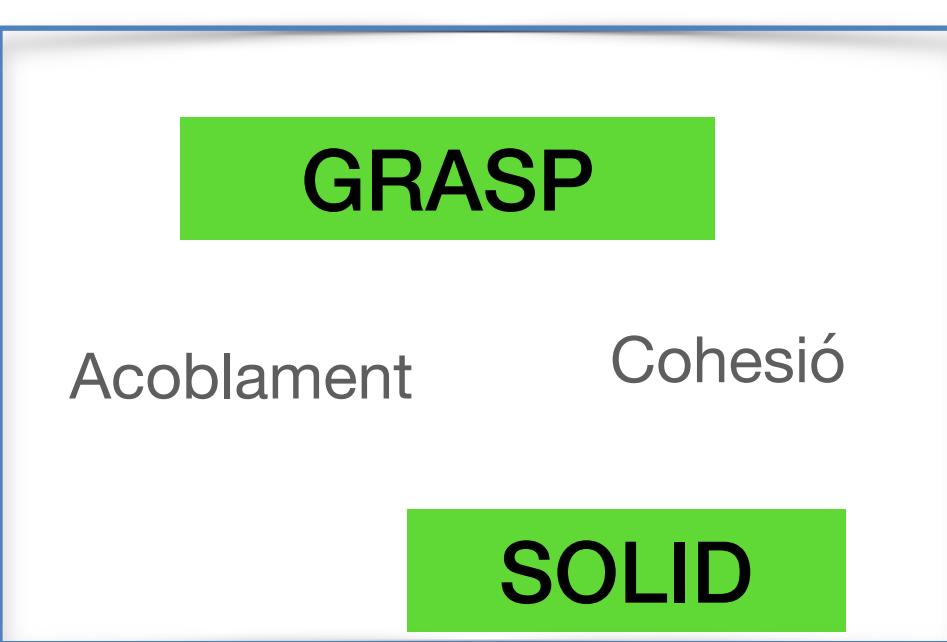
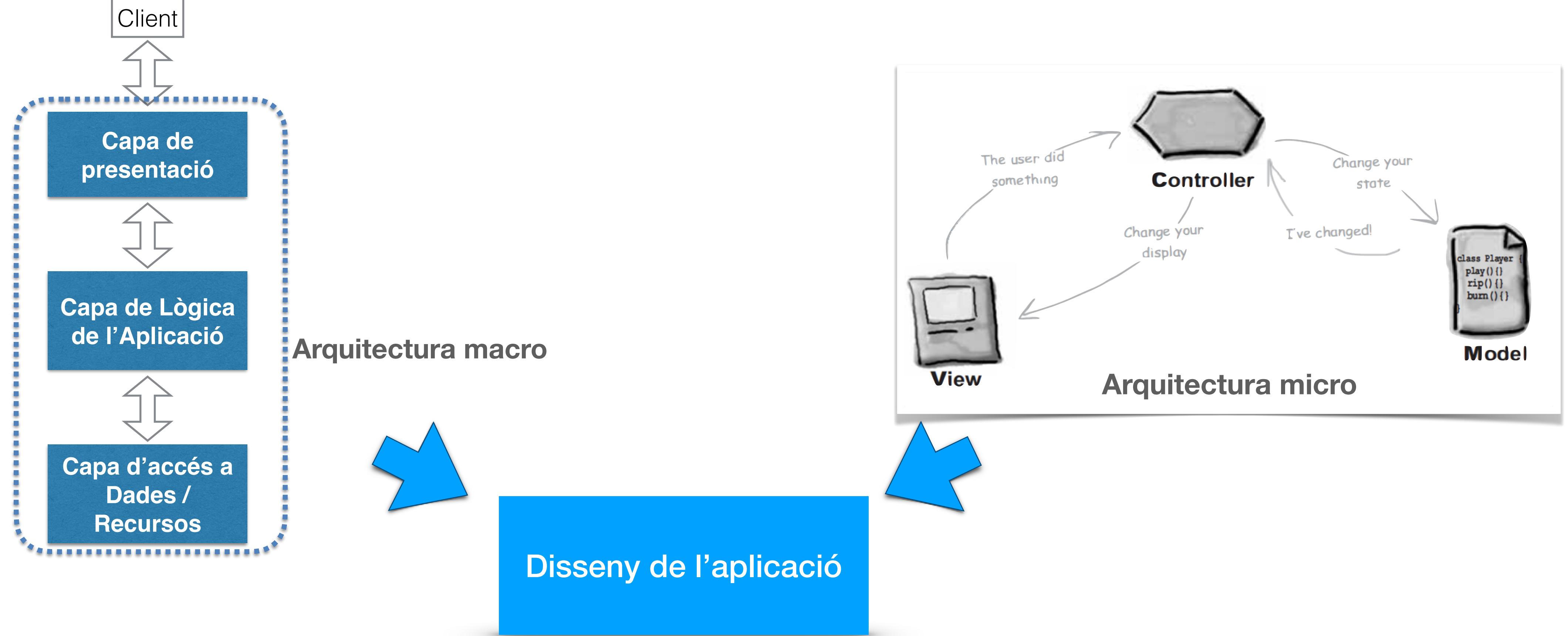
Curs 2021/22



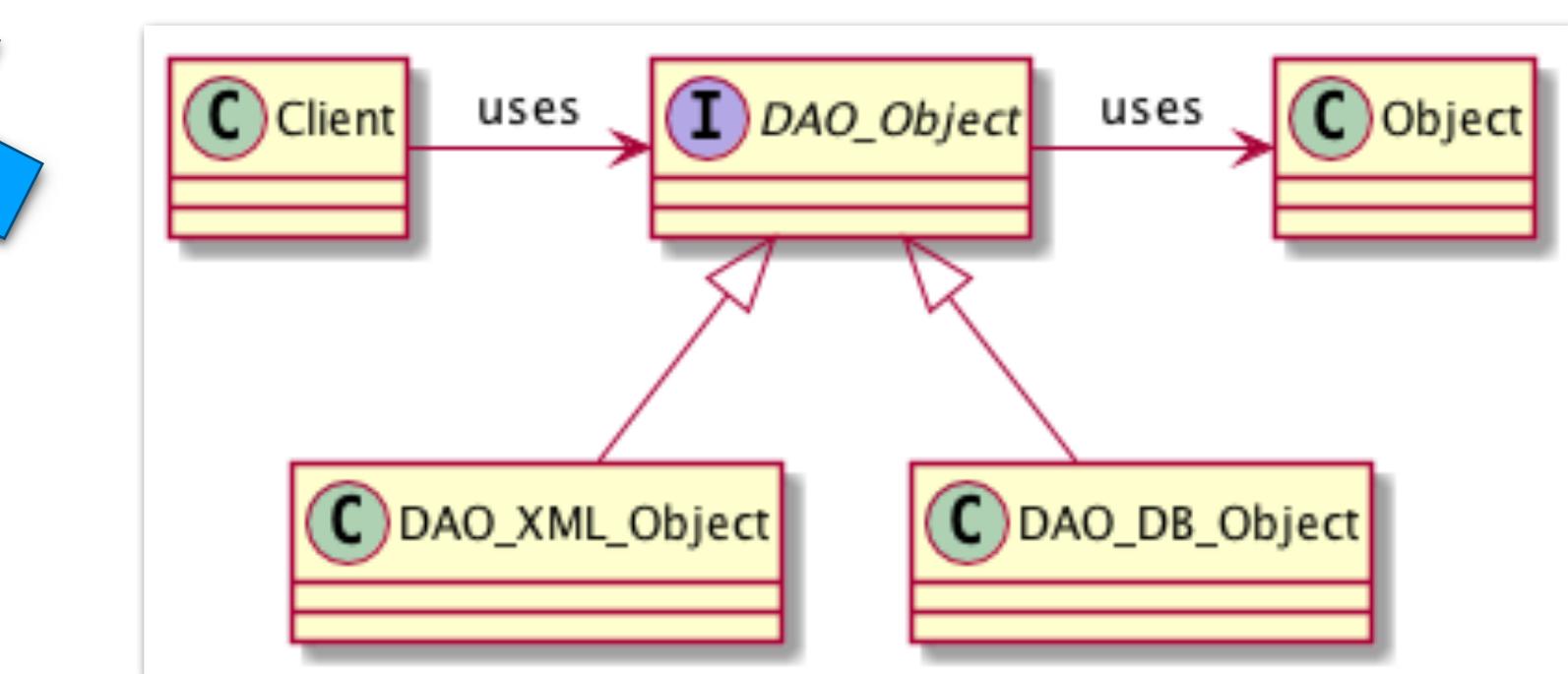
UNIVERSITAT DE
BARCELONA

Temari

1	Introducció al procés de desenvolupament del software	
2	Anàlisi de requisits i especificació	
3	Disseny	3.1 Introducció
4	Del disseny a la implementació	3.2 Patrons arquitectònics
5	Ús de frameworks de testing	3.3 Criteris de Disseny: G.R.A.S.P. 3.4 Principis de Disseny: S.O.L.I.D. 3.5 Patrons de disseny

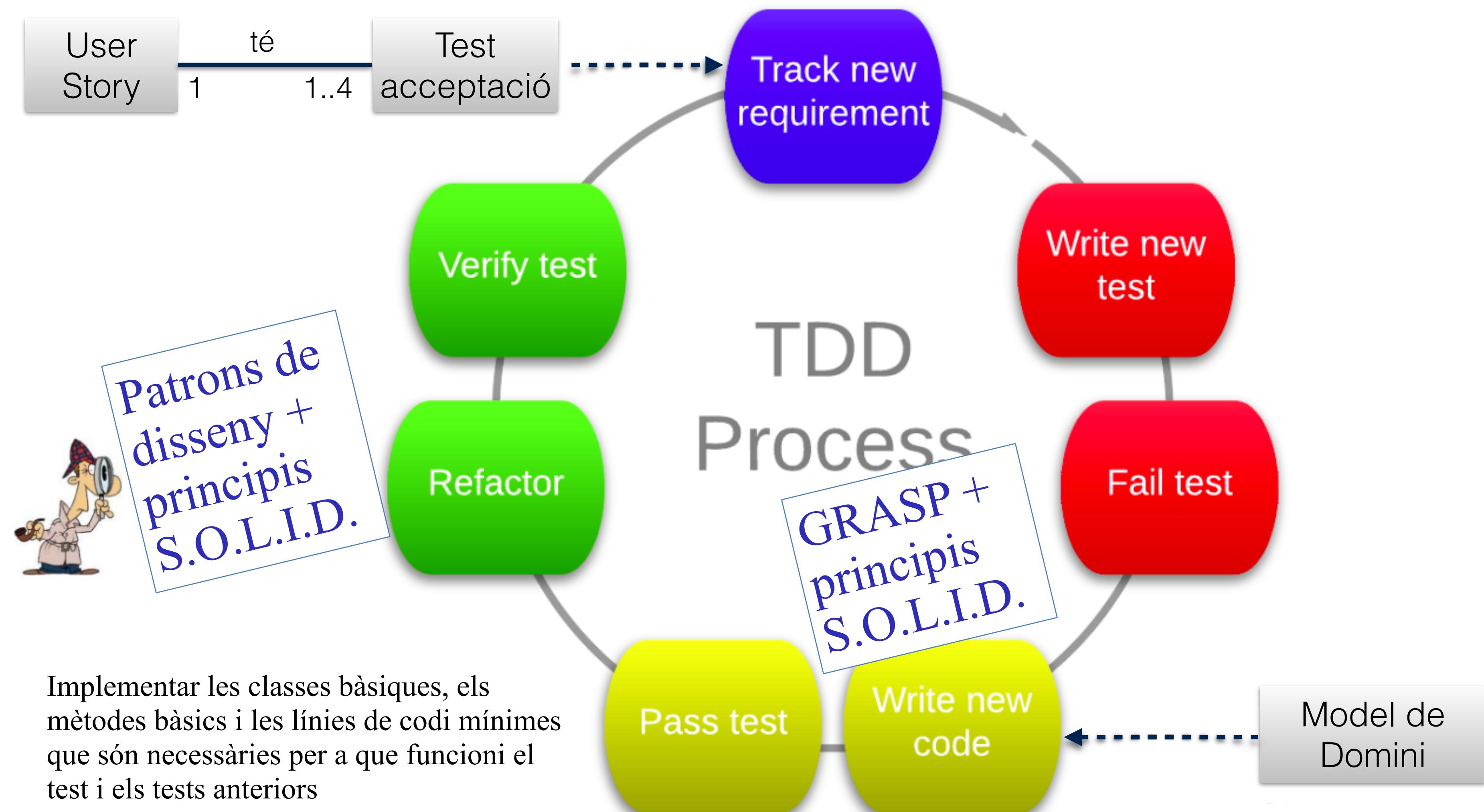


Criteris de disseny



Patrons de disseny

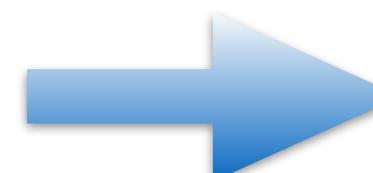
Quan s'apliquen?



