

# MODELAT ORIENTAT A OBJECTES AMB UML

---

## Tema 4

**Enginyeria del Programari**  
ETS Enginyeria Informàtica  
DSIC – UPV

*Curs 2021-2022*

# Objectius

- Mostrar la necessitat de construir models per a resoldre problemes complexos i de grans dimensions
- Comprendre què és el modelatge conceptual i distingir-ho clarament del disseny
- Aprendre un subconjunt de UML, com a notació de modelatge OO
- Modelar l'estructura d'un sistema. Modelar el comportament d'un sistema.

# Continguts


1. Motivació.
2. Modelat OO
  - Visió d'un sistema programari OO
3. Notació UML
  - Diagrama de Classes **(Part 1)**

---

  - Diagrama de Casos d'Ús **(Part 2)**
  - Diagrames de Seqüència
  - Altres diagrames

# Bibliografia bàsica

 Booch, G., Rumbaugh, J., Jacobson, I., UML. El Lenguaje Unificado de Modelado. UML 2.0 2ª Edición. Addison-Wesley, 2006

 Stevens, P., Pooley, R. Utilización de UML en Ingeniería del Software con Objetos y Componentes. 2ª Edición. Addison-Wesley Iberoamericana 2007

 Weitzenfeld, A., Ingeniería del Software OO con UML. Java e Internet. Thomson, 2005

 , ...

 [www.uml.org](http://www.uml.org)

# Motivació

Què és un model?

“Un model és una simplificació de la realitat”

Per què modelem?

Construïm models per a comprendre millor el sistema que estem desenvolupant

- Ens ajuden a **visualitzar** com és o volem que siga un sistema.
- Ens permeten **especificar** l'estructura o el comportament d'un sistema
- Ens proporcionen plantilles que ens guien en la **construcció** d'un sistema
- **Documenten** les decisions que hem adoptat

# Motivació

- Modelat Orientat a Objectes
  - Apareixen els llenguatges de programació OO
  - Es requereix un nou enfocament d'anàlisi i disseny

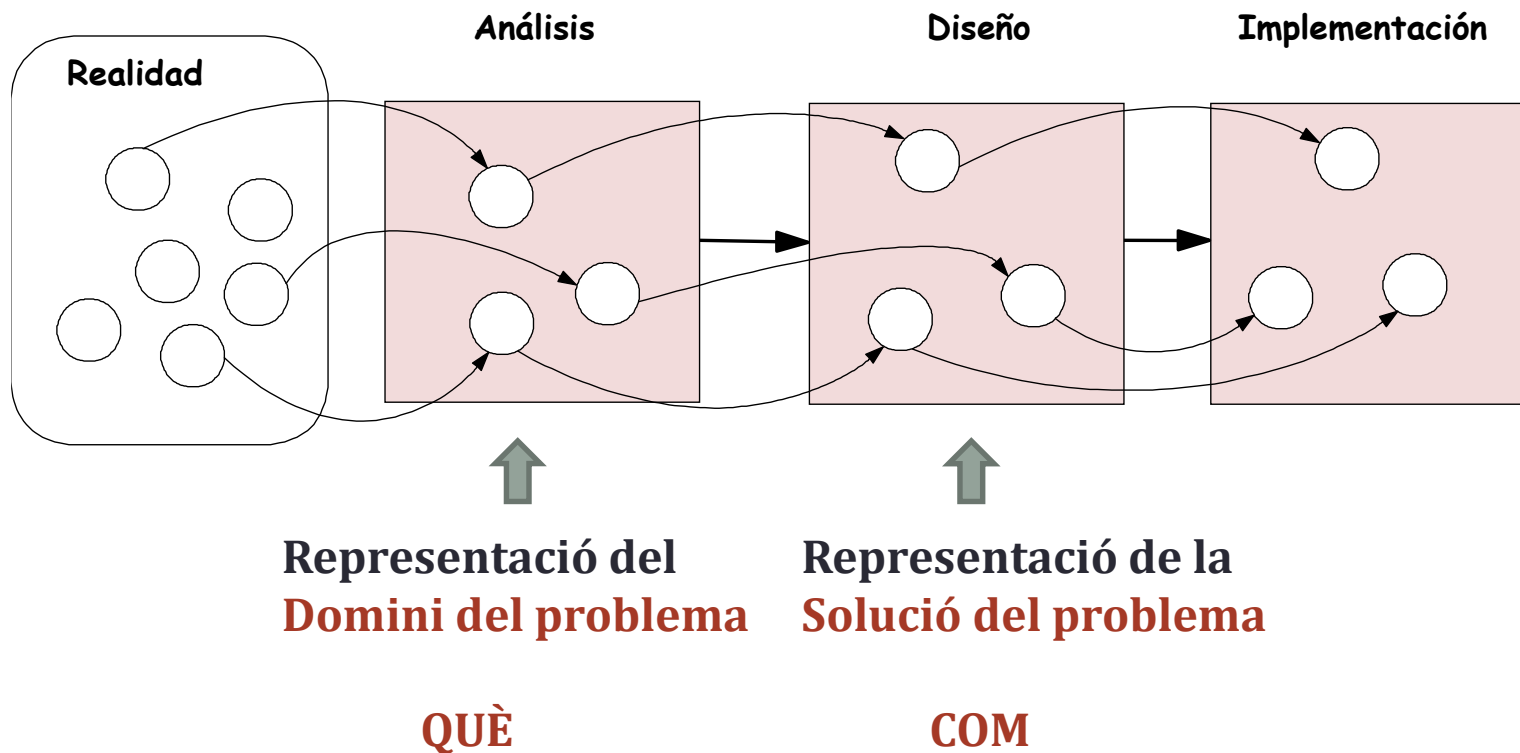
*“Un procés que examina els requisits des de la perspectiva de les classes i objectes descoberts en el vocabulari del domini del problema”*  
(Booch, 1994)



