

# Tema 4 **Model de disseny**

**Maria Salamó Llorente**

**Disseny de Software**

Enginyeria Informàtica

Facultat de Matemàtiques, Universitat de Barcelona

## **PART 3. Finalitzant el disseny (pas previ a la codificació)**

4.4 Diagrama/es de classes

4.5 Visibilitat

- Per atribut
- Per paràmetre
- Local
- Global

## 4.4 Diagrama/es de classes de disseny (DCD)

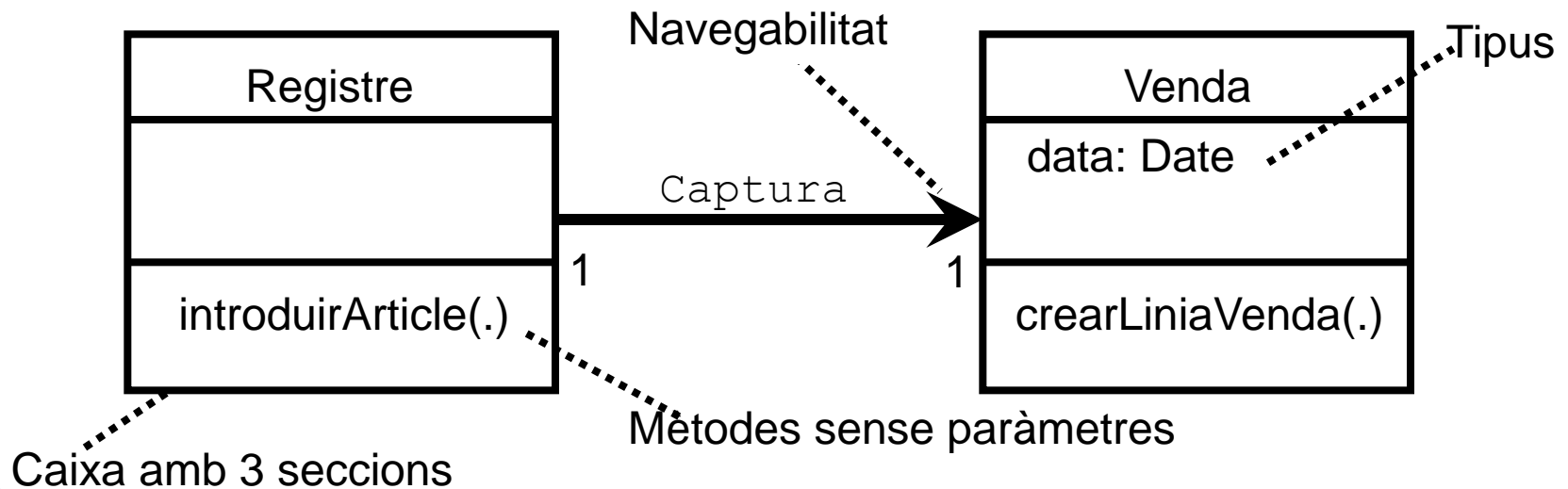
# Definició i components

## Què és?

Un **diagrama de classes de disseny** (DCD) il·lustra les especificacions per classes software i interfícies en una aplicació