3.1. Introducció

Aspectes que denoten un disseny “dolent” (*code smell*)

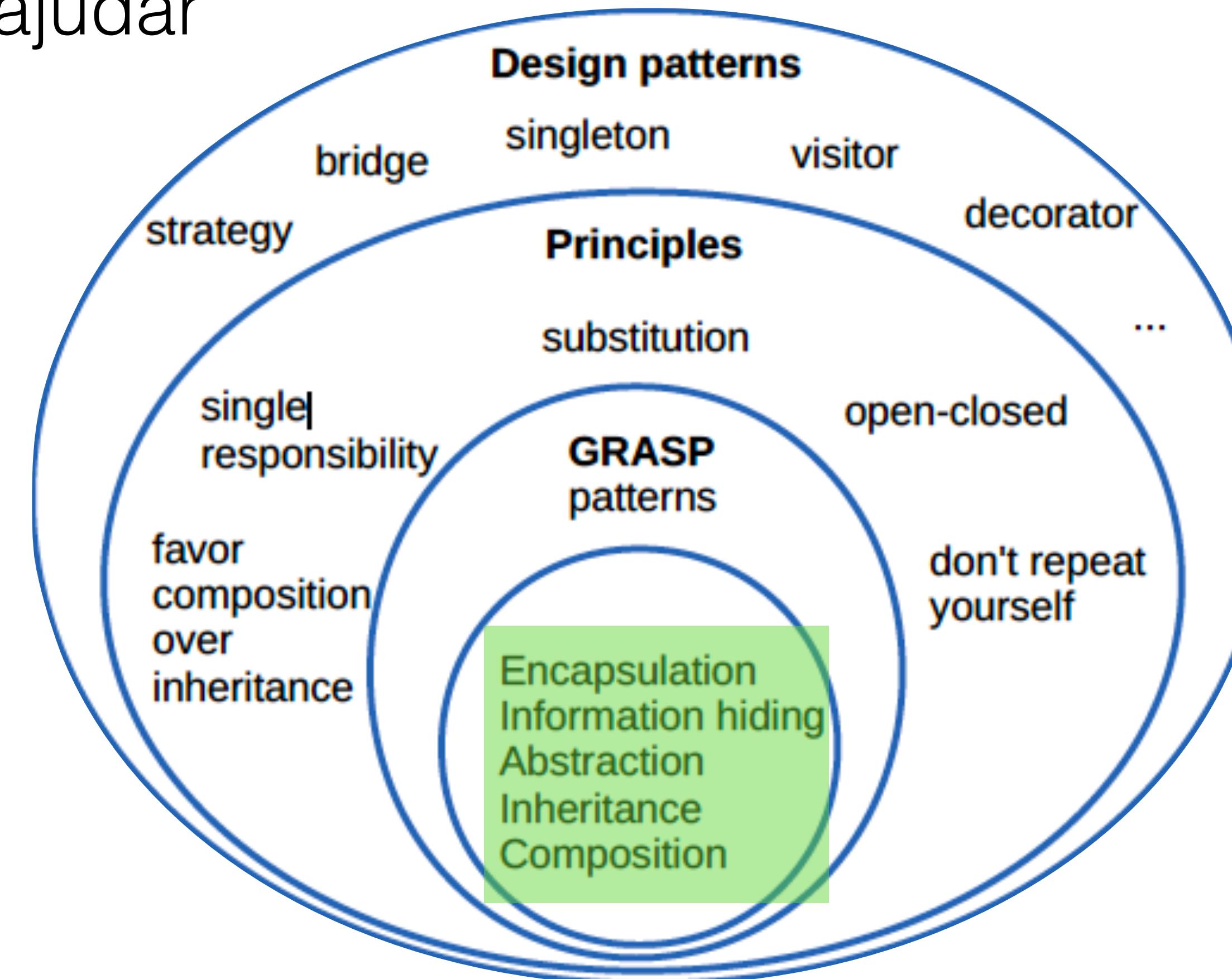
- **Rigidesa**: l'impacte d'un canvi en el software és impredecible (cada canvi produceix una cascada de canvis en moltes altres classes o el codi és tan complicat que costa entendre'l)
- **Inmobilitat**: No es pot reutilitzar codi o parts de codi
- **Fragilitat**: A cada canvi, el software es trenca en llocs on no hi han relacions conceptuais
- **Viscositat**: Impossibilitat de canviar el codi sense canviar el disseny. Provoca que fer un pedaç addicional al codi és més fàcil que no pas canviar tot el disseny



Dependències entre classes

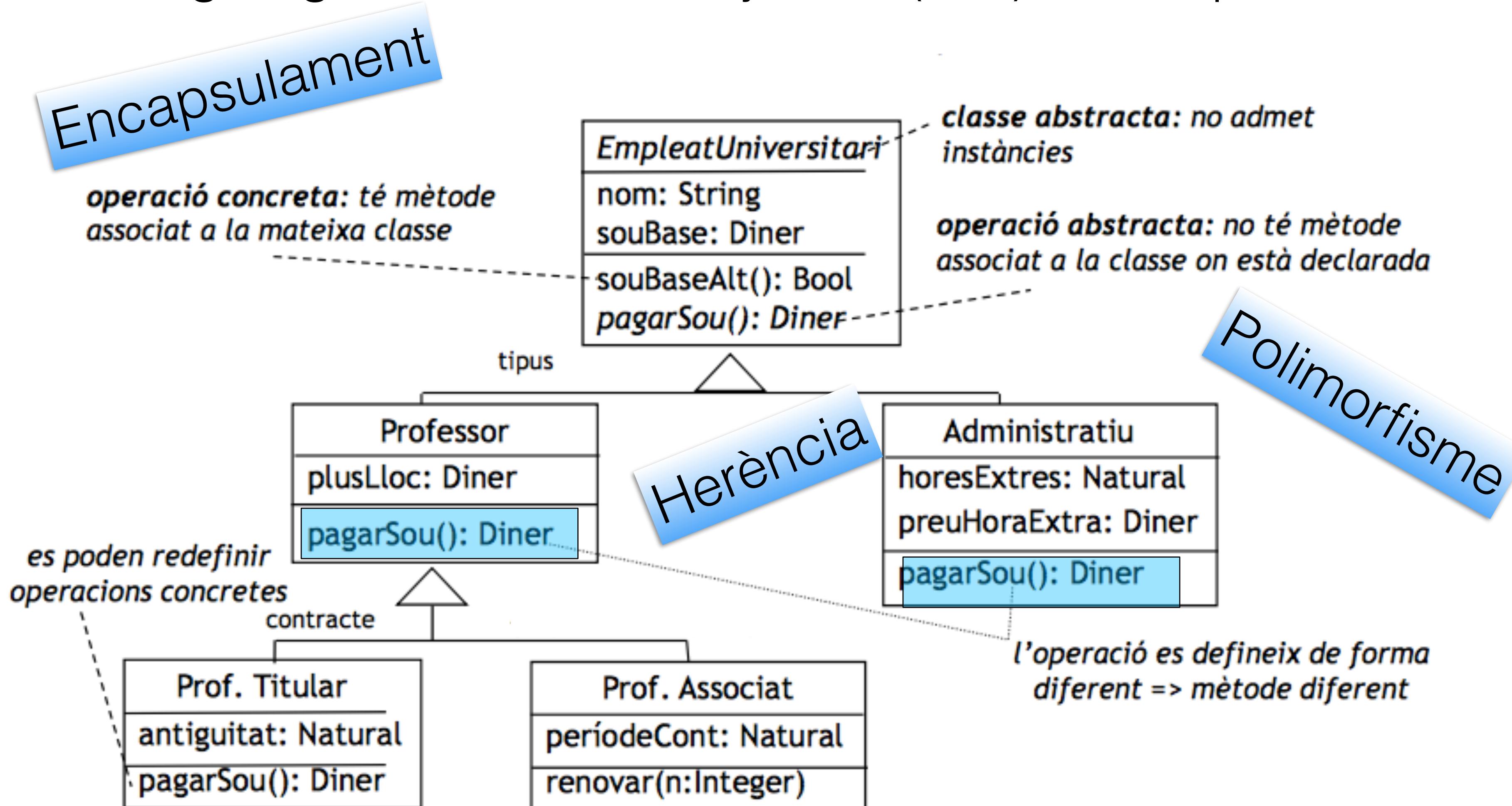
Com dissenyo en OO?

- No hi ha una metodologia que doni el millor disseny però hi han principis, heurístiques i patrons que poden ajudar



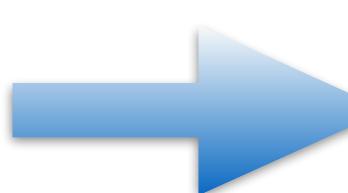
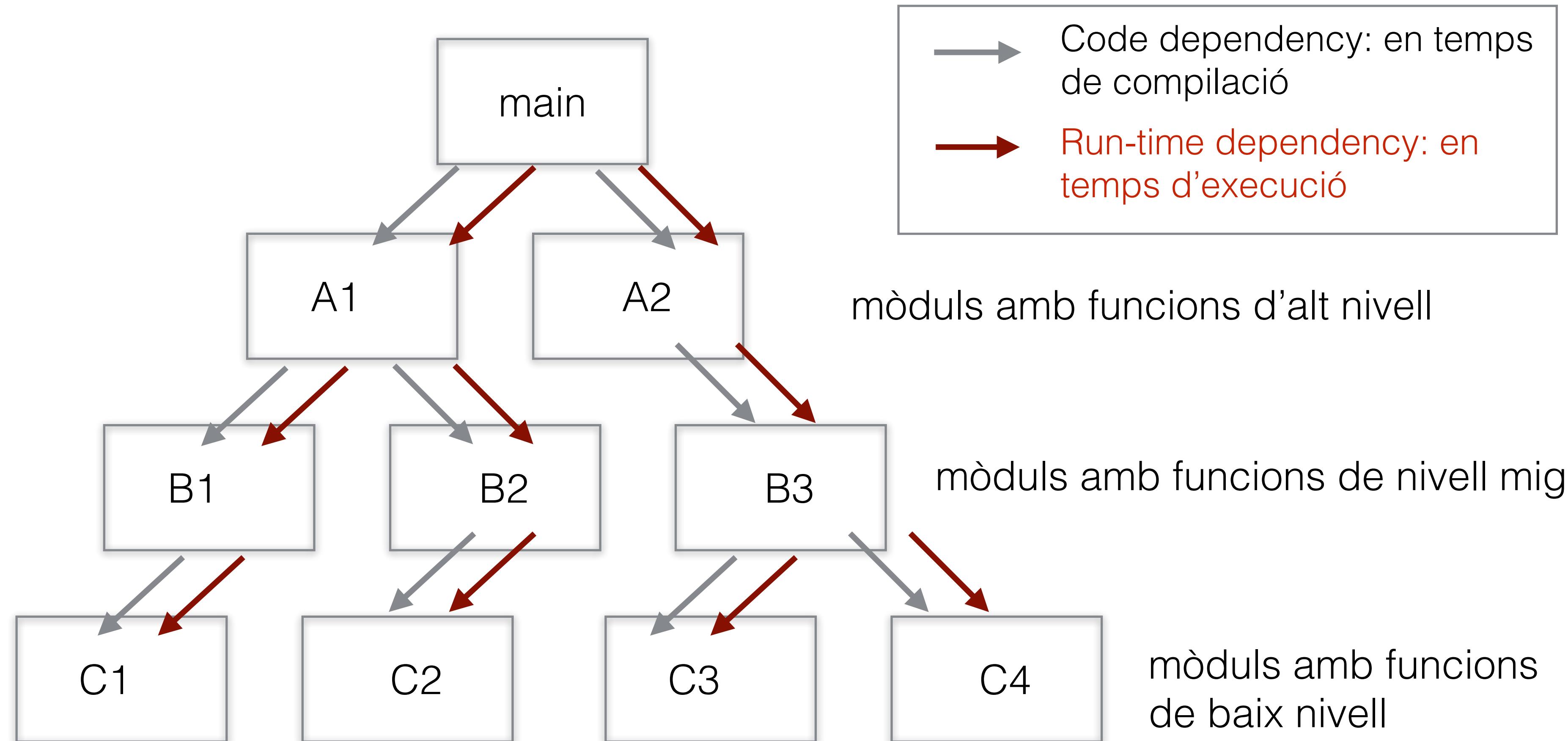
3.1. Introducció

Llenguatges Orientat a Objectes (OO): Què aporta?



3.1. Introducció

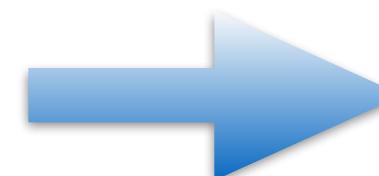
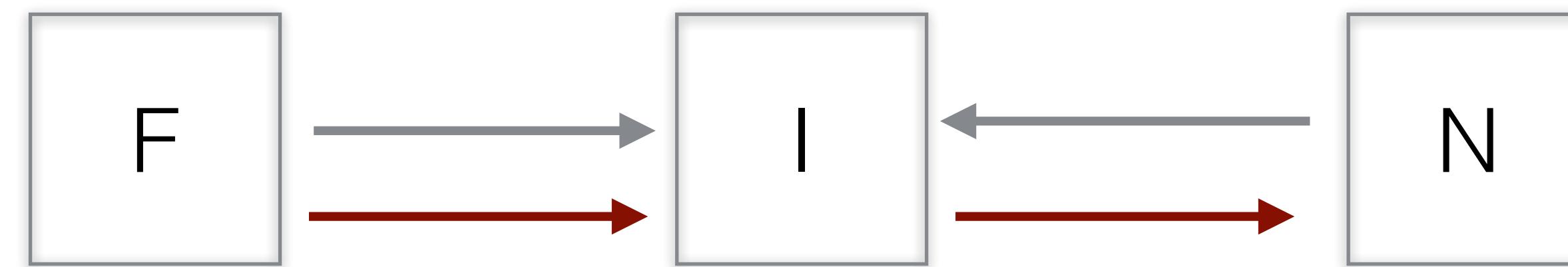
En dissenys no orientats a objectes:



El sentit de les dependències de codi és el mateix que les dependències en execució

3.1. Introducció

Aportació del disseny Orientat a Objectes

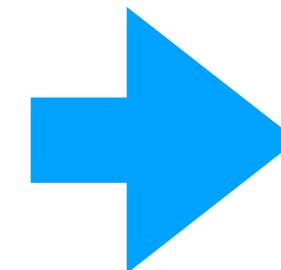


L'arquitectura d'execució no restringeix l'arquitectura del codi

Del Model de Domini al Diagrama de Classes

COM ES REPARTEIX EL CODI PER ACONSEGUITR UN BON DISSENY?