1. Pròxim als **mecanismes cognitius humans**
2. Desenvolupament **incremental** sota una **noció** comuna d'objecte.

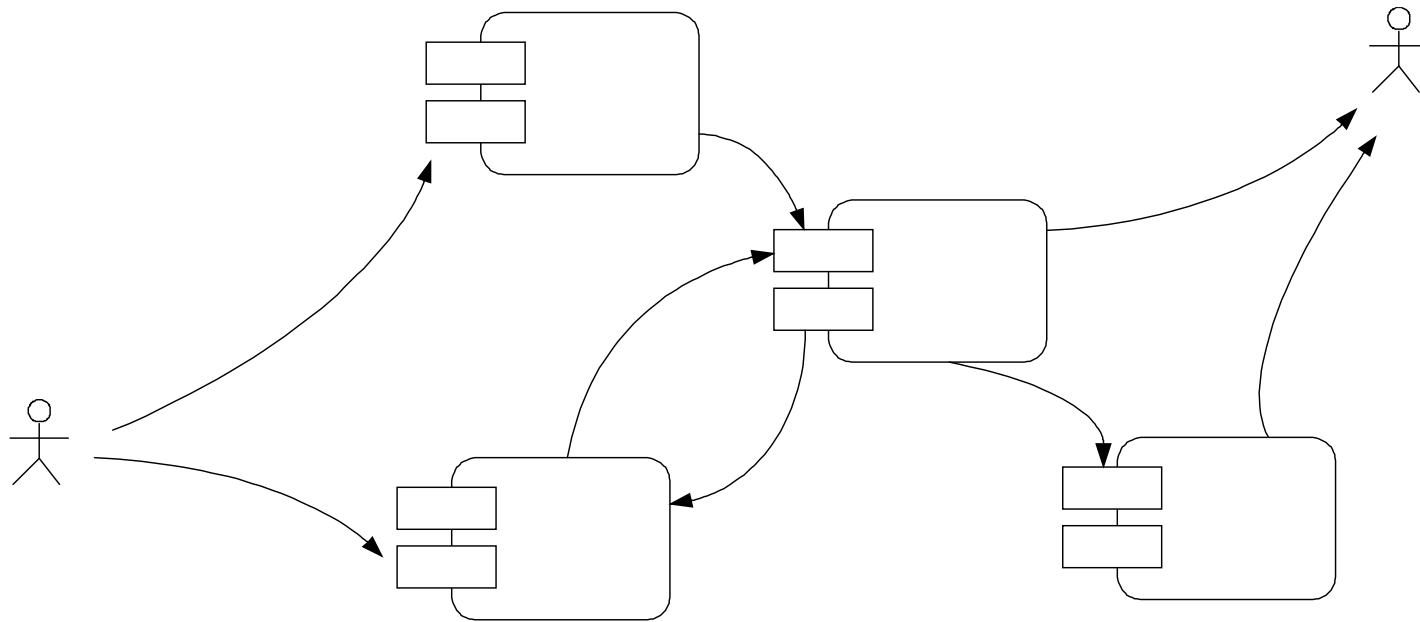
# Motivació

- En l'enfocament OO, la descomposició del sistema es basa en *els objectes* o *classes d'objecte* que es descobreixen en el domini del problema



# Visió d'un Sistema Programari OO

**Visió estàtica + Visió dinàmica**





# Sistema Programari OO - Visió estàtica

- **Objecte:**

- Entitat que existeix en el món real
- Té identitat, una estructura i un estat

*L'avió amb matrícula 1234*

*L'avió amb matrícula 6754*

- **Classe:**

- Descriu un conjunt d'objectes amb les mateixes propietats i un comportament comú.
- Relacions entre classes

Avió

# Sistema Programari OO - Visió dinàmica

- Els objectes es comuniquen mitjançant la **invocació de mètodes** d'altres objectes.
- Es descriuen aspectes d'un sistema que canvien amb el temps.
  - Interaccions entre objectes
  - Possibles estats d'un objecte
  - Transicions entre estats
  - Quins esdeveniments es produeixen
  - Quines operacions s'executen