Aquest diagrama es desenvolupa en paral·lel amb els diagrames d'interacció

## Components

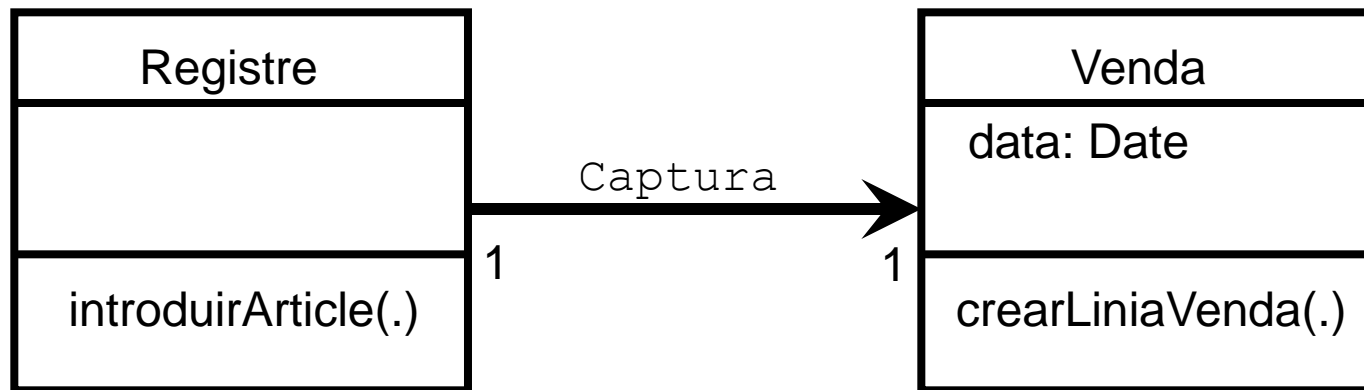


# Diferència entre models

## Model de Domini (classes conceptuals)



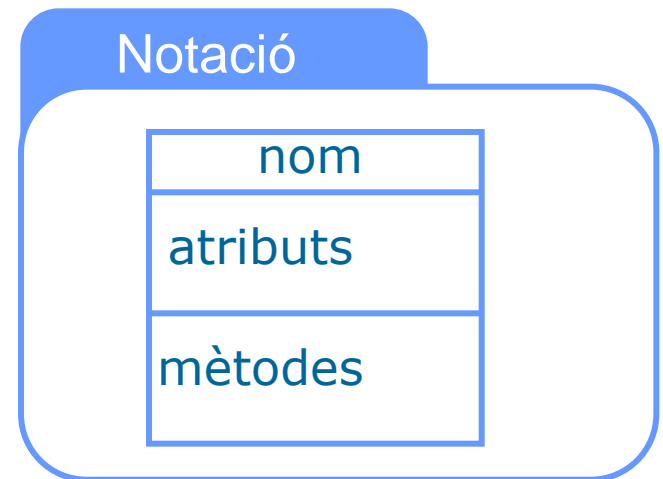
## Model de Disseny (classes software)



# Contingut del DCD

## Comprèn:

- Classes, associacions i atributs
- Interfícies amb les seves operacions i constants
- Mètodes
- Informació sobre els tipus dels atributs
- Navegabilitat
- Visibilitat
- Dependències



# Passos per crear un DCD

- Pas 1. Identificar classes i il·lustrar-les
- Pas 2. Afegir els noms dels mètodes
- Pas 3. Afegir informació de tipus i visibilitat (inclou atributs i paràmetres)
- Pas 4. Afegir associacions i navegabilitat
- Pas 5. Afegir relacions de dependència

# Pas 1. Identificar classes

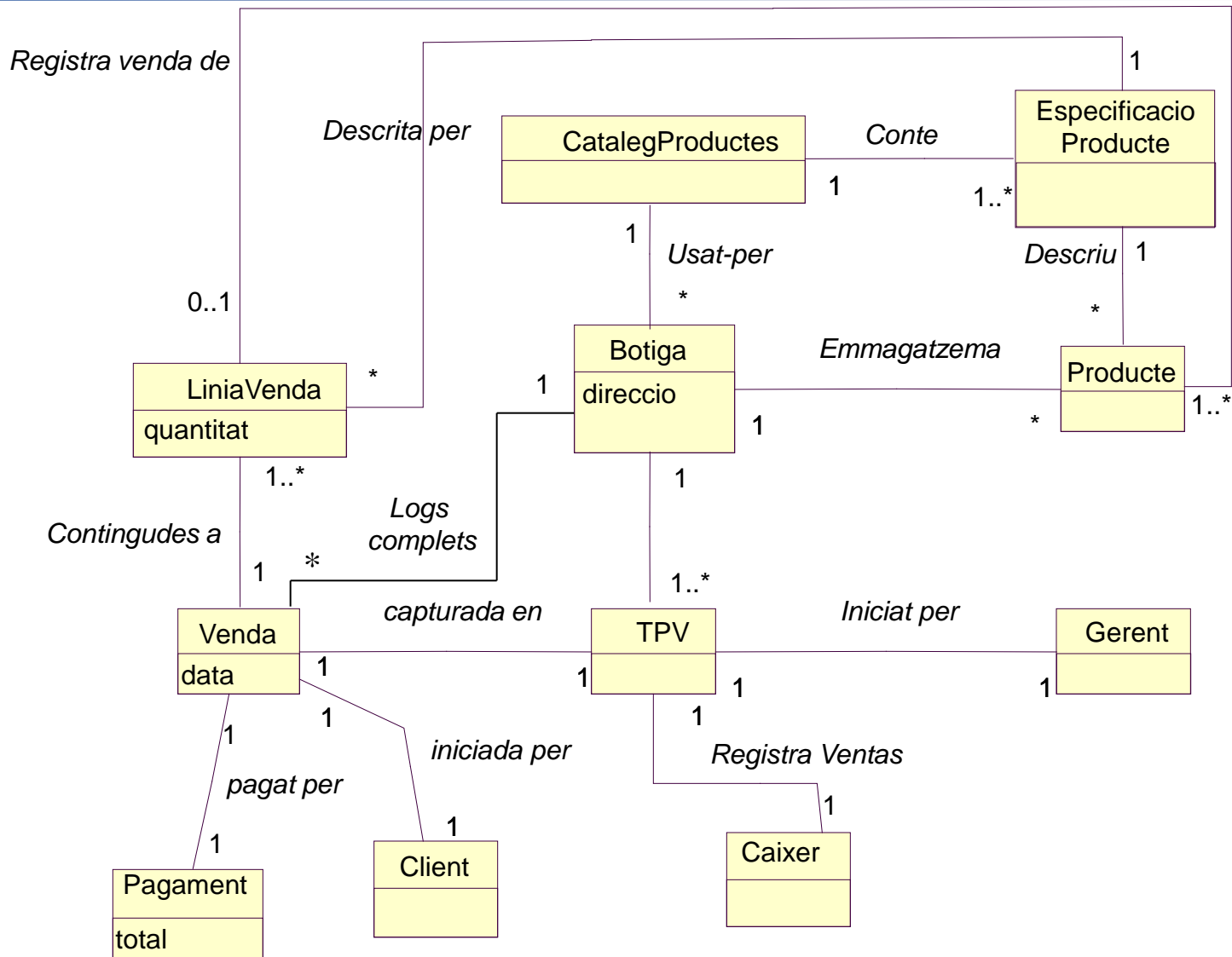
## Pas 1. Identificar classes i il·lustrar-les