En cas que [context]
quan [event]
el sistema [resultat]



test d'acceptació
+
codi nou (si és necessari)
+
refactoring

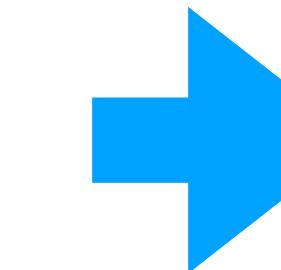
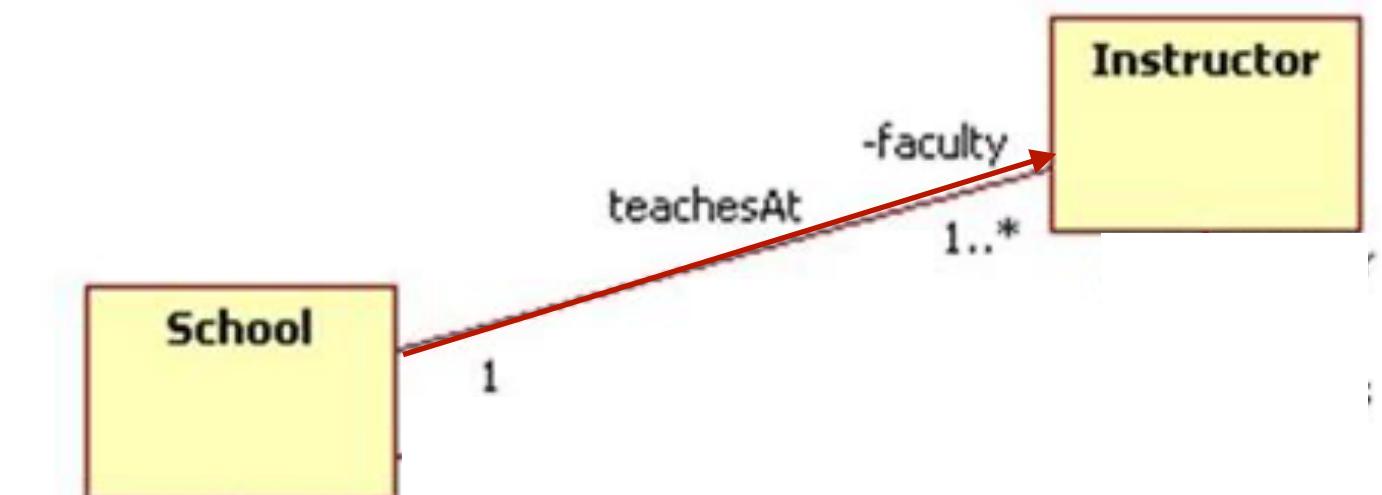
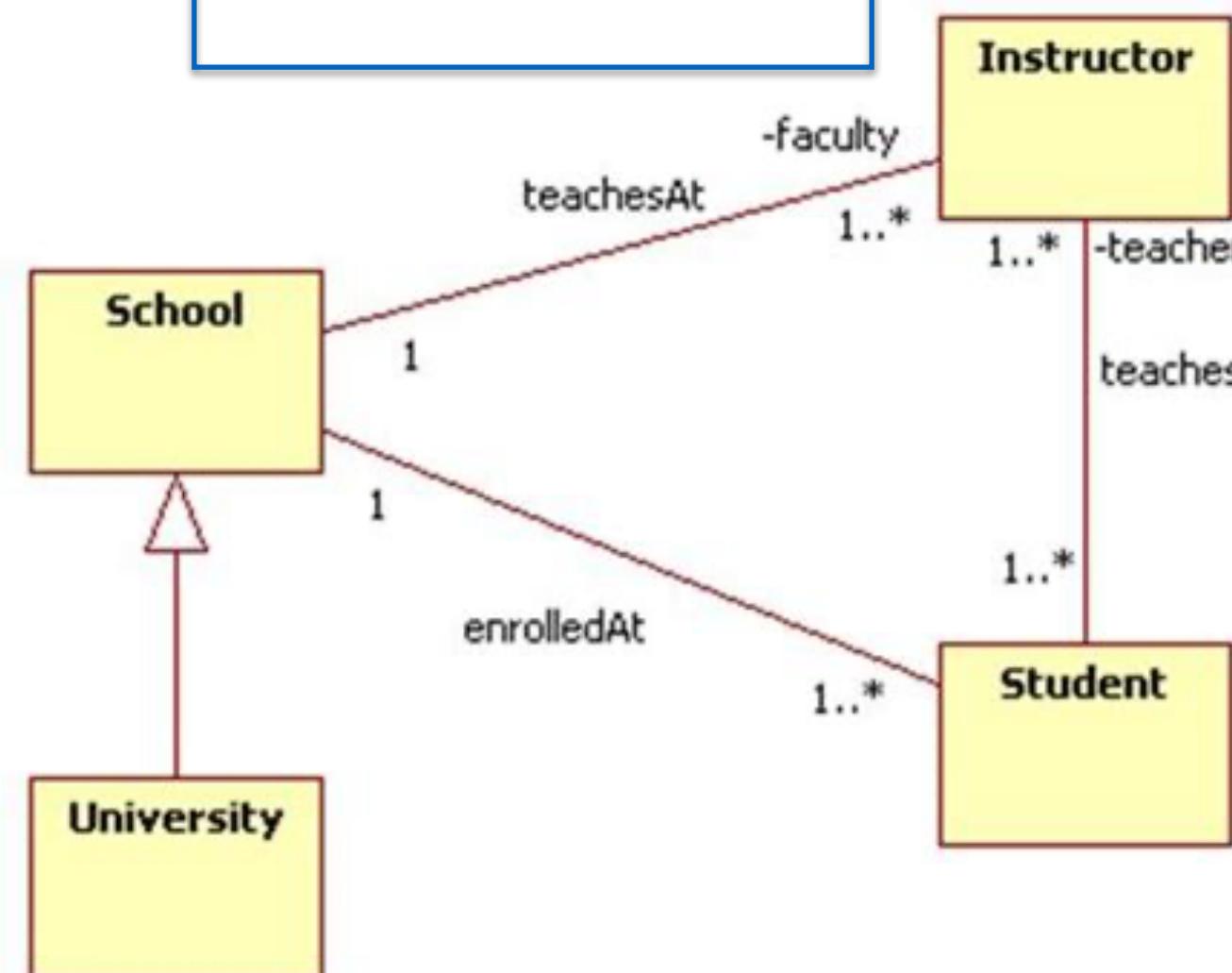


DIAGRAMA DE CLASSES

Model de Domini



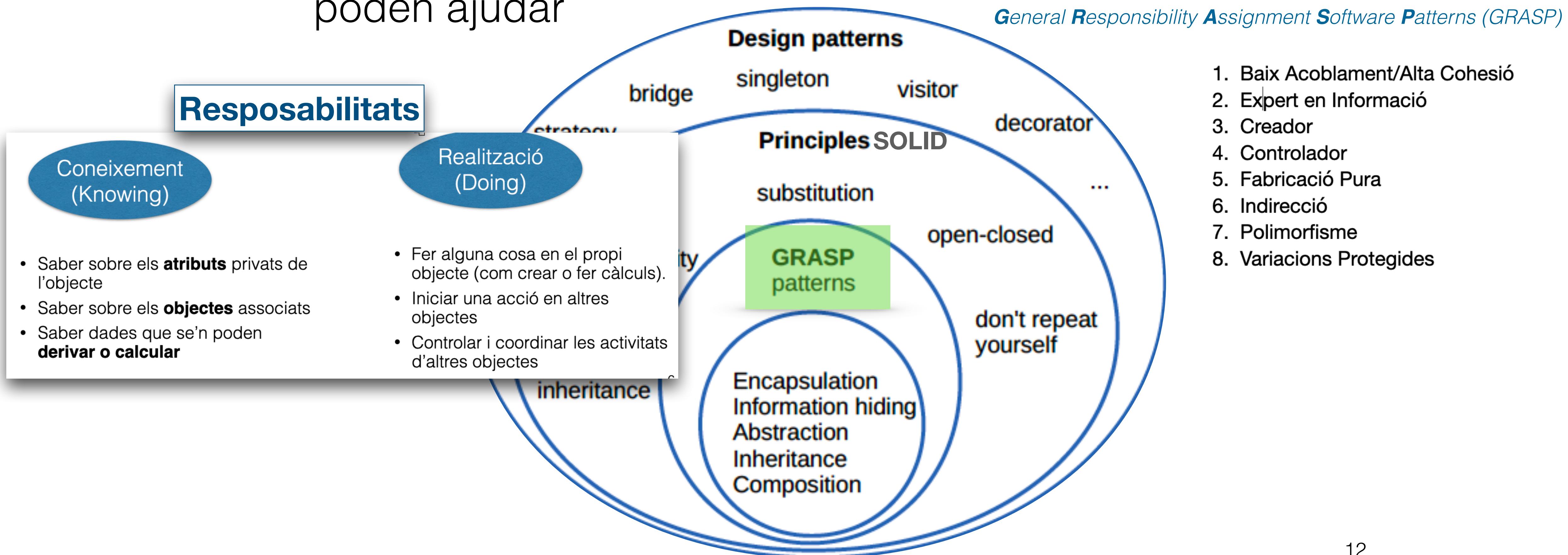
Temari

1	Introducció al procés de desenvolupament del software
2	Anàlisi de requisits i especificació
3	Disseny
4	Del disseny a la implementació
5	Ús de frameworks de testing

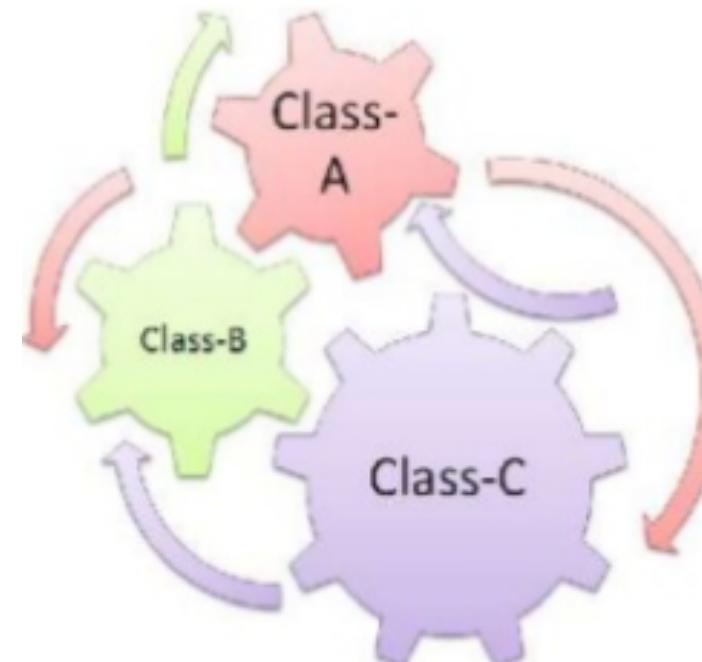
3.1	Introducció
3.2	Patrons arquitectònics
3.3	Criteris de Disseny: G.R.A.S.P.
3.4	Principis de Disseny: S.O.L.I.D.
3.5	Patrons de disseny

Com dissenyo en OO?

- No hi ha una metodologia que doni el millor disseny però hi han principis, heurístiques i patrons que poden ajudar



3.3.1.Baix acoblament/Alta cohesió



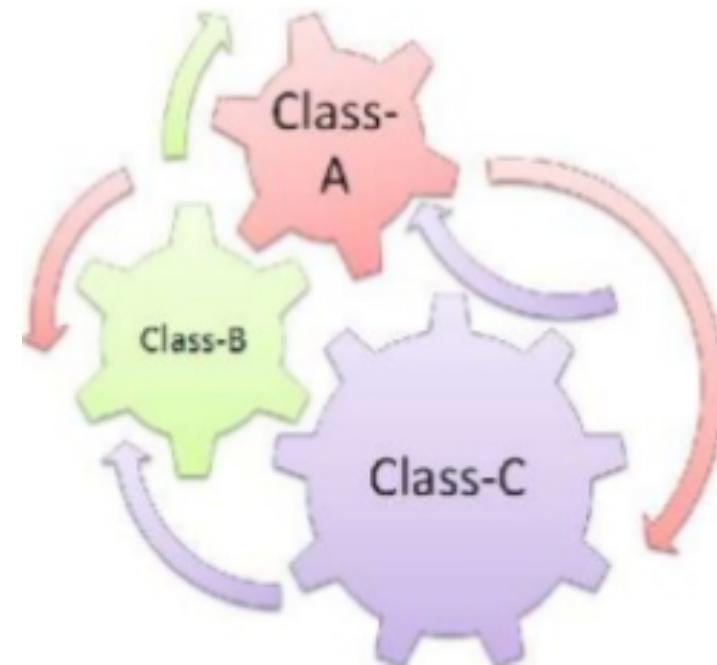
- **Acoblament:**

mesura del grau de connexió, coneixement i dependència d'una classe respecte d'altres classes.

- És necessari veure'l amb ajuda d'altres criteris o patrons
- Acoblar objectes “globals” estables no és un problema

Acoblament BAIX

3.3.1.Baix acoblament/Alta cohesió



- **Acoblament:**

mesura del grau de connexió, coneixement i dependència d'una classe respecte d'altres classes.

- És necessari veure'l amb ajuda d'altres criteris o patrons
- Acoblar objectes "globals" estables no és un problema

Acoblament BAIX



Classes amb:

- pocs mètodes
- número petit de línies de codi
- no fan gaire feina
- feina relacionada

- **Cohesió:**

mesura del grau de relació i de concentració de les diverses responsabilitats d'una classe (atributs, associacions, mètodes,...)