# UML



## Unified Modeling Language

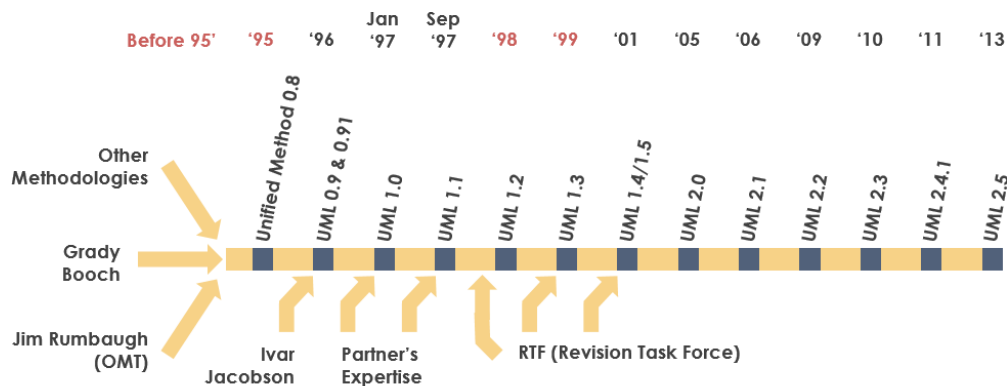
---



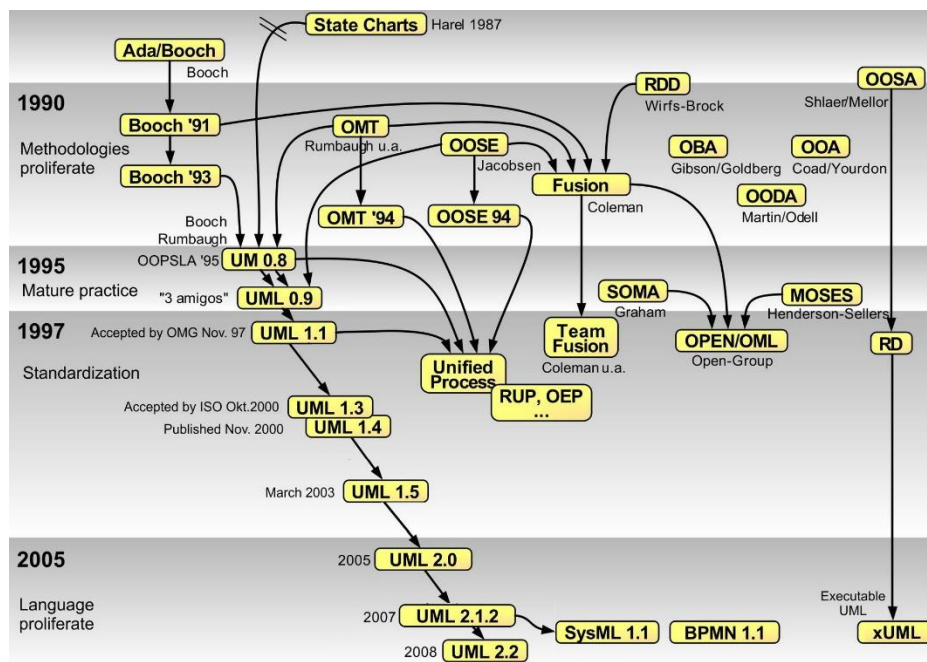
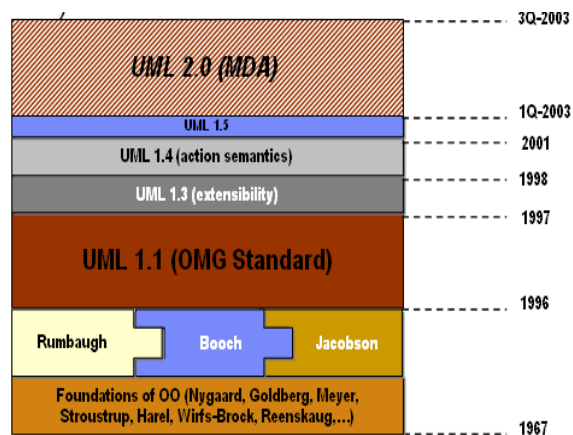
The Unified Modeling Language™ (UML™) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, creating a "blueprint" for construction.