### Passos:

1. Trobar classes fent un recorregut als diagrames de seqüència i llistant les classe que apareixen en ells
  2. Dibuixar un diagrama de classes per les classes trobades, incloent-hi els atributs identificats per aquestes classes en el model de domini
- ❶ No totes les classes de domini es convertiran en classes de disseny



# Exemple model domini TPV



# Exemple

## Classes identificades en el Terminal Punt de Venda

LiniaVenda
quantitat

CatalegProductes

Especificacio Producte

Venda
data

Botiga
direccio

Producte

Pagament
total

Client

TPV

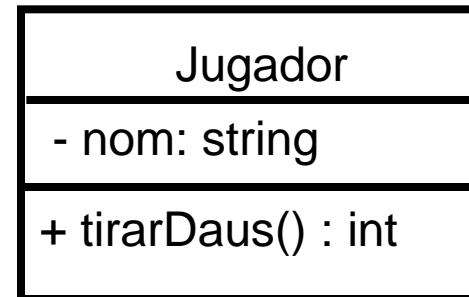
Caixer

Gerent

## Pas 2. Afegir els noms dels mètodes

- **Pas únic:** Trobar els mètodes inicialitzant els diagrames d'interacció.

Exemple: Si en algun diagrama de seqüència, el missatge *tirarDaus()* és enviat a una instància de la classe *Jugador* llavors la classe *Jugador* haurà d'incloure un mètode *tirarDaus()*



## Pas 2. Afegir els noms dels mètodes

### Aspectes rellevants

- El missatge ***create***. Els constructors i destructors normalment no apareixen en el diagrama
- **Mètodes d'accés** a atributs (setter/getter). Habitualment s'omet
- **Missatges a multiojectes**. No s'han d'afegir
- Sintaxis depenent del llenguatge. Mantenir el format UML fins que es realitzi la conversió en el codi

# Exemple

## Alguns dels mètodes identificats en el problema del TPV

LiniaVenda
quantitat
calculaSubTotal()

CatalegProductes

Venda
data : date
afegeixLiniaVenda( ...) realitzaPagament(...) calculaTotal ()

TPV
afegeixVenda ( ...)

## Pas 3. Afegir informació de tipus i visibilitat

- Es pot afegir informació dels tipus dels atributs, paràmetres i valors de retorn dels mètodes
- Si el diagrama es crea per a una eina CASE (com EclipseUML, Rational Rose o ArgoUML) amb generació automàtica de codi, s'han d'indicar de forma exhaustiva tots els detalls
- Si el diagrama es crea per a que altres desenvolupadors puguin llegir-lo, un excessiu nivell de detall pot impactar negativament la facilitat de comprensió del diagrama (tot i que és preferible passar-se a quedar-se curt)

# Exemple

## Alguns dels mètodes identificats en el problema del TPV

### LiniaVenda

- quantitat: int  
+ calculaSubTotal(): double

### CatalegProductes

### Venda

- data : date  
+ afegeixLiniaVenda( ...)  
+ realitzaPagament(...)  
+ calculaTotal () : double

### TPV

+ afegeixVenda ( ...)