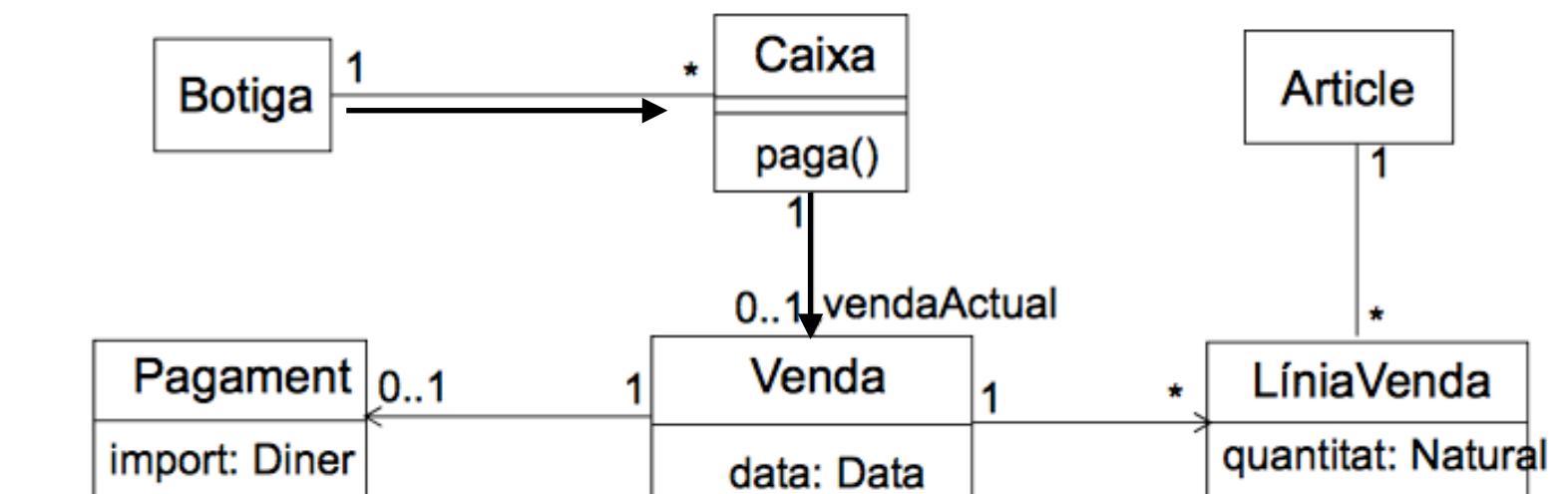
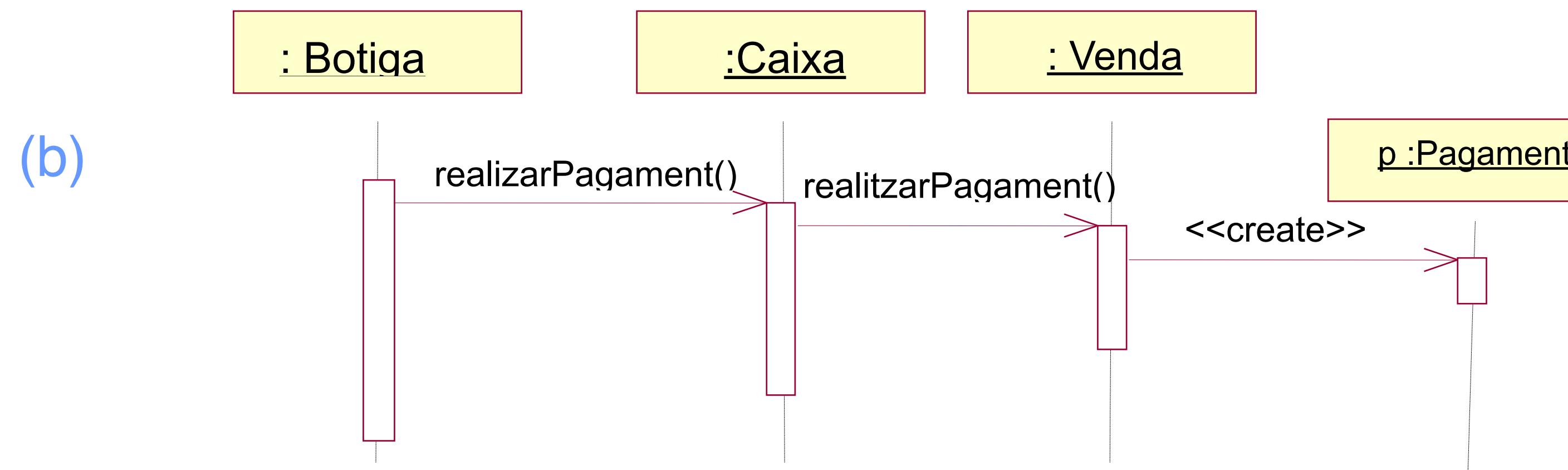
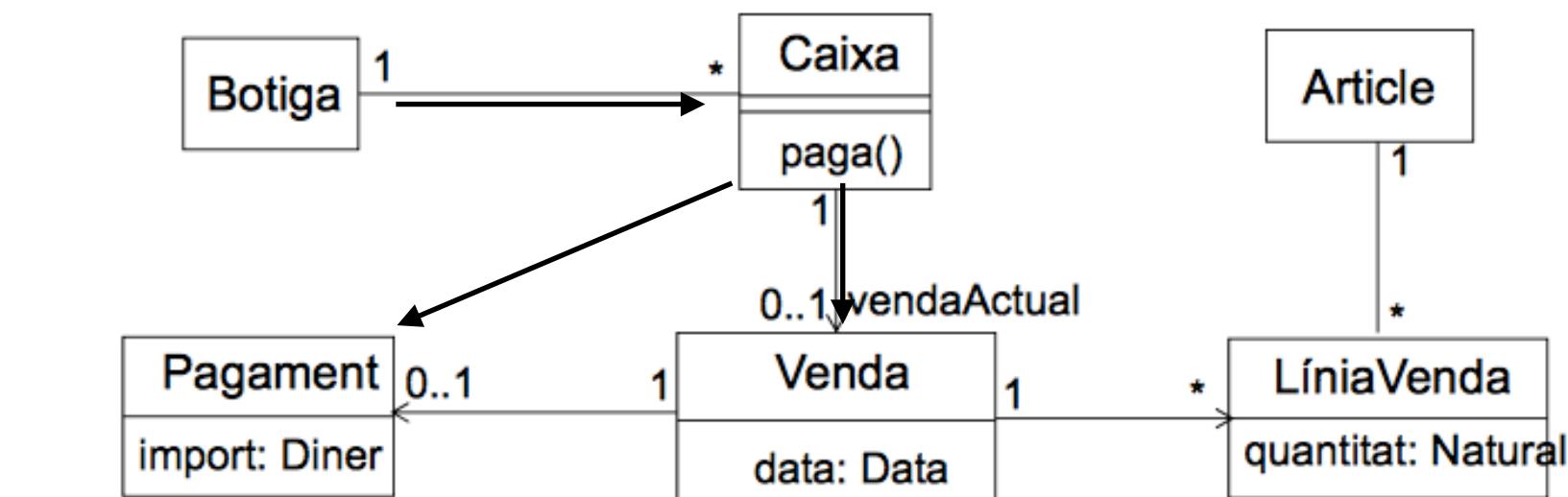
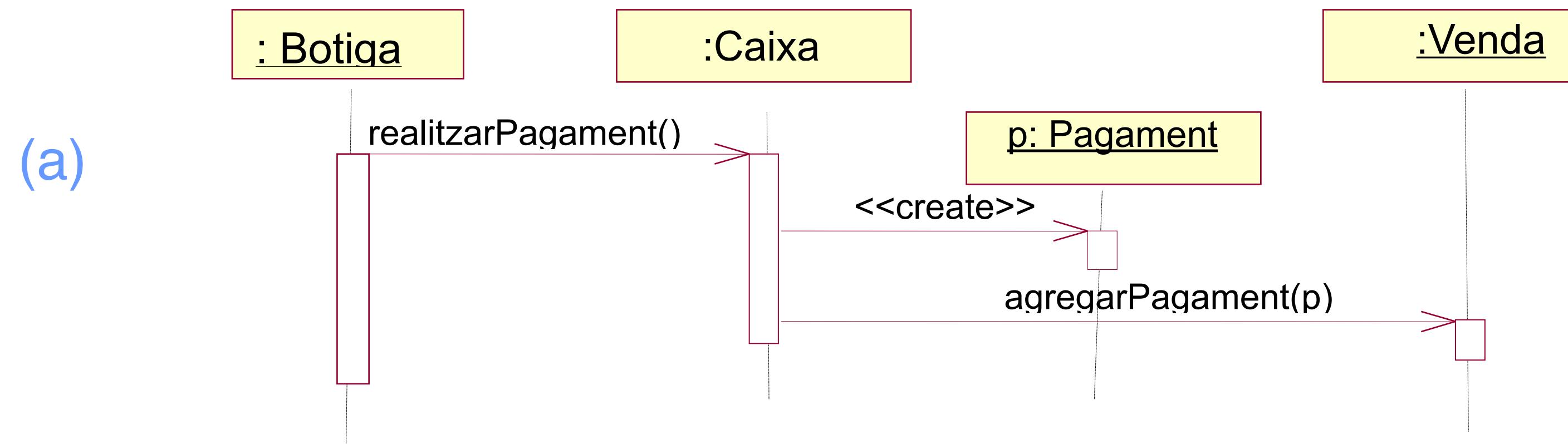
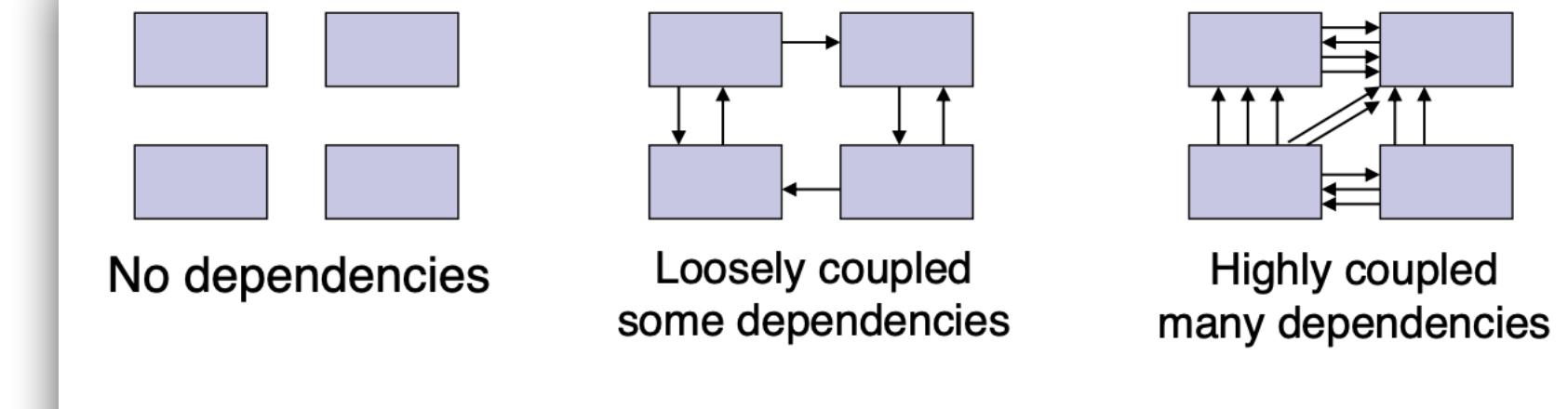
ALTA Cohesió



Pot donar lloc a més indireccions (o més crides)

Exemple d'aplicació

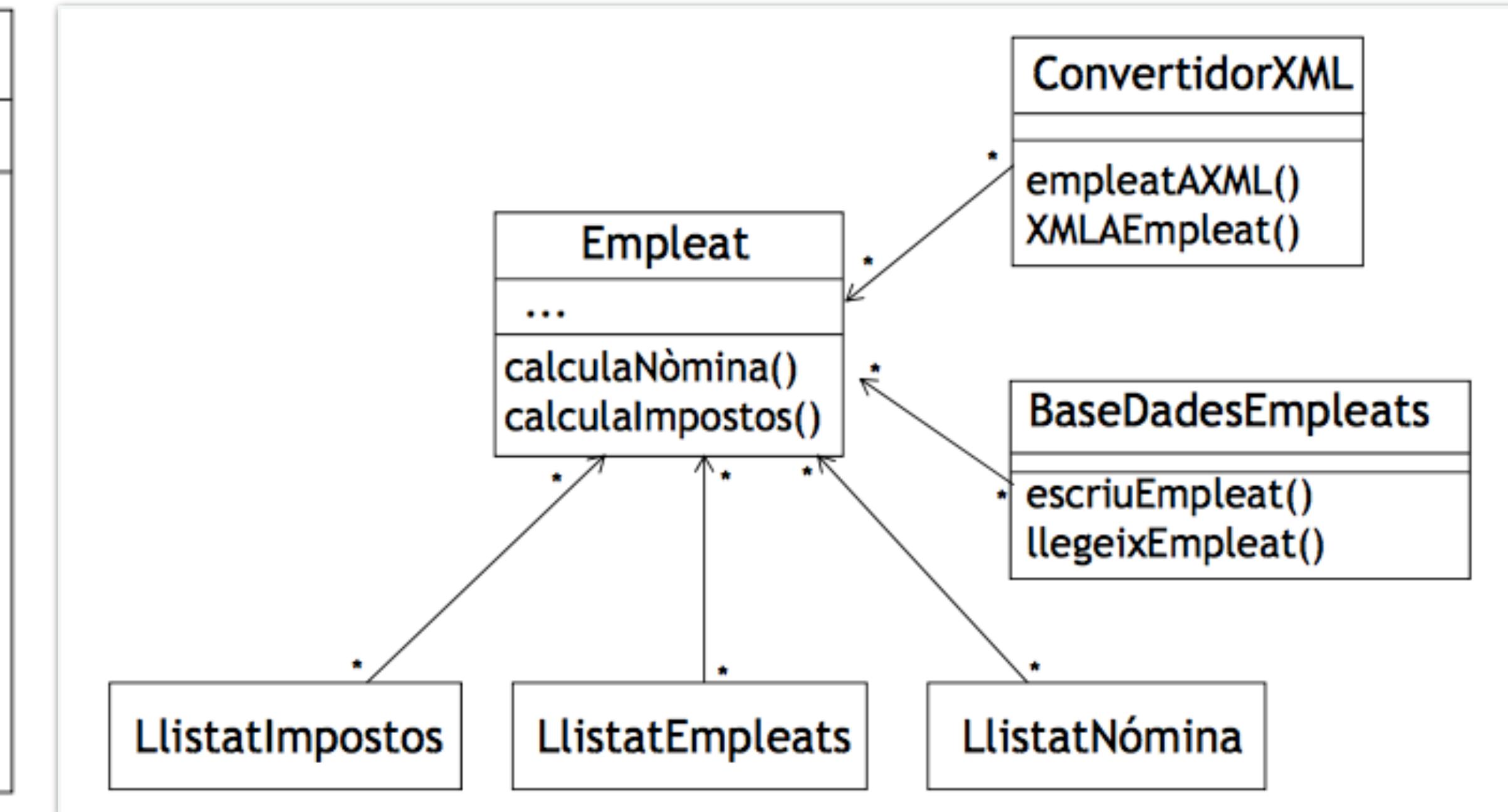
- Què podem dir d'aquests dissenys?



Exemple cohesió

Empleat
...
calculaNòmina()
calculalImpostos()
escriuADisc()
llegeixDeDisc()
creaXML()
llegeixDeXML()
mostraEnLlistatNòmina()
mostraEnLlistatImpostos()
mostraEnLlistatEmpleats()

BAIXA Cohesió



ALTA Cohesió

3.3.2. Expert en Informació

- **Expert en informació :**

Problema: a quina classe assigno una responsabilitat concreta?