- Visualitzar
- Especificar
- Construir
- Documentar

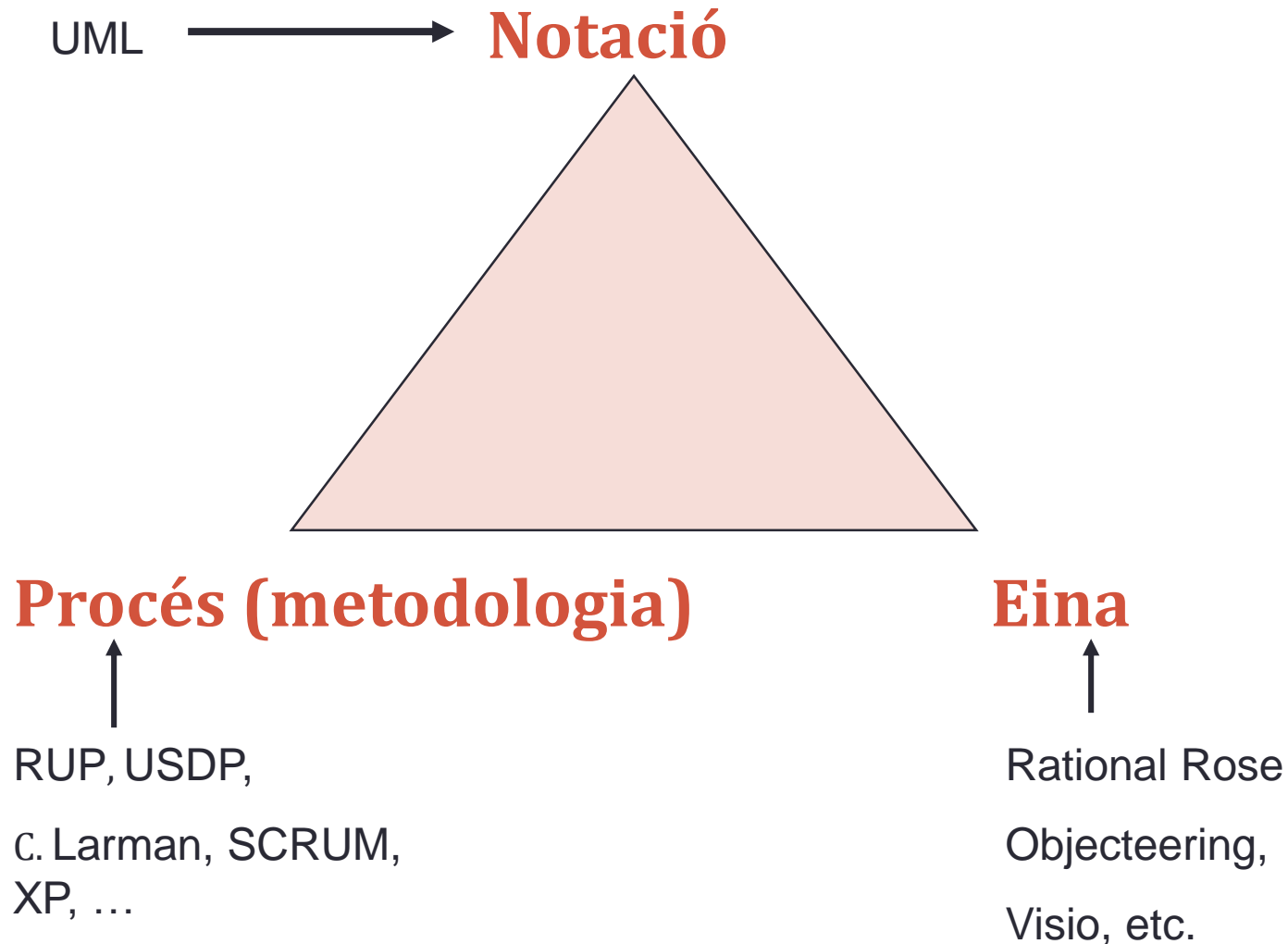
# UML - Evolució



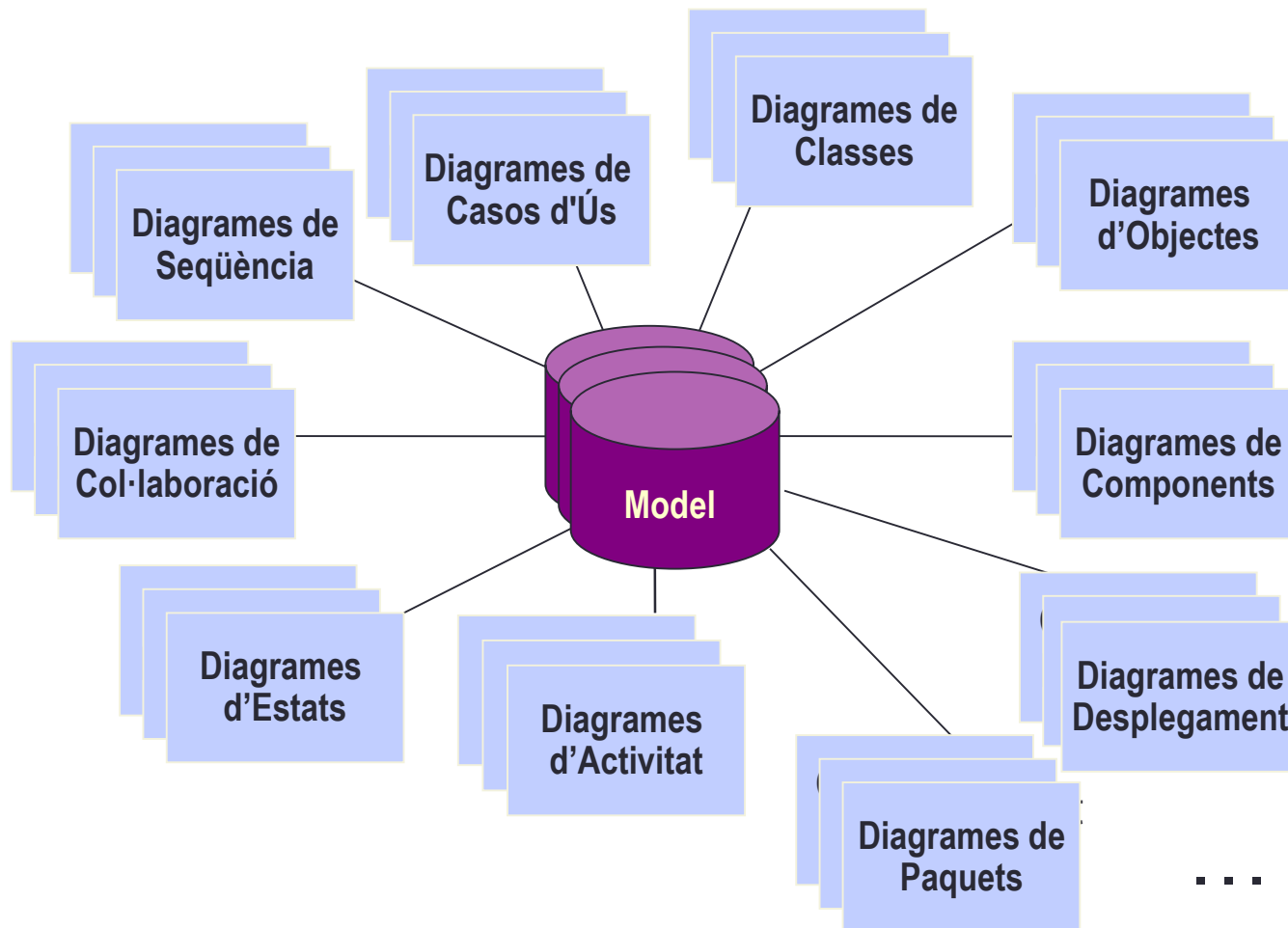
Before '95' - Fragmentation ▶ '95' - Unification ▶ '98' - Standardization ▶ '99' - Industrialization