### Notació:

- atributs privats
- mètodes de servei públics
- mètodes de suport privats

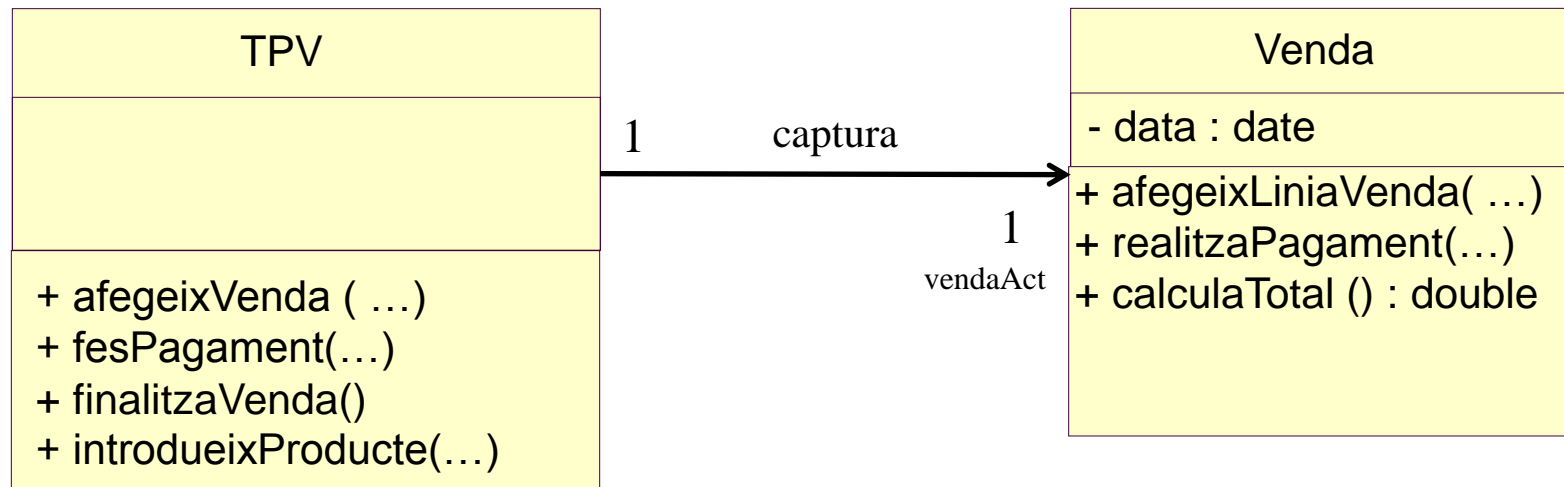
## Pas 4. Afegir associacions i navegabilitat

- **Navegabilitat:** és una propietat d'un rol d'una associació que ens indica que és possible navegar unidireccionalment d'objectes de l'origen a objectes del destí segons la direcció indicada per la fletxa
- En el DCD cada rol es pot decorar amb una fletxa de navegabilitat. Per a la majoria de les associacions és molt important indicar la navegabilitat
- La navegabilitat indica també visibilitat. Habitualment, visibilitat per atribut