Assignar una responsabilitat concreta a la classe que té la informació necessària para a complir-la

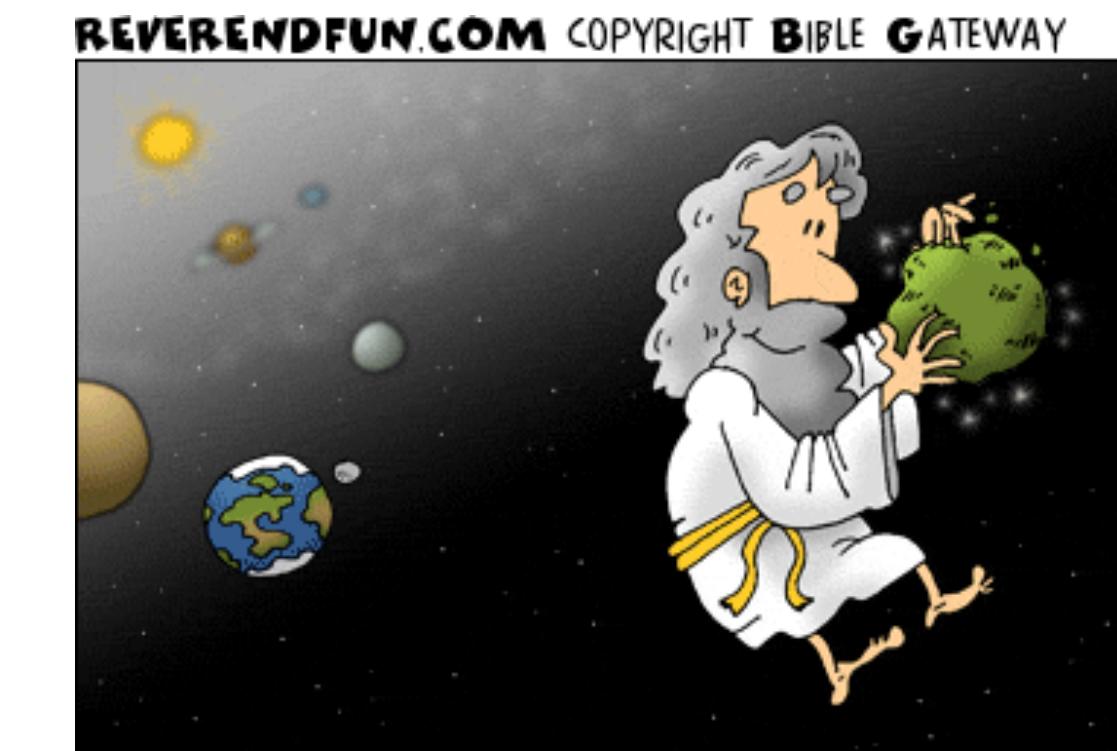


El Client vol saber la quantitat total dels productes del seu “Carrito”, qui té la responsabilitat de fer-ho?

3.3.3. Creator

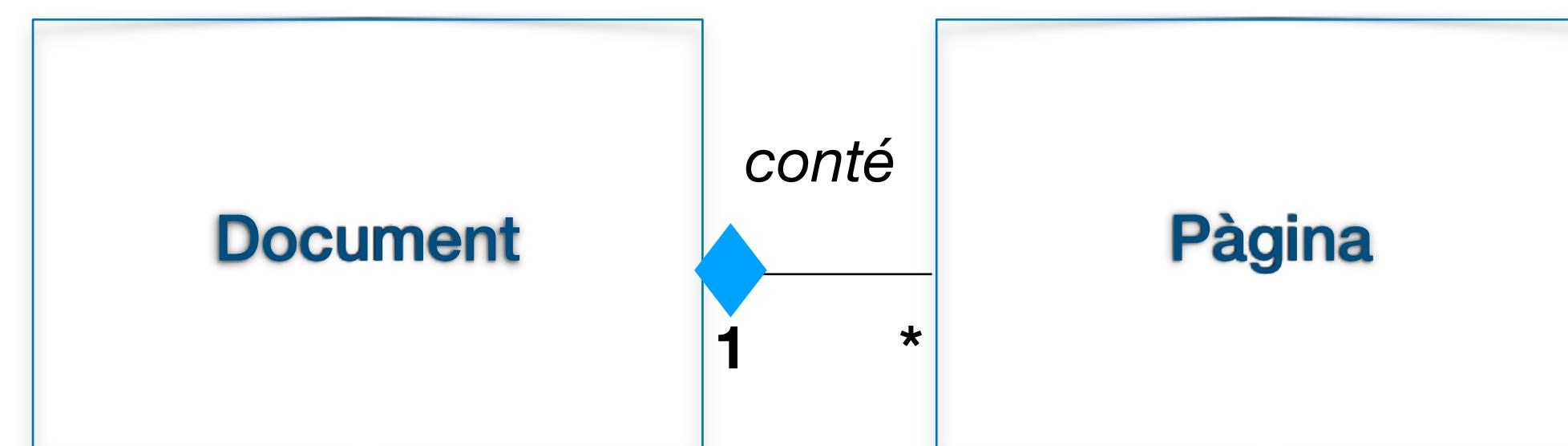
- **Creador :**

Problema: quina és la classe responsable de crear-ne una altra?



(See Genesis 1:1-2:3)
04-16-2010
AND THEN, FOR FUN, GOD CREATED ONE
WITH SUPERHEROES, UNICORNS, LIGERS,
OCEANS OF ORANGE SODA ... AND IT WAS
"COOL" MORE THAN "GOOD"

Assignació de la responsabilitat d'una classe B per a crear la classe A si B necessita està connectat a A en algun event

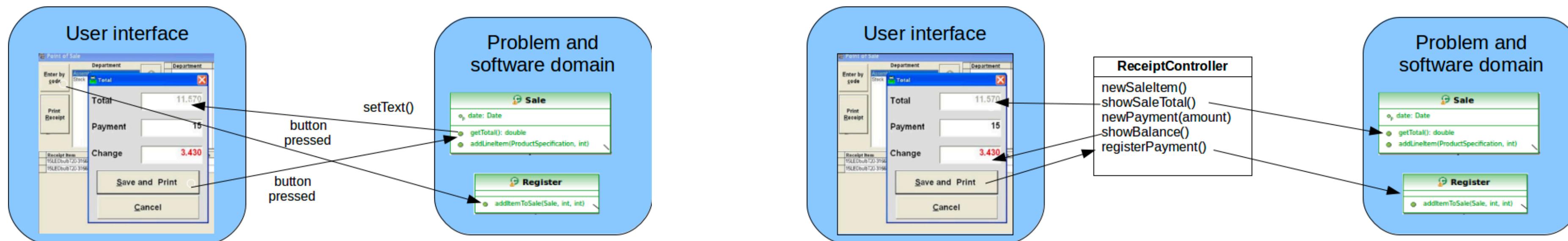


3.3.4. Controlador

- **Controlador:**

Problema: Qui és el responsable de manegar els events d'input sobre un objecte que es reben de la interfície o de la capa de presentació?

Assignació de la responsabilitat a un objecte “root” (o controlador) o bé a un “cas d'ús” dins del sistema

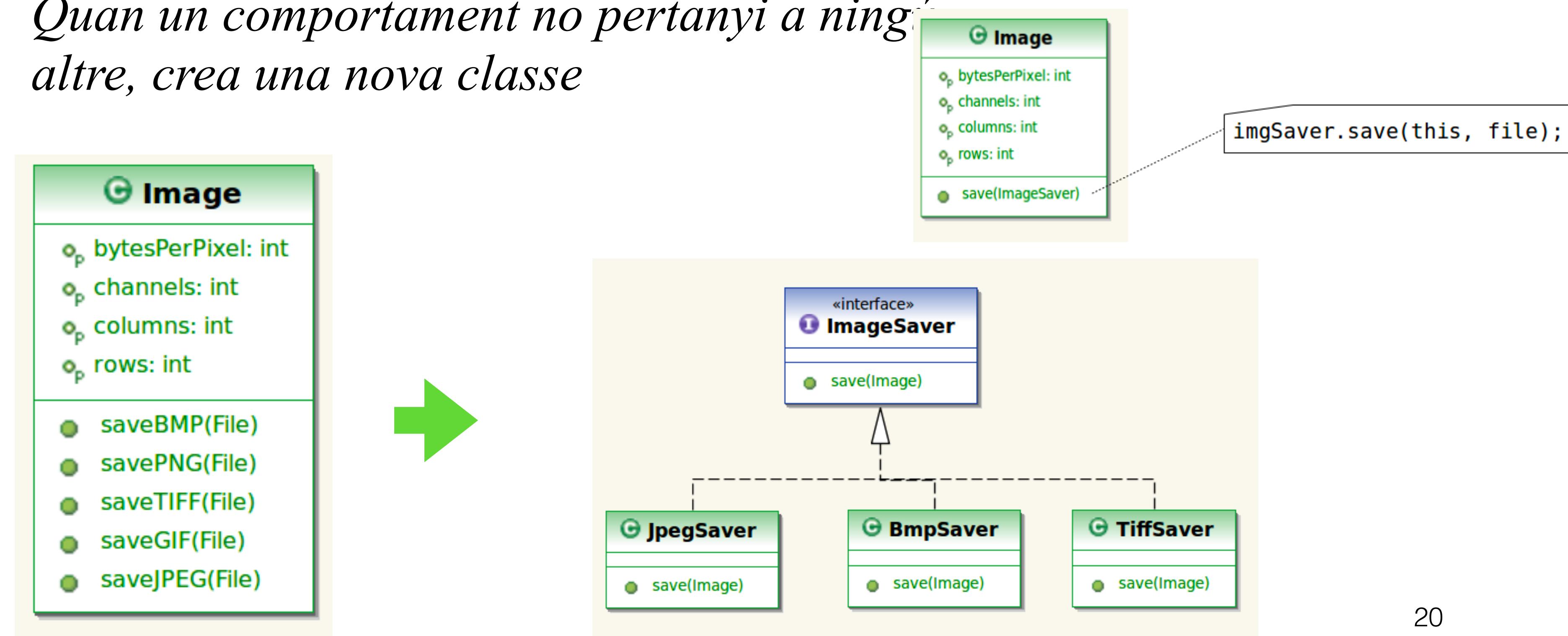


3.3.5. Fabricació Pura

- **Fabricació Pura :**

Problema: Qui és el responsable quan no hi han classes del Model de Domini que ens aconselli l'expert en informació?

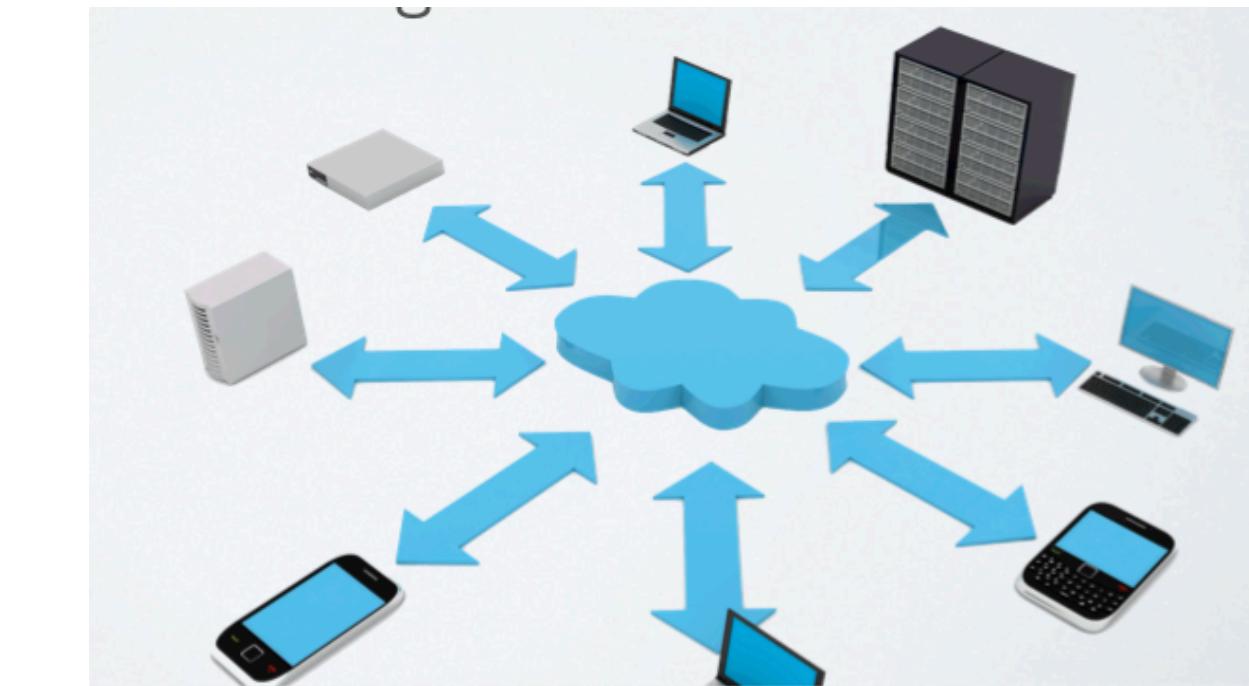
Quan un comportament no pertanyi a ningú altre, crea una nova classe



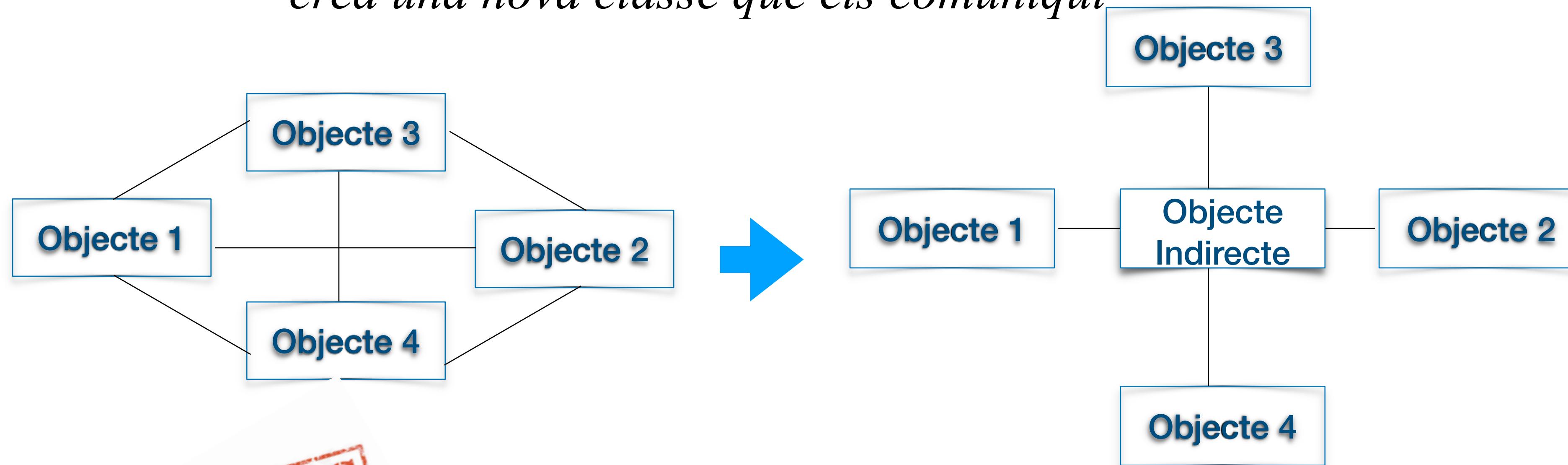
3.3.6. Indirecció

- **Indirecció :**

Problema: Com assignar responsabilitats per evitar acoblament directa entre dues o més classes?