# UML - Triangle de l'èxit



# UML



# UML - Tipus de diagrames

Diagrama de Classe (incloent Diagrama d'Objectes). **Part 1**

Diagrama de Casos d'Ús. **Part 2**

## Diagrames de Comportament

- Diagrama d'Estats

- Diagrama d'Activitat

## Diagrames d'Interacció

- Diagrama de Seqüència

- Diagrama de Col·laboració

## Diagrames d'implementació

- Diagrama de Components

- Diagrama de Desplegament



# Part 1:

## Diagrama de Classes

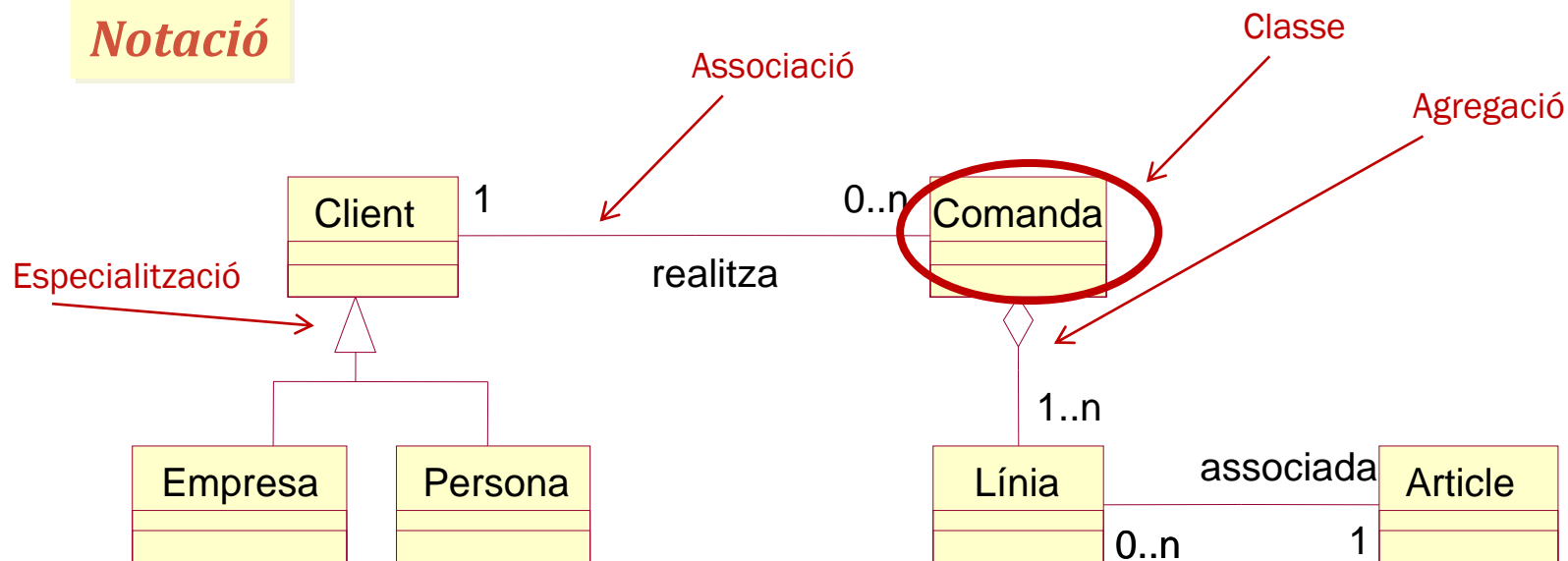
- Classes – Objectes
- Relacions entre classes
  - Associació
  - Agregació
  - Composició
  - Especialització/  
Generalització  
(Herència)



# Diagrama de Classes

- Mostra l'estructura estàtica del sistema, mostrant les classes i les relacions entre elles
- És l'eina principal de la major part dels mètodes OO

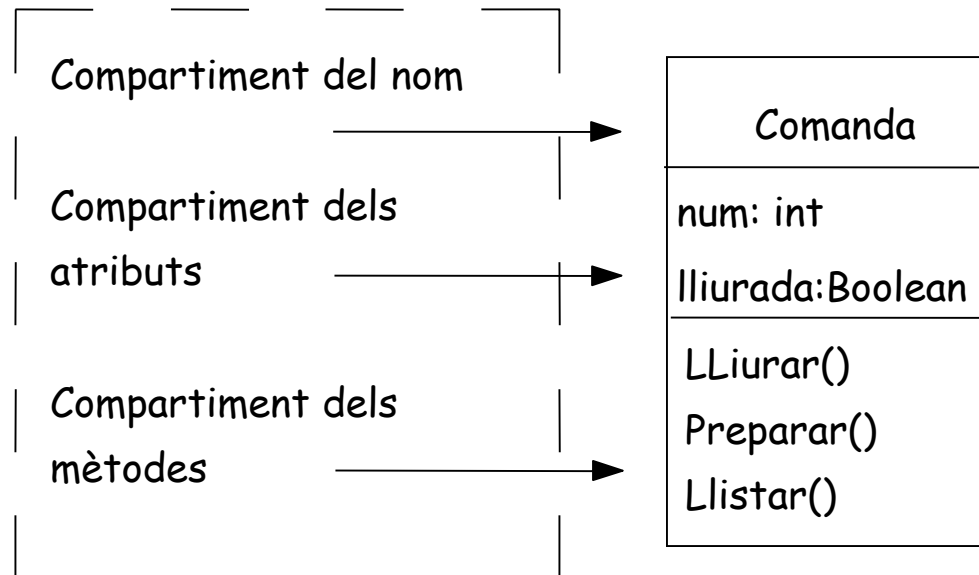
## Notació



# Classe

- És la descripció d'un grup d'objectes amb estructura, comportament i relacions similars

## Notació



# Classe

- Els atributs/operacions poden ser:

- **(-)** Privats
- **(#)** Protegits
- **(+)** Públics

## Regles de visibilitat

+ Atribut públic : int  
# Atribut protegit : int  
- Atribut privat : int

+ "Operació pública"  
# "Operació protegida"  
- "Operació privada"

- Els atributs es poden representar mostrant únicament el nom.
- Els atributs no inclouen referència a altres objectes, aquestes referències es representen mitjançant enllaços.
- Un atribut derivat es representa com a /Atribut : Tipus
- Un mètode és la implementació d'una operació

# Classes / Objectes

Objectes

Classes

**Un Arbre Binari:**  
**Arbre Binari**

**Houston: Ciutat**

Nom Ciutat: Houston  
poblacio: 3.000.000

(Persona)