# Exemple

## Exemple de navegabilitat i visibilitat

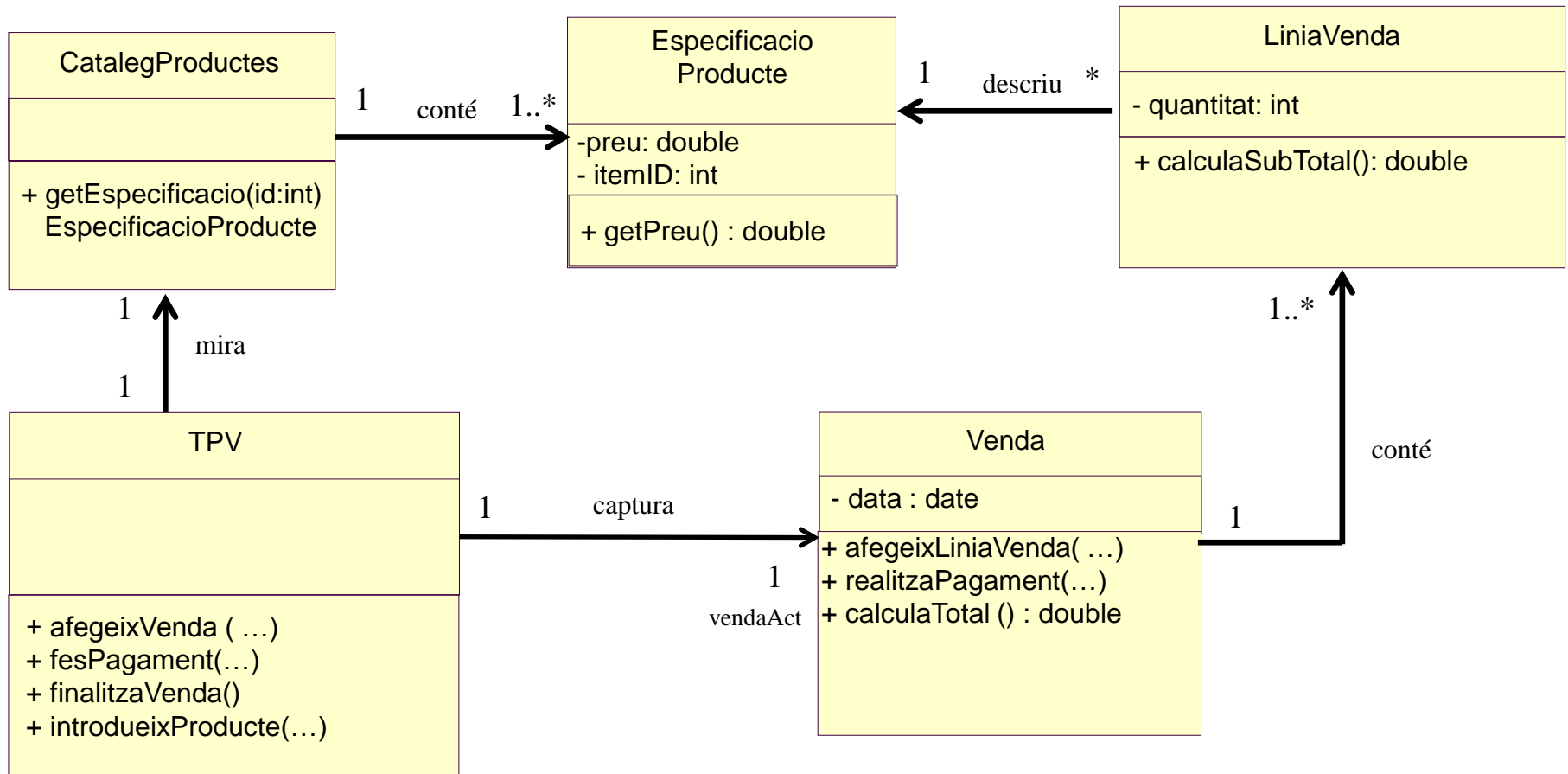


## Pas 4. Afegir associacions i navegabilitat

- Només s'han d'incloure en el DCD les associacions requerides per satisfer la visibilitat i les necessitats de memòria indicades pels diagrames d'interacció
- Situacions que suggereixen la necessitat d'incloure una associació amb navegabilitat d'A a B:
  - A envia un missatge a B
  - A crea una instància de B
  - A necessita mantenir una connexió amb B