*Per reduir l'acoblament entre objectes,
crea una nova classe que els comuniqi*



WARNING

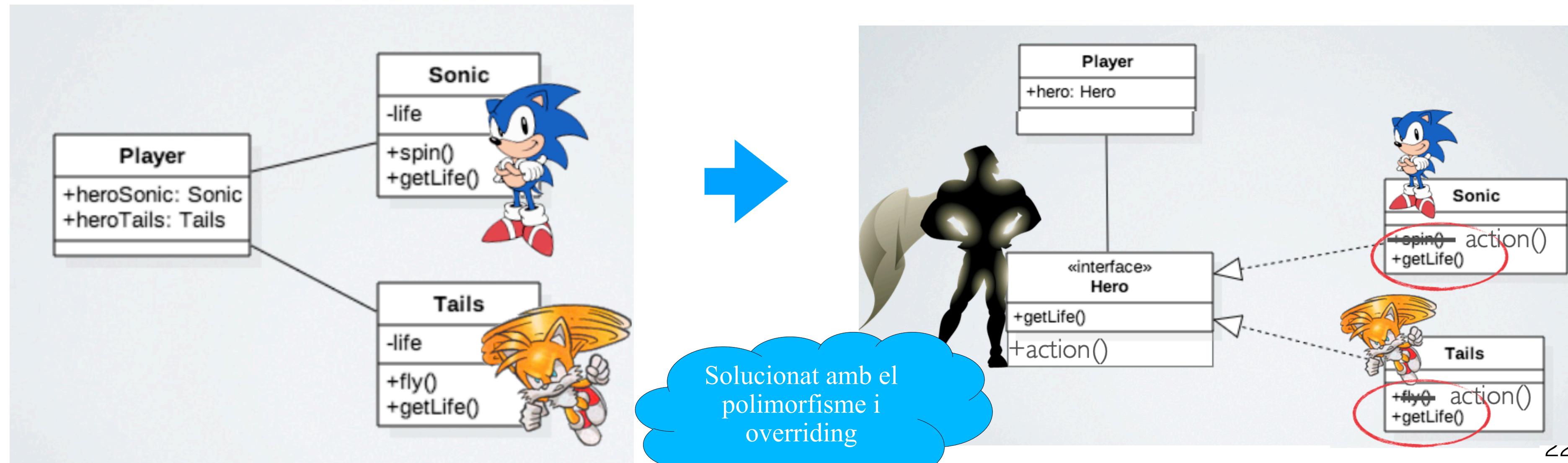
Dóna lloc a més indireccions (o més crides)

3.3.7. Polimorfisme

- **Polimorfisme :**

Problema: com manegar alternatives de comportament basades en el tipus? Com fer components “plugables”?

Permetre noms iguals a mètodes que donen el mateix servei a diferents classes (Subclasses o interfícies). Canvia automàticament el comportament segons el tipus.



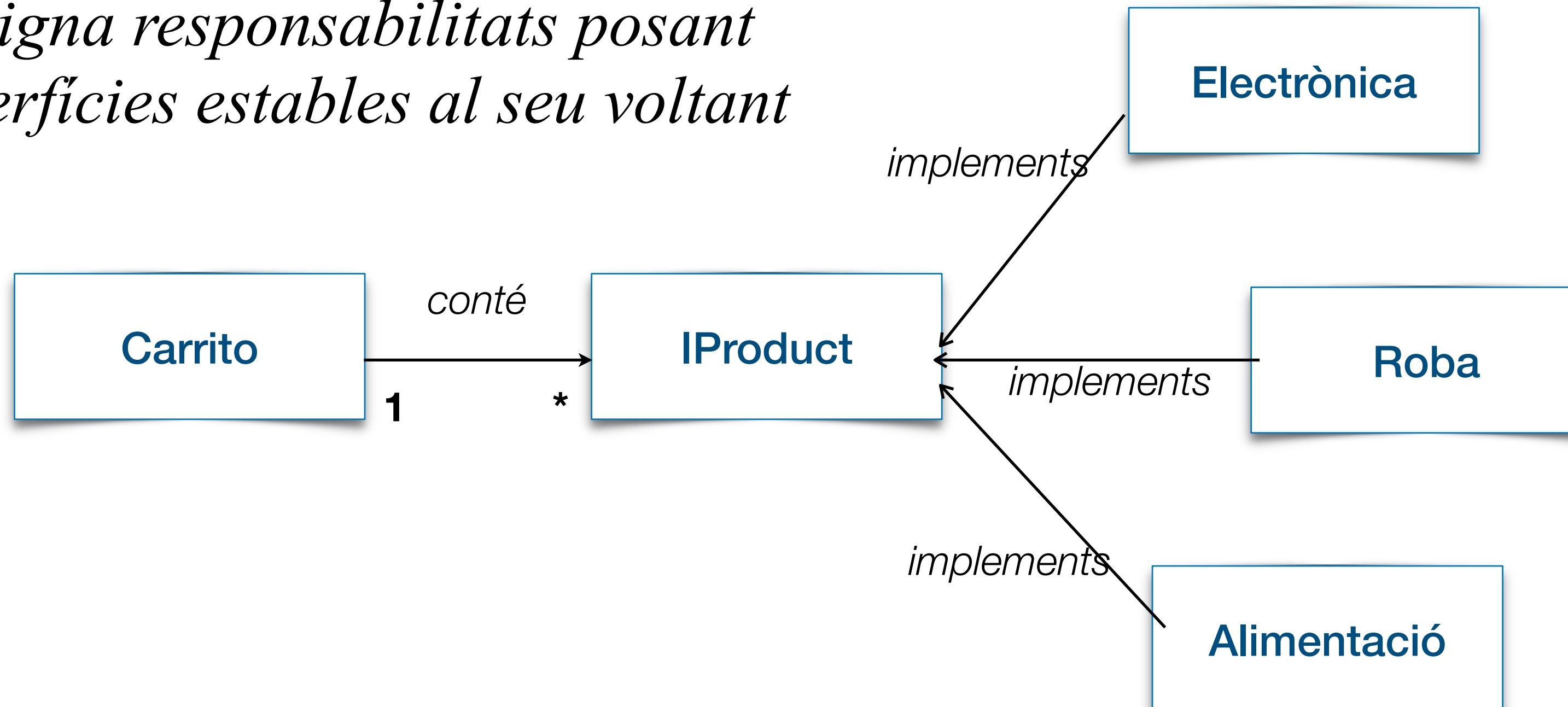
3.3.8. Variacions protegides

- **Variacions protegides :**

Problema: Com dissenyar objectes, subsistemes per a que les variacions o canvis en aquests elements no suposi un impacte no desitjat en d'altres elements?



Identifica els punts inestables i assigna responsabilitats posant interfícies estables al seu voltant



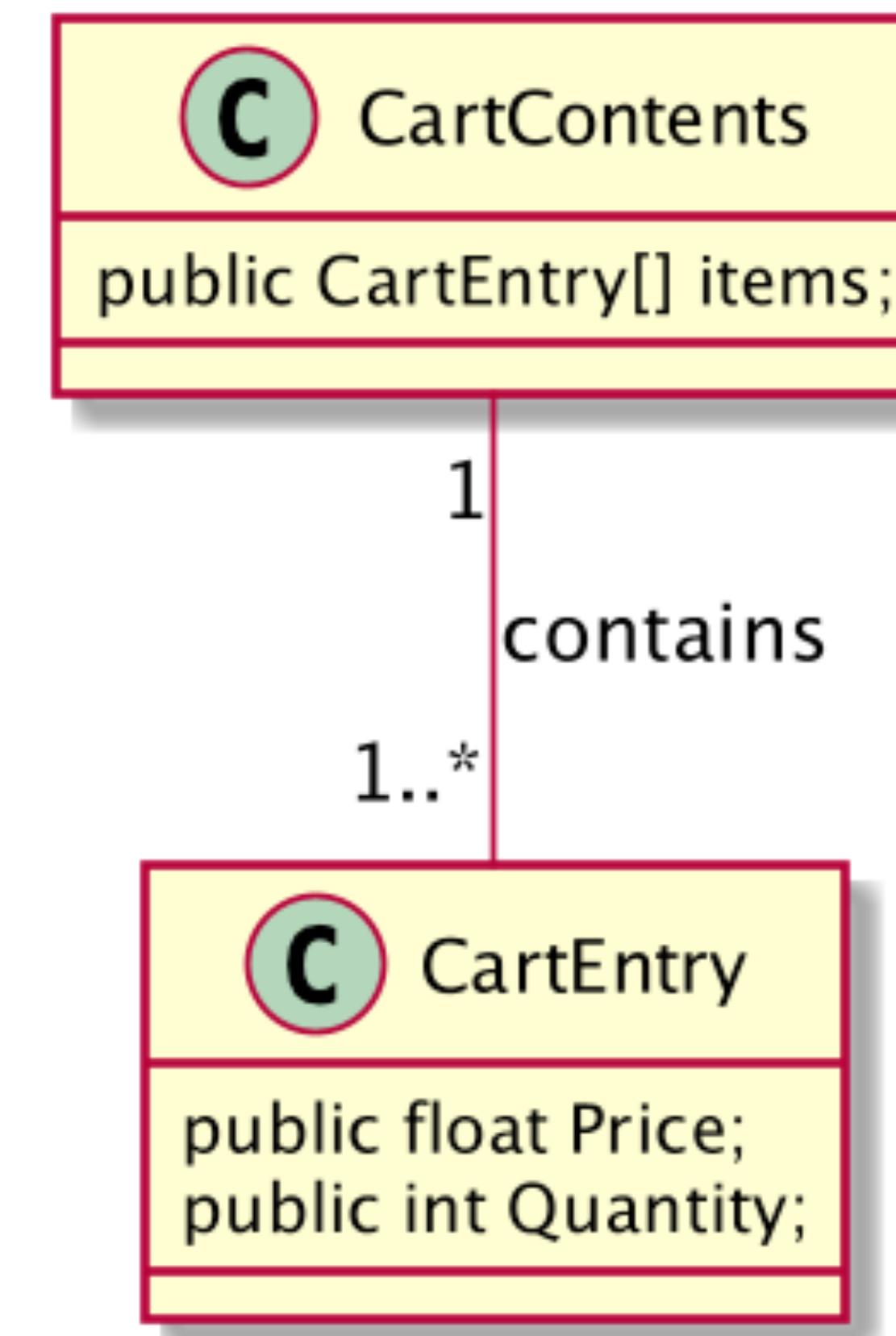


Exemple

Calcular el total d'una compra

```
public class CartEntry
{
    public float Price;
    public int Quantity;
}

public class CartContents
{
    public CartEntry[] items;
}
```





Exemple

Calcular el total d'una compra

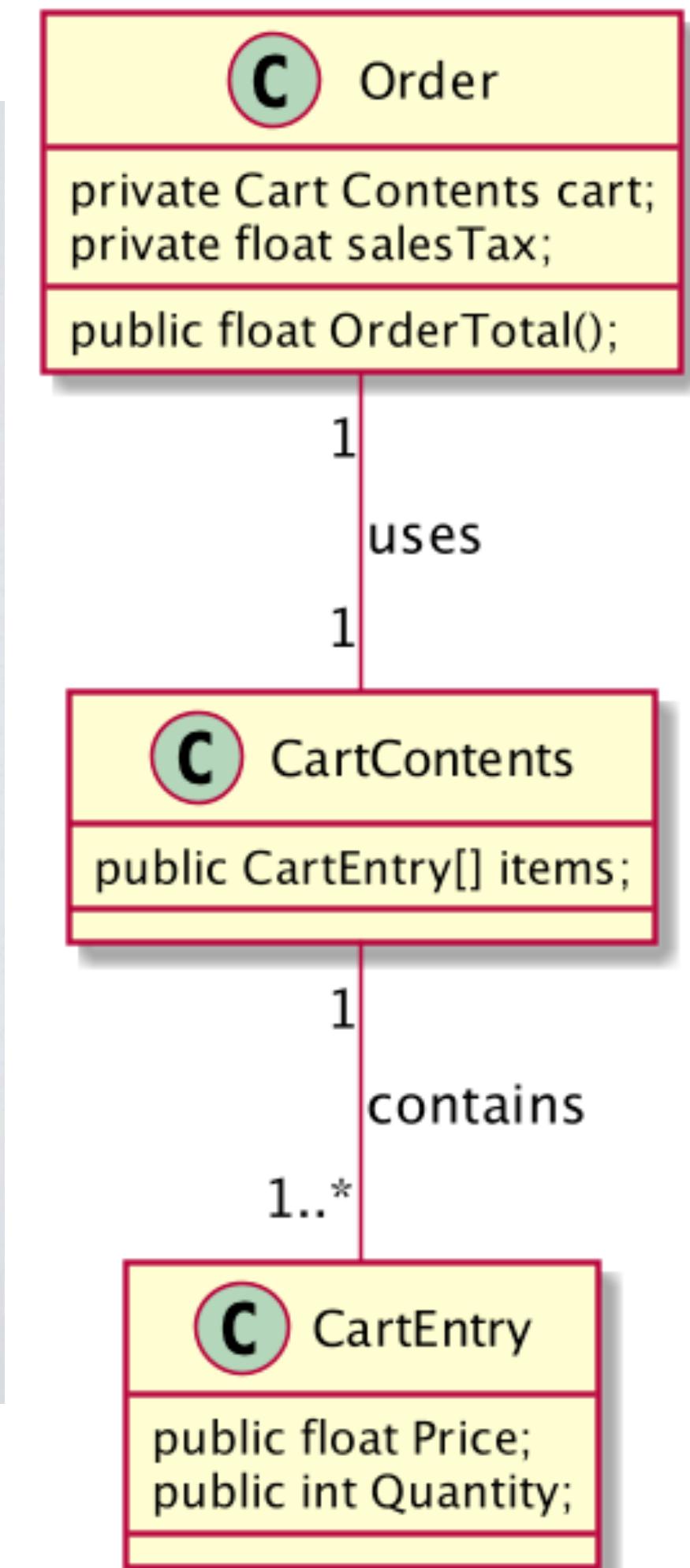
```
public class CartEntry
{
    public float Price;
    public int Quantity;
}

public class CartContents
{
    public CartEntry[] items;
}
```

```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += cart.items[i].Price * cart.items[i].Quantity;
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```





Exemple

Calcular el total d'una compra

```
public class CartEntry
{
    public float Price;
    public int Quantity;
}

public class CartContents
{
    public CartEntry[] items;
}
```

```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += cart.items[i].Price * cart.items[i].Quantity;
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```