Pepe

**Arbre Binari**

**Ciutat**

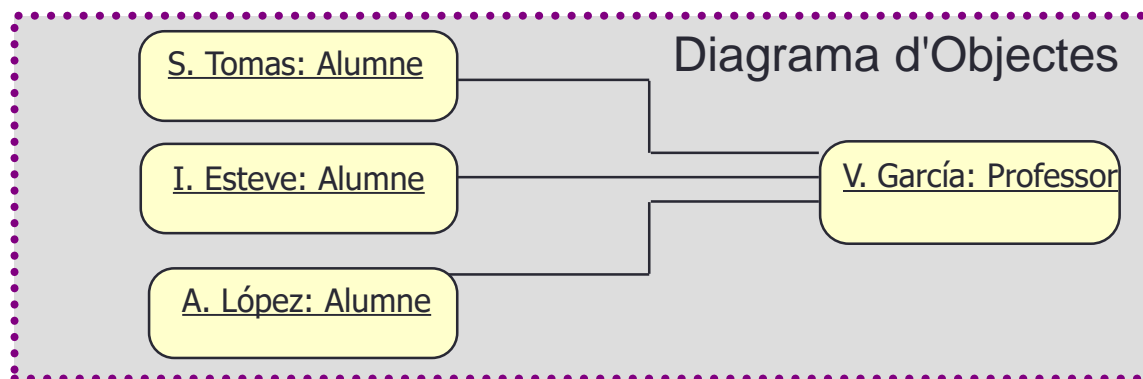
- Nom Ciutat: String  
- Poblacio: Real

**Persona**

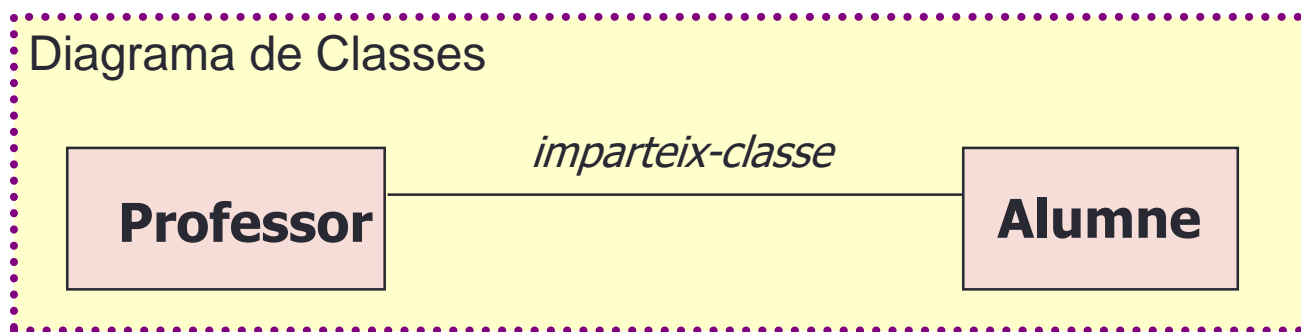
Nom: String

# Associacions

- Un enllaç és una connexió física o conceptual entre objectes
- Una associació és una relació estructural que especifica que els objectes d'un element estan connectats amb els