# Exemple

## Associacions i navegabilitat en una part del Terminal Punt de Venda

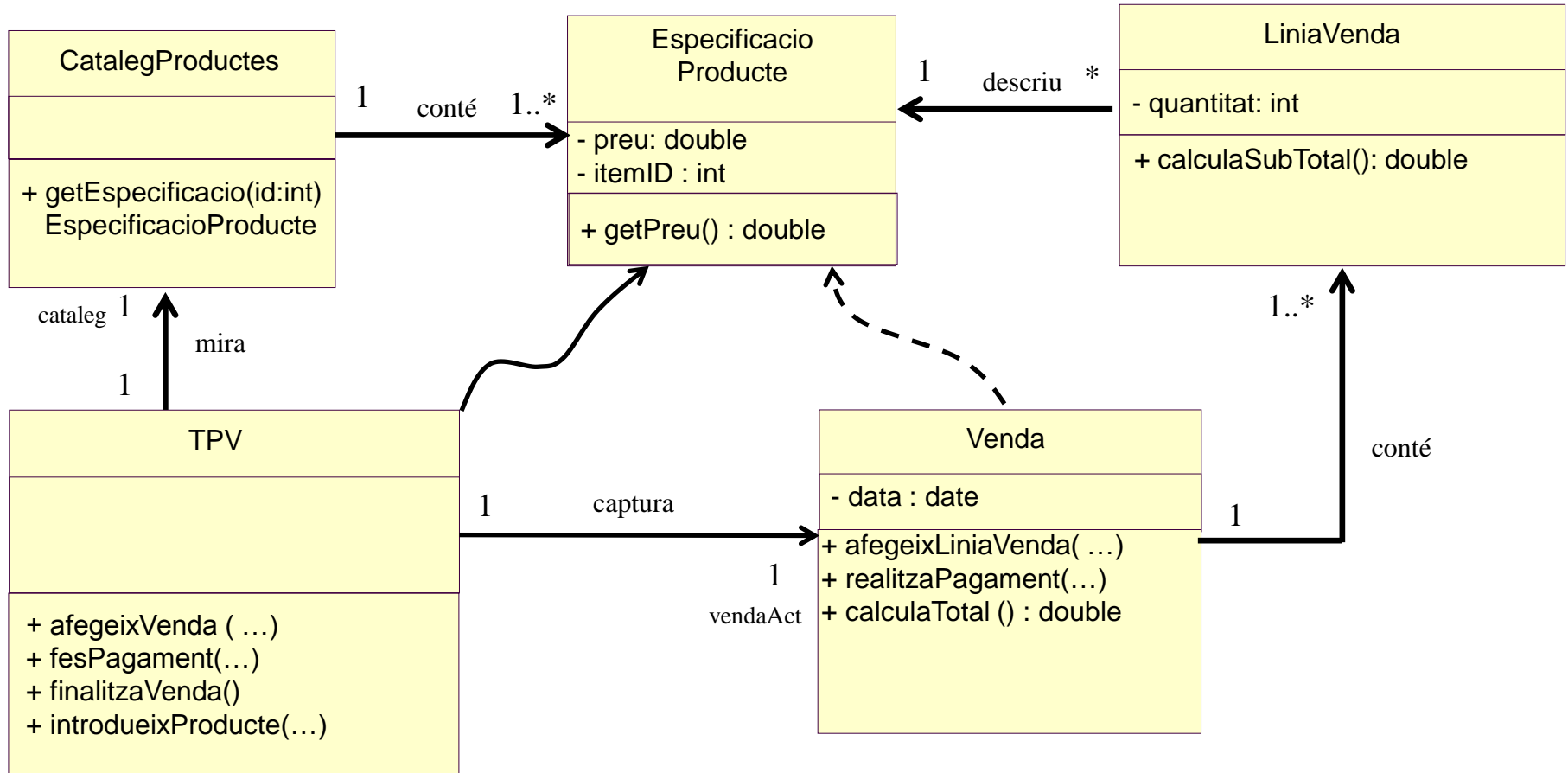


## Pas 5. Afegir relacions de dependència

- UML inclou una relació de dependència genèrica que indica que un element de qualsevol tipus (classes, casos d'ús, etc.) té coneixement d'un altre
- En els DCD pot ser molt útil representar visibilitat entre classes que no siguin per atribut (si és per atribut s'usa una associació), ja sigui paràmetre global o local.