Exemple

Calcular el total d'una compra

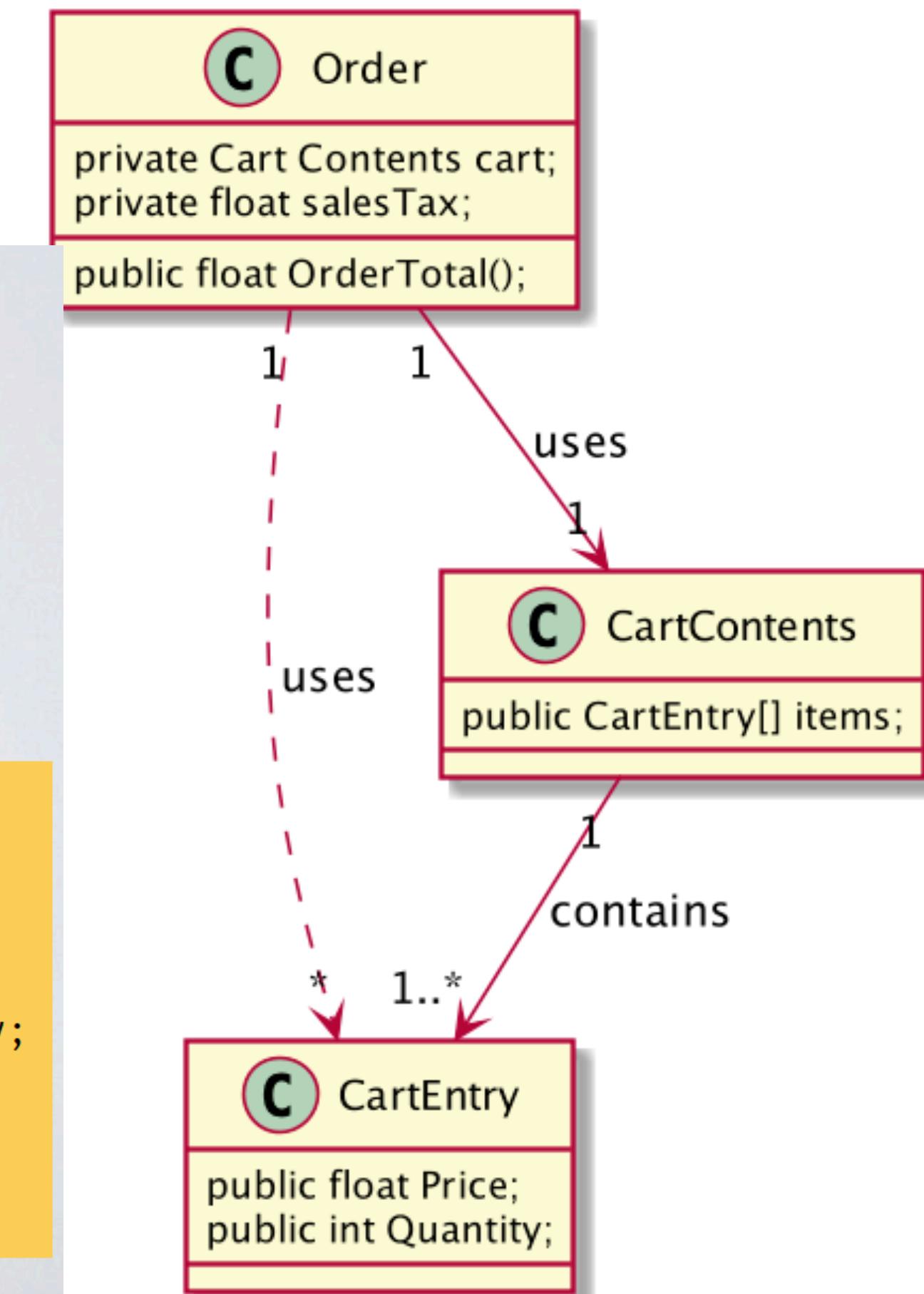
```
public class CartEntry
{
    public float Price;
    public int Quantity;
}

public class CartContents
{
    public CartEntry[] items;
}
```

```
public class Order
{
    private CartContents cart;
    private float salesTax;

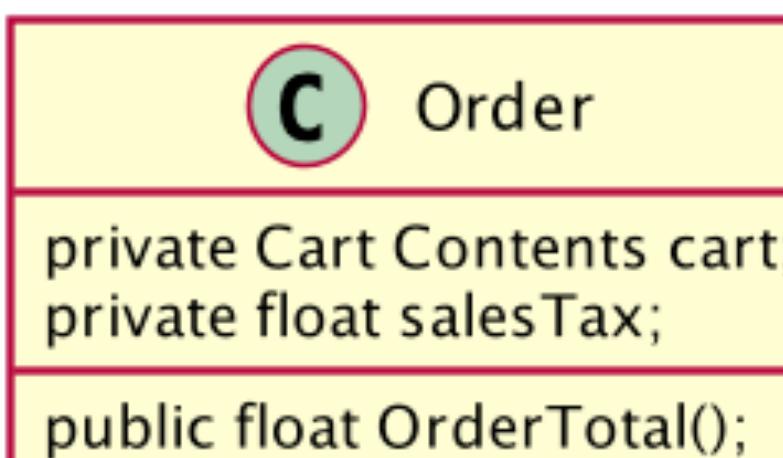
    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += cart.items[i].Price * cart.items[i].Quantity;
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```



Exemple

Qui es l'expert?



1
uses
1



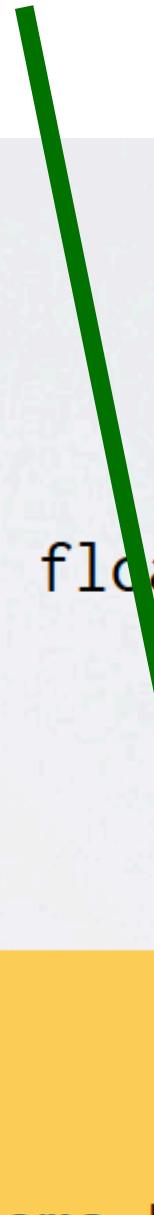
1
contains
1..*



```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += cart.items[i].Price * cart.items[i].Quantity;
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```





Exemple

Calcular el total d'una compra

Per expert en la
informació
hauria de ser
CartEntry

```
public class CartEntry
{
    private float Price;
    private int Quantity;

    public float GetLineItemTotal()
    {
        return Price * Quantity;
    }
}
```

```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

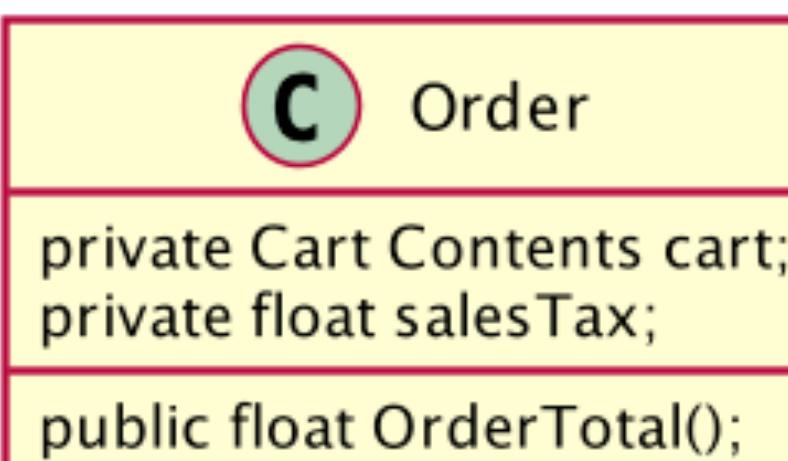
    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += item.GetLineItemTotal();
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```





Exemple

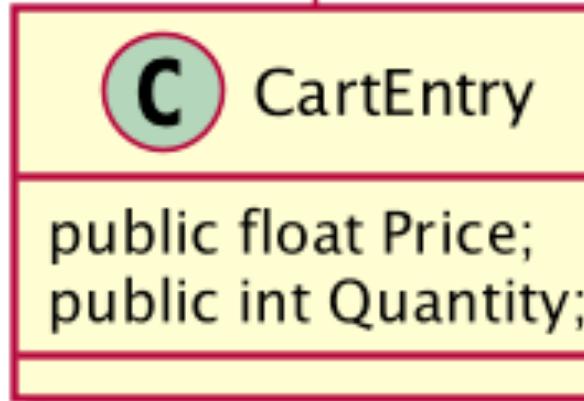
Qui es l'expert?



1
uses
1



1
contains
1..*



```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += item.GetLineItemTotal();
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```

A green arrow points from the word "contains" in the UML diagram to the line "cart.items.Length" in the code, highlighting the relationship between the two.



Exemple

Qui es l'expert?

```
public class C
{
    private fl
    private in
    public flo
    {
        return
    }
}

public class CartContents
{
    private CartEntry[] items;

    public float GetCartItemsTotal()
    {
        float cartTotal = 0;
        foreach (CartEntry item in items)
        {
            cartTotal += item.GetLineItemTotal();
        }
        return cartTotal;
    }
}
```

Per expert en la informació hauria de ser CartContents

```
public class Order
{
    private CartContents cart;
    private float salesTax;

    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

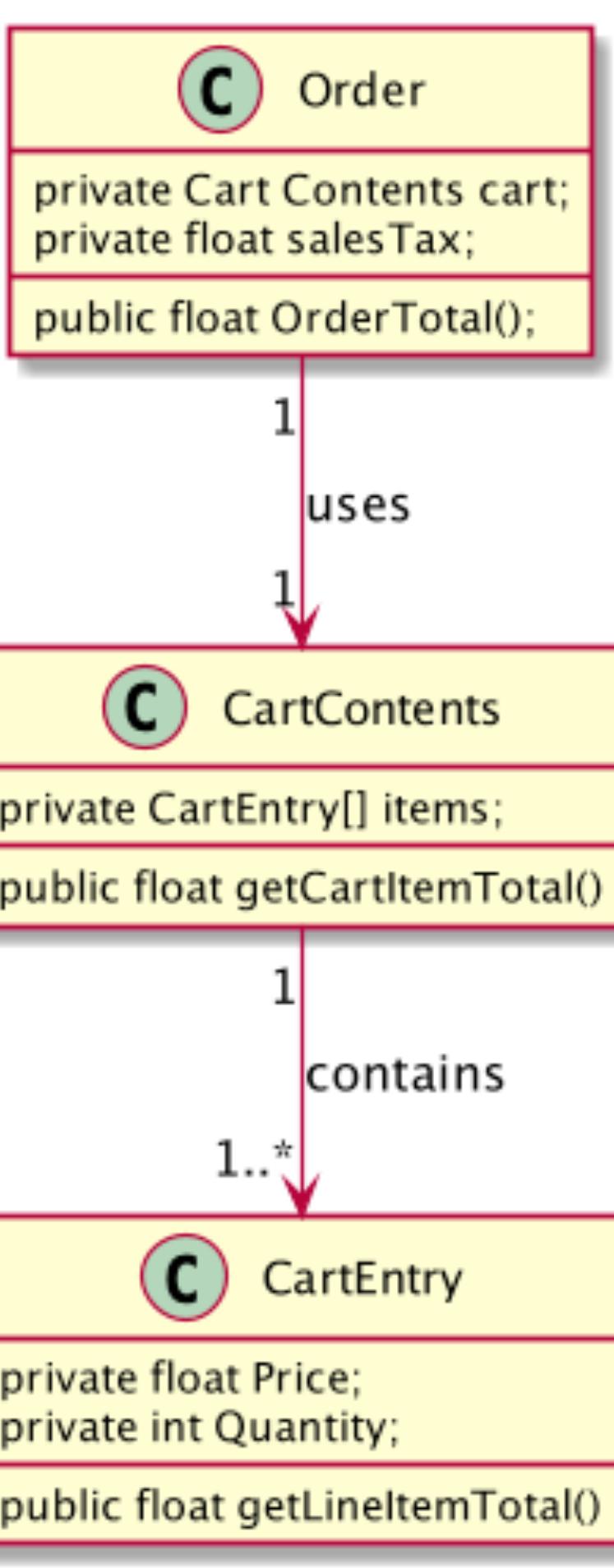
    public float OrderTotal()
    {
        float cartTotal = 0;
        for (int i = 0; i < cart.items.Length; i++)
        {
            cartTotal += item.GetLineItemTotal();
        }
        cartTotal += cartTotal*salesTax;
        return cartTotal;
    }
}
```





Exemple

Calcular el total d'una compra



```
public class CartEntry
{
    private float Price;
    private int Quantity;

    public float GetLineItemTotal()
    {
        return Price * Quantity;
    }
}

public class CartContents
{
    private CartEntry[] items;

    public float GetCartItemsTotal()
    {
        float cartTotal = 0;
        foreach (CartEntry item in items)
        {
            cartTotal += item.GetLineItemTotal();
        }
        return cartTotal;
    }
}
```

```
public class Order
{
    private CartContents cart;
    private float salesTax;

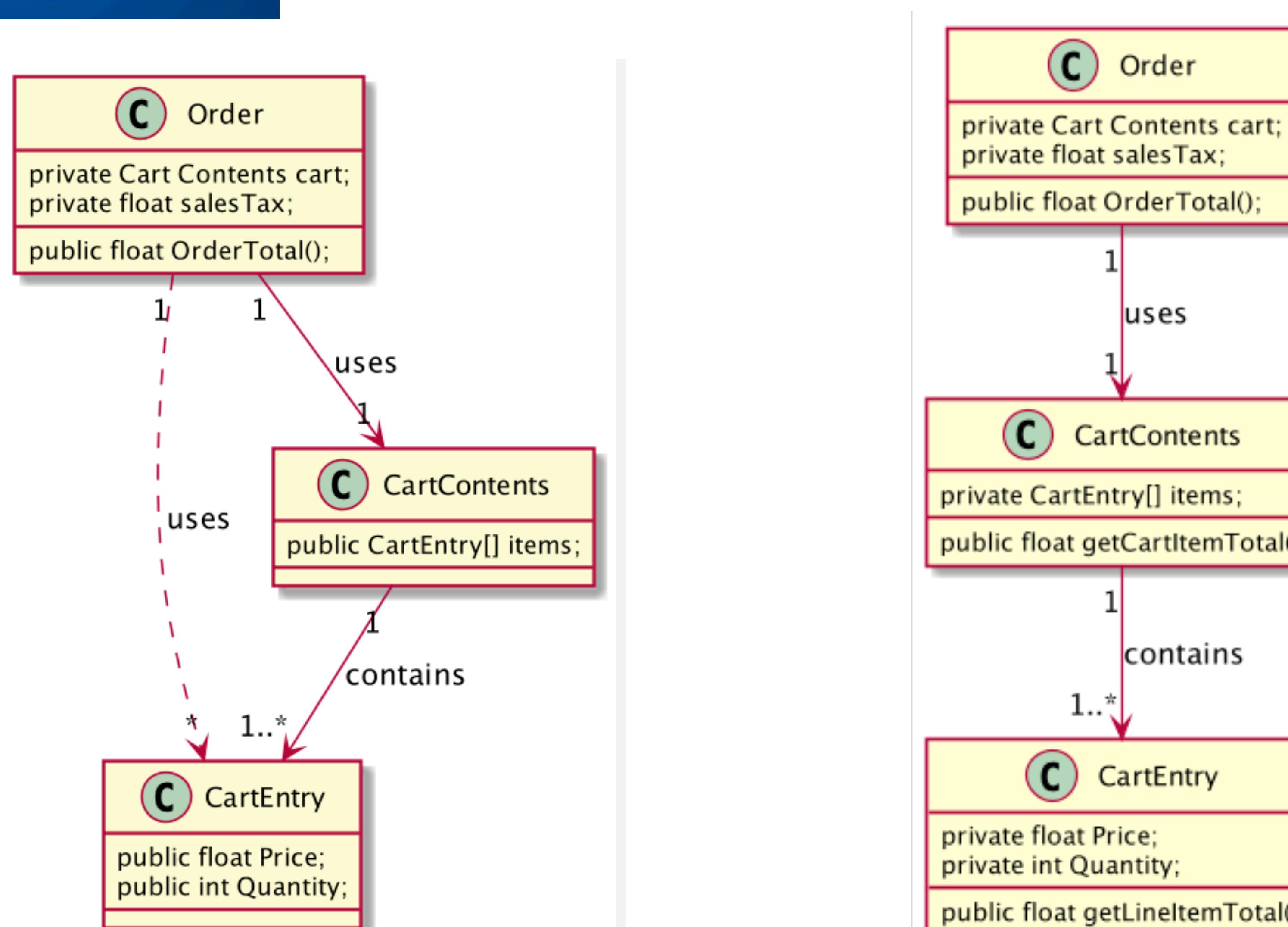
    public Order(CartContents cart, float salesTax)
    {
        this.cart = cart;
        this.salesTax = salesTax;
    }

    public float OrderTotal()
    {
        return cart.GetCartItemsTotal() *
               (1.0f + salesTax);
    }
}
```