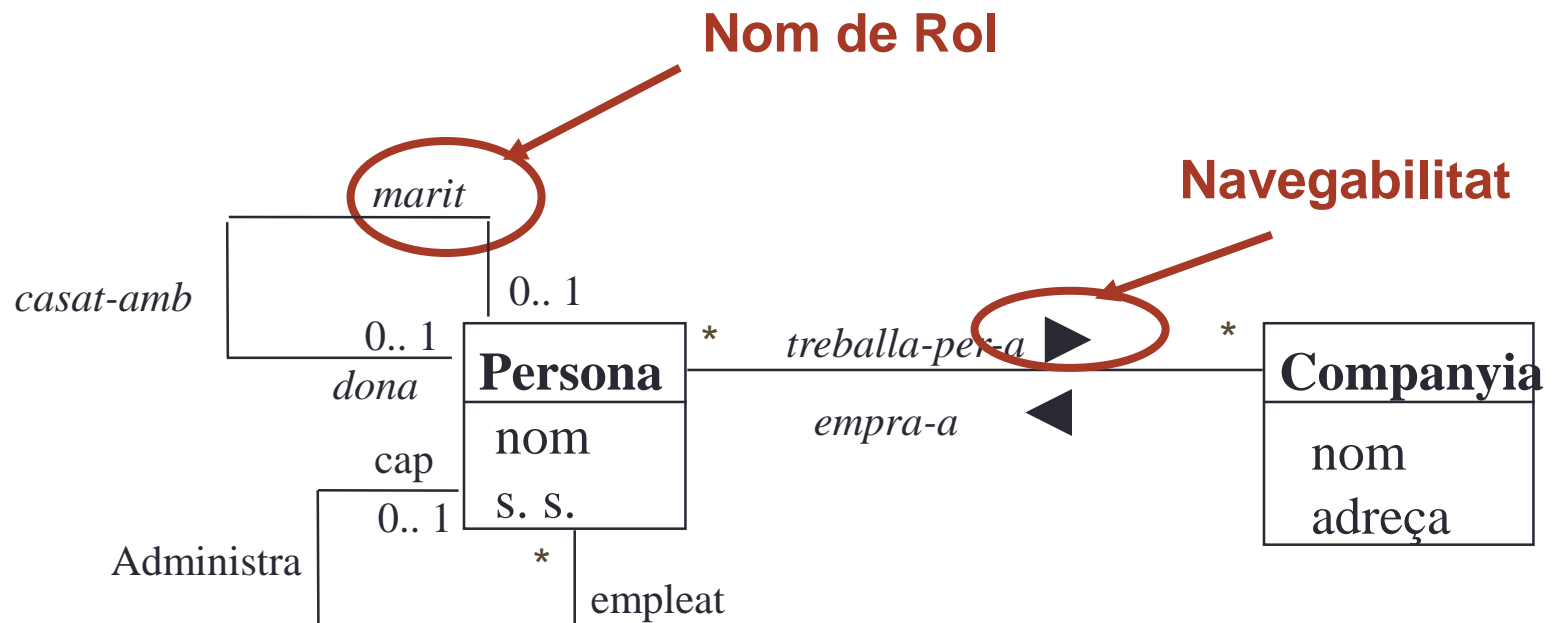
**Enllaç**



**Associació**

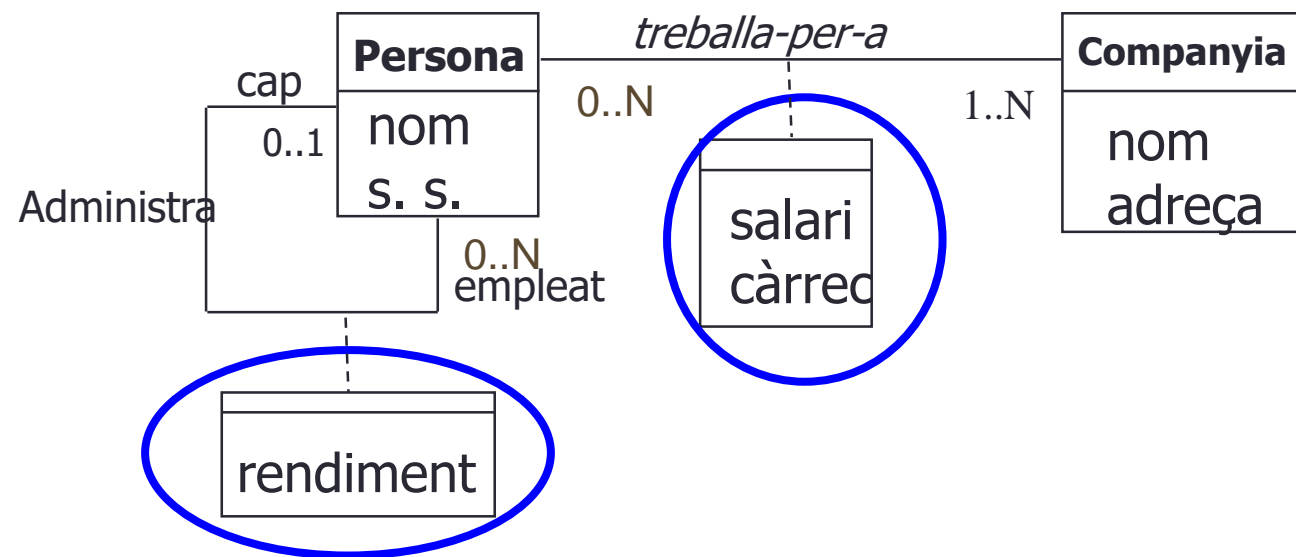
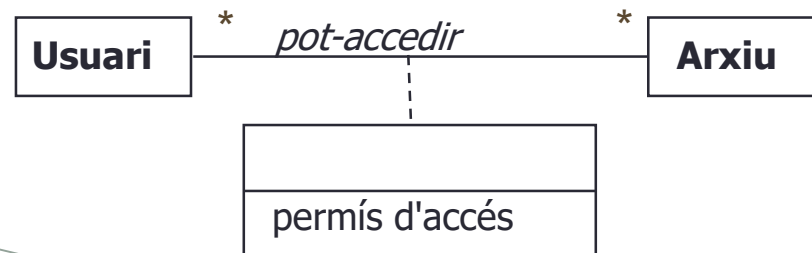
# Associacions

- Tota associació és bidireccional, és a dir, es pot navegar en els dos sentits, des d'objectes d'una classe a objectes de l'altra.
- Té un nom, o noms de rol
- Multiplicitat: 1, 0..1, 0.. N (\*), 1.. N, M.. N, M,



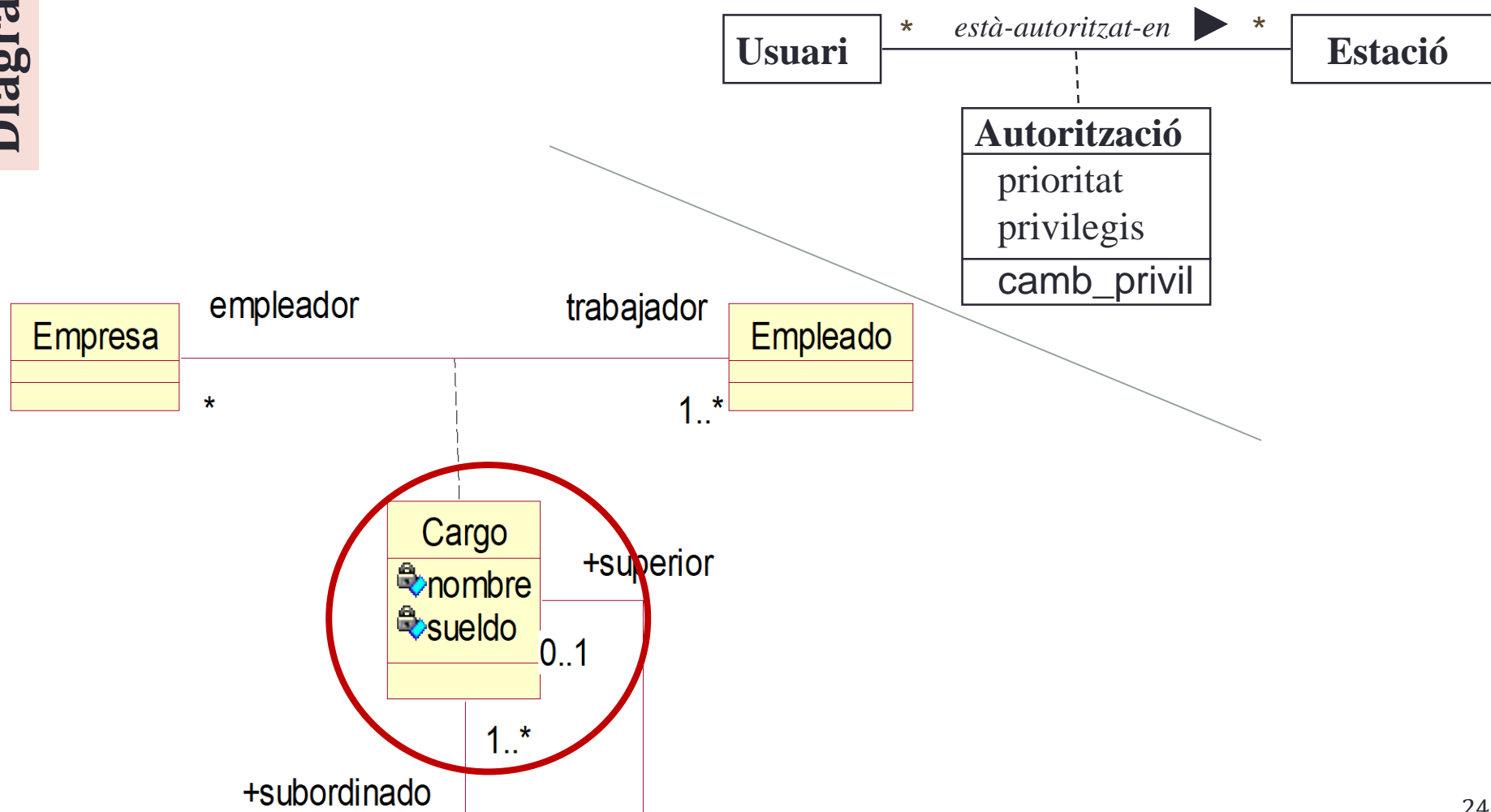
# Associacions - Atributs d'Enllaç

- En una associació entre classes, la pròpia relació pot tenir propietats, denominats atributs d'enllaç.