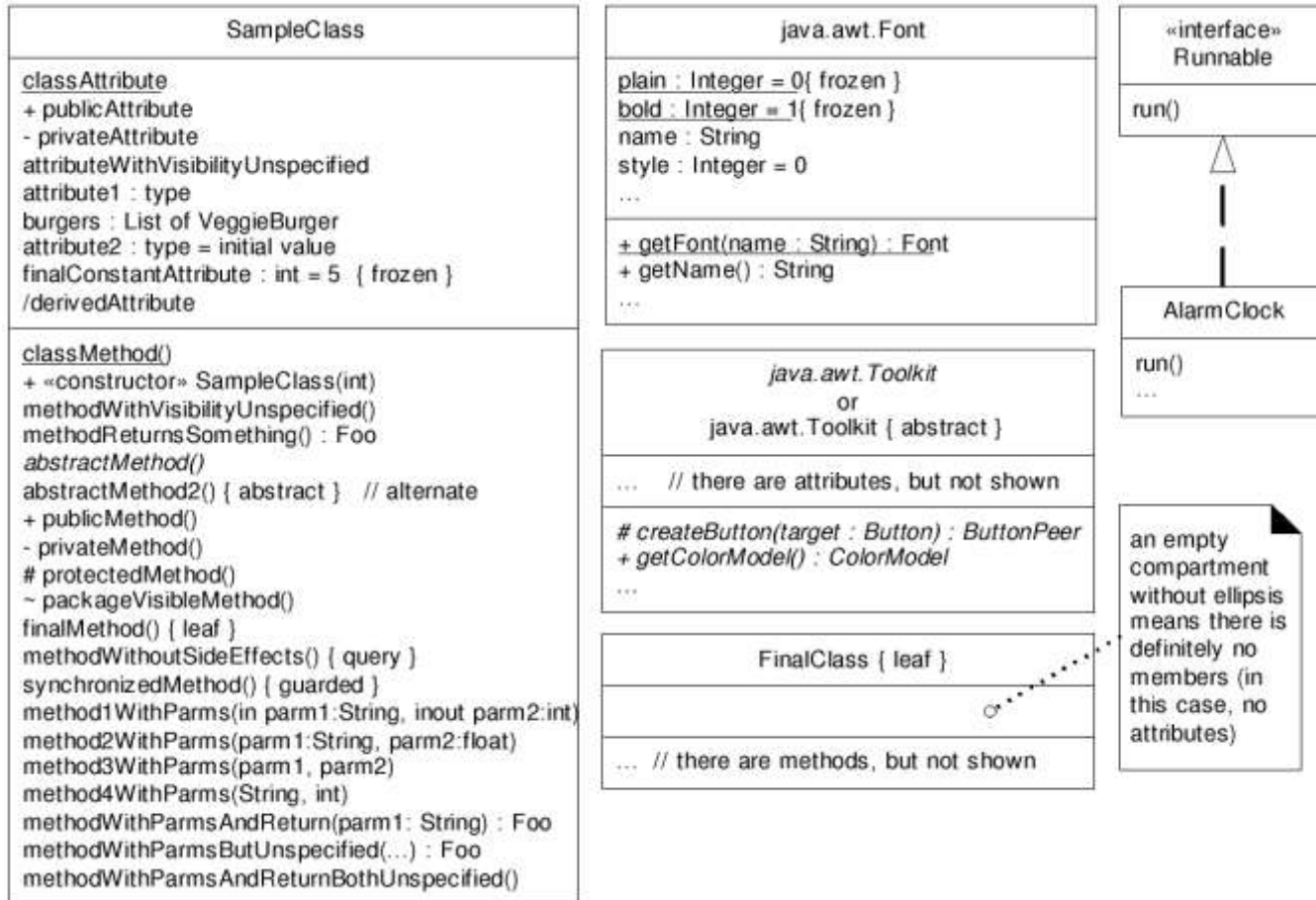
# Exemple

## Exemple de dependències en el Terminal Punt de Venda



# Visibilitat dels membres en UML

Si no s'especifica la visibilitat (private, protected, public) d'un atribut o mètode en UML **no** se li assigna visibilitat per defecte, sinó que queda "no especificada"



# Exemple

## Visibilitat dels membres en el Terminal Punt de Venda

CatalegProductes
+ getEspecificacio(id:int) : EspecificacioProducte

Especificacio Producte
- preu: double - itemID : int
+ getPreu() : double

LiniaVenda
- quantitat: int
+ calculaSubTotal(): double

TPV
+ afegeixVenda ( ...) + fesPagament(...) + finalitzaVenda() + introdueixProducte(...)

Venda
- data : date
+ afegeixLiniaVenda( ...) + realitzaPagament(...) + calculaTotal () : double

# Notació UML per als cossos dels mètodes

## NOTACIO UML:

- La implementació d'un mètode es pot mostrar en una nota UML entre claus { }
- Es pot mostrar tant en pseudocodi com en un llenguatge concret

TPV

+ afegeixVenda ( ...)  
+ fesPagament(...)  
+ finalitzaVenda()  
+ introdueixProducte(id, quant)

```
{  
    EspecificacioProducte spec = cataleg.getEspecificacio(id);  
    vendaAct.afegeixLiniaVenda(spec, quant);  
}
```

```
{  
    public void introdueixProducte(int id, int quant)  
    {  
        EspecificacioProducte spec = cataleg.getEspecificacio(id);  
        vendaAct.afegeixLiniaVenda(spec, quant);  
    }  
}
```



## 4.5 Visibilitat

# Definició i Tipus

**Què és la Visibilitat?** és la capacitat d'un objecte de veure o tenir referències d'un altre objecte

Com s'aconsegueix la visibilitat des d'un objecte A a un objecte B?