Exemple

Calcular el total d'una compra



Troba els Creadors:

- B “conté” A
- B guarda A
- B usa A
- B té dades per construir A

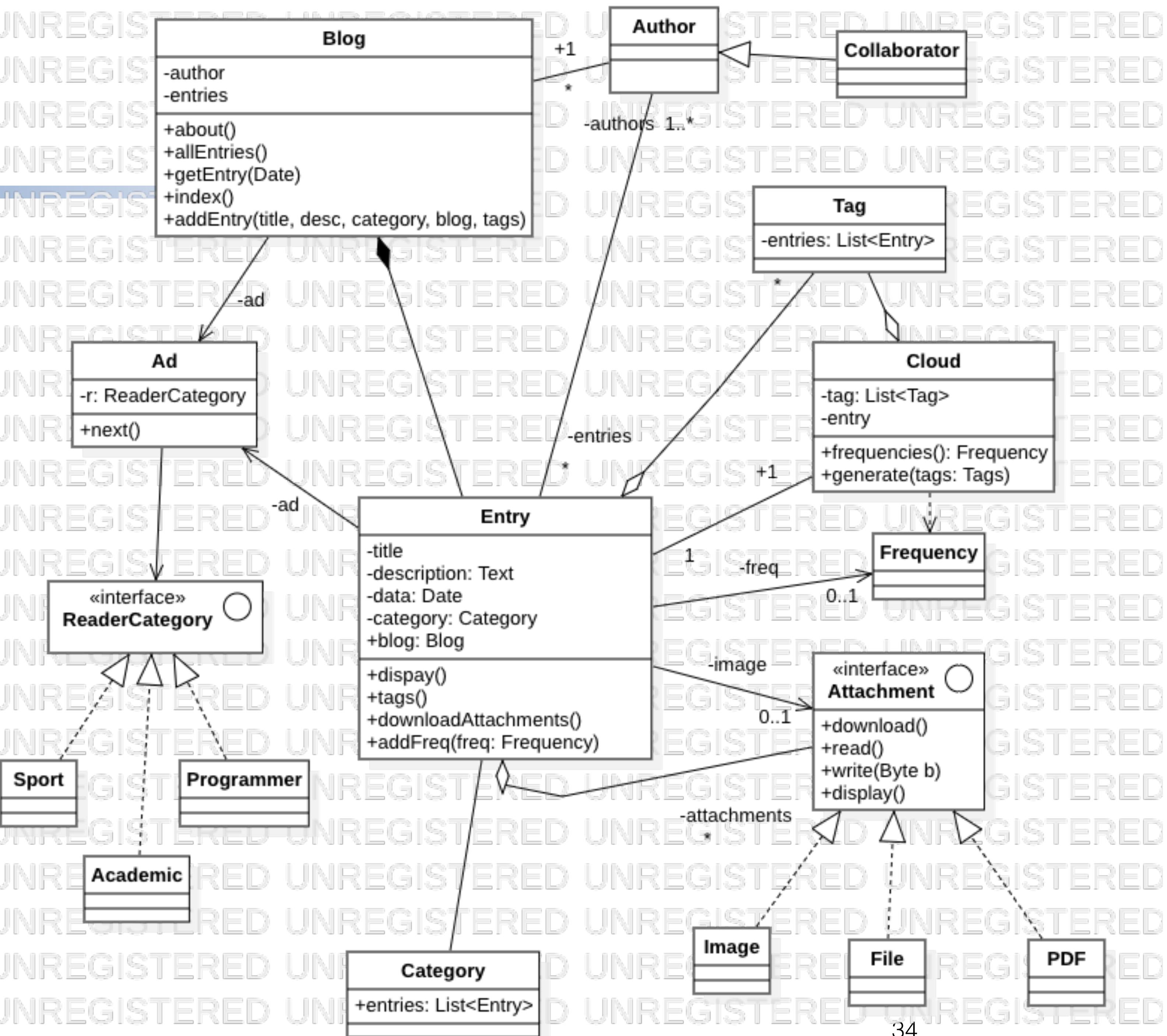
Qui crea Attachment?

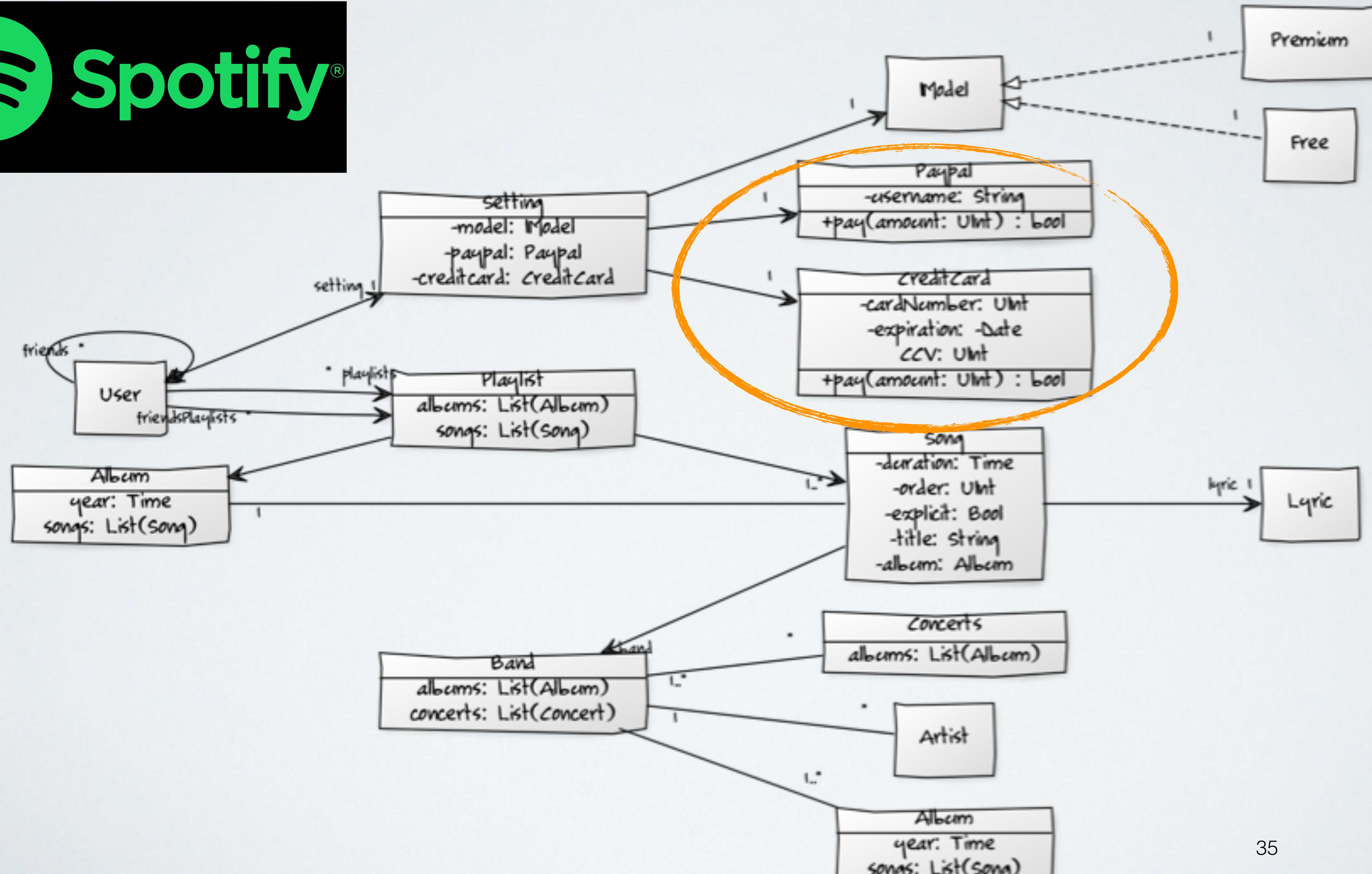
Qui crea Reader Category

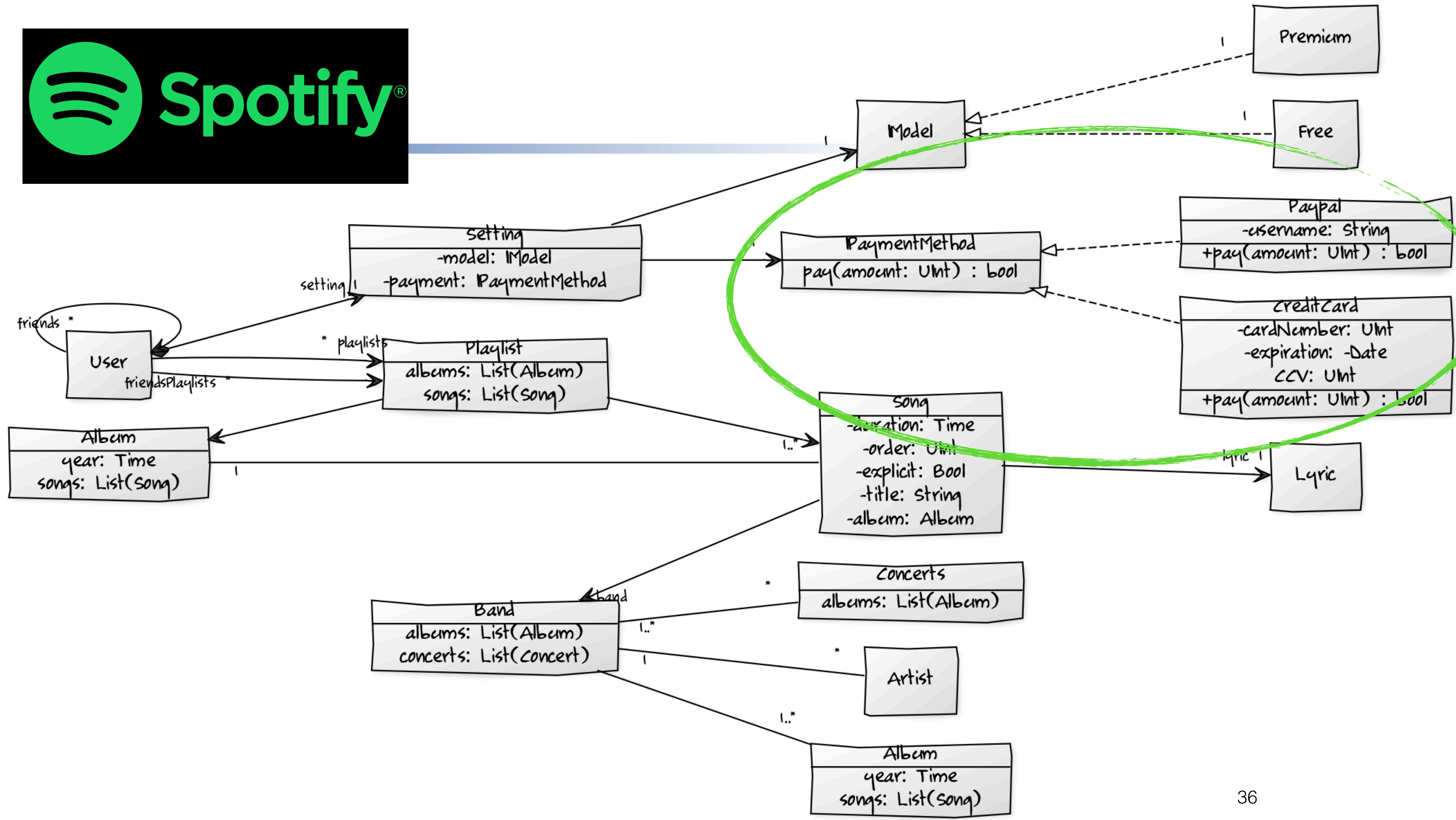
Qui crea Category?

Qui crea Entry?

Cloud i Tag? Qui crea a qui?







Exercici per pensar

- Com aniries construint el Diagram de Classes i el codi de les funcionalitats de UBCultura que es detallen en el Problema del Campus?

Objectiu: Partint del Model de Domini, quin Diagrama de Classes obtens després de dissenyar/implementar les següents funcionalitats de test? Detalla els 4 Diagrames de Classes que vas obtenint...Quins patrons GRASP estàs usant?

1. BuscarClient(DNI) retorna el nom del client
2. Afegir un préstec a un client, donat el DNI del client
3. Llistar tot el material en préstec d'UBCultura en un cert dia
4. Ara es volen també tenir vendes i no només préstecs, com canviaria això el teu diagrama de classes?

Test d'acceptació	Criteri GRASP	Breu explicació de les classes canviades