# Associacions - Classe Associació

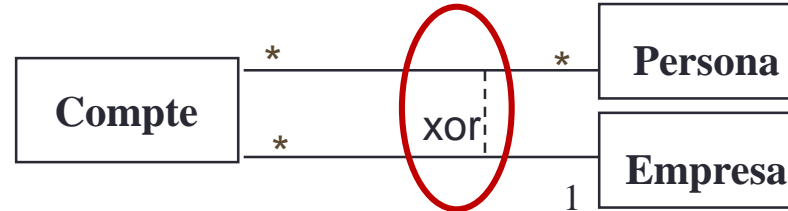
- L'atribut d'enllaç pot ser una classe.





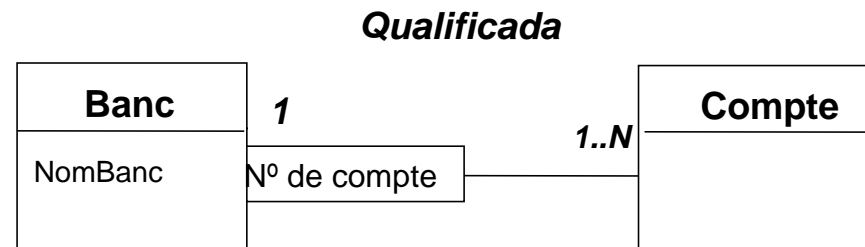
# Associacions - Associació excloent

## ■ XOR



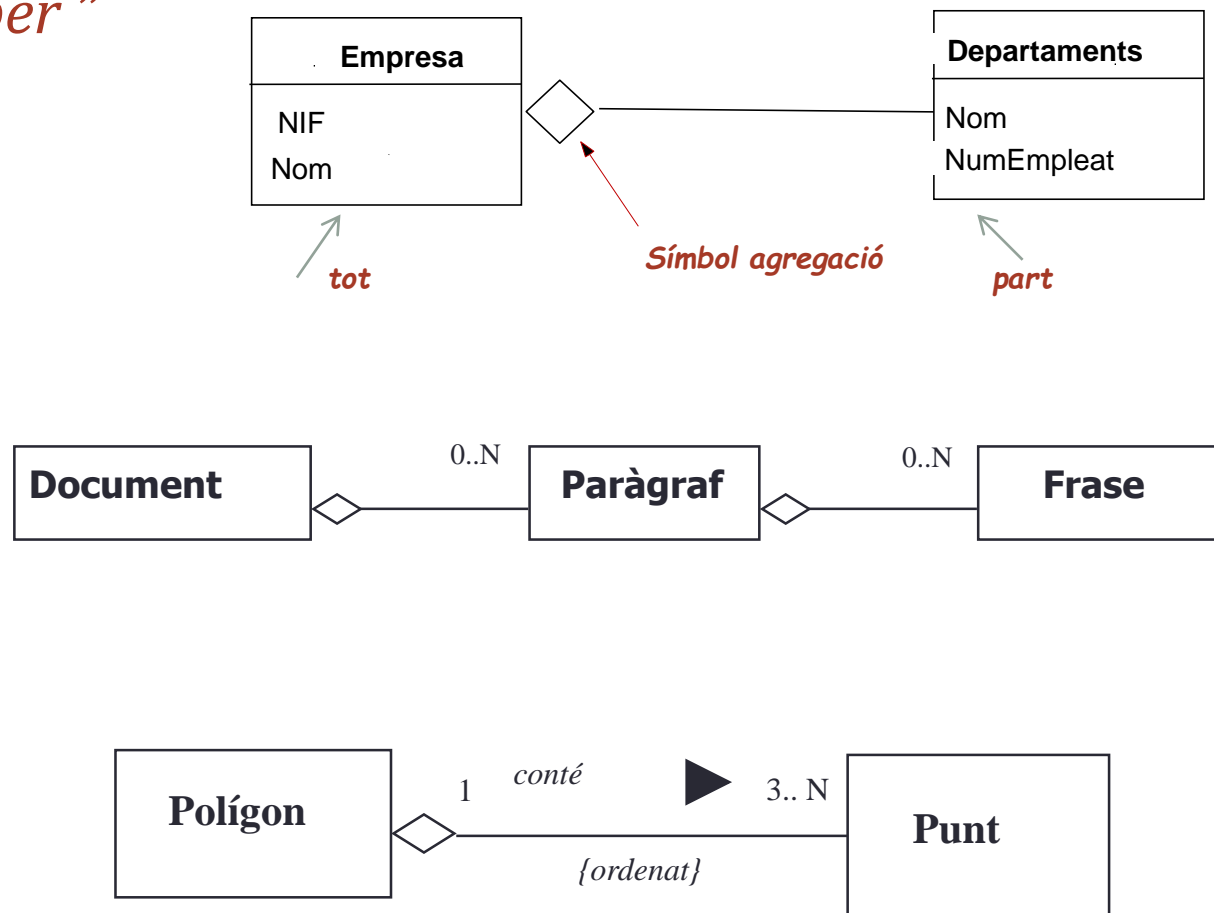
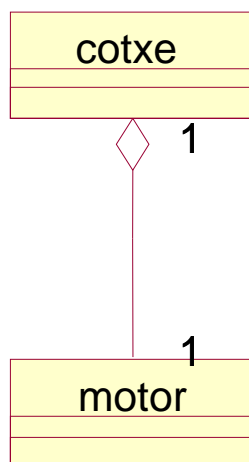
# Associacions - Associació qualificades

- Qualificadors ens serveixen per a refinar més el model, indicant l'índex per a recórrer la relació (*Com identificar un objecte o conjunt d'objectes en l'altre extrem?*)



# Agregació

- És una associació amb unes propietats semàntiques addicionals.
- “*està format per*”



# Agregació