1. **Visibilitat per atribut:** B és un atribut d'A
2. **Visibilitat per paràmetre:** B és un paràmetre d'un mètode d'A
3. **Visibilitat local:** B és un objecte local (no paràmetre) d'A
4. **Visibilitat global:** B és d'alguna manera visible globalment

# 1. Visibilitat per atribut

## Visibilitat per atribut

A té visibilitat per un atribut de B si B és un atribut d'A.  
Visibilitat permanent ja que persisteix mentre A i B existeixen

TPV

+ afegeixVenda ( ...)  
+ fesPagament(...)  
+ finalitzaVenda()  
+ introdueixProducte(id, quant)

```
class TPV {  
    ...  
    private CatalegProductes cataleg;  
    ....  
}
```

```
{  
    public void introdueixProducte(int id, int quant)  
    {  
        EspecificacioProducte spec = cataleg.getEspecificacio(id);  
        vendaAct.afegixLiniaVenda(spec, quant);  
    }  
}
```

## 2. Visibilitat per paràmetre

### Visibilitat per paràmetre

A té visibilitat per paràmetre de B quan B es passa com paràmetre a un mètode d'A.  
És una visibilitat relativament temporal ja que persisteix tan sols en l'àmbit del mètode.  
Després de visibilitat per atribut és la segona més comú

Venda

- data : date

+ afegeixLiniaVenda( spec: EspecificacioProducte, quant:int)  
+ realitzaPagament(...)  
+ calculaTotal () : double

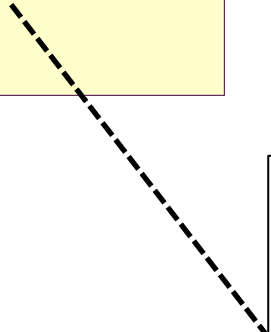
```
{  
    public void afegeixLiniaVenda(EspecificacioProducte spec, int quant)  
    {  
        ....  
        s1 = new LiniaVenda( spec, int);  
    }  
}
```

# Transformar visibilitats

Visibilitat per paràmetre a visibilitat per atribut

És comú transformar la visibilitat per paràmetre en visibilitat per atribut

LiniaVenda
- quantitat: int
+ calculaSubTotal(): double



```
{ // inicialitzant un mètode (e.g., constructor java)
public LiniaVenda (EspecificacioProducte spec, int quant)
{
    ....
    especificacioProd = spec ; // el parametre agafa visibilitat per atribut
    ....
}
}
```

# 3. Visibilitat local

- És una visibilitat temporal, persisteix únicament durant l'àmbit del mètode
- A té visibilitat local d'A quan B és declarat com un objecte local dins d'un mètode d'A
- Formes d'aconseguir visibilitat local:
  - Crear una nova instància local i assignar-la a una variable local
  - Assignar a una variable local l'objecte que retorna la invocació d'un mètode
- És comú transformar la visibilitat local en visibilitat per atribut

# Visibilitat local (cont.)

## Visibilitat local

Es pot tenir visibilitat local de forma implícita sense assignar l'objecte que retorna la invocació d'un mètode a una variable local.

### TPV

- + afegeixVenda ( ...)
- + fesPagament(...)
- + finalitzaVenda()
- + introduceixProducte(id, quant)

```
{  
  public void introduceixProducte(int id, int quant)  
  {  
    // visibilitat local usant l'assignació de l'objecte que es retorna  
    EspecificacioProducte spec = catalog.getEspecificacio(id);  
    vendaAct.afegexLiniaVenda(spec, quant);  
  }  
}
```

## 4. Visibilitat global

- És una visibilitat permanent, persisteix mentre A i B existeixen
- A té visibilitat global de B quan B és global a A.
- És la menys recomanable de les visibilitats (crea dissenys amb un alt acoblament)
- La manera més recomanable de tenir visibilitat global és mitjançant l'ús del patró **Singleton**

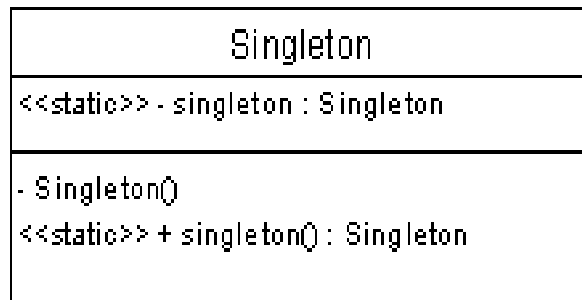


# Visibilitat global (cont.)

Visibilitat Global: B és d'alguna manera visible globalment

Visibilitat permanent – persisteix mentre ho fan A i B

El patró de disseny **Singleton**



```

<<static>> + singleton(): Singleton
  if
    singleton = null
  then
    singleton = new Singleton;
  end if
  return singleton;
  
```

Singleton.getsingleton().metode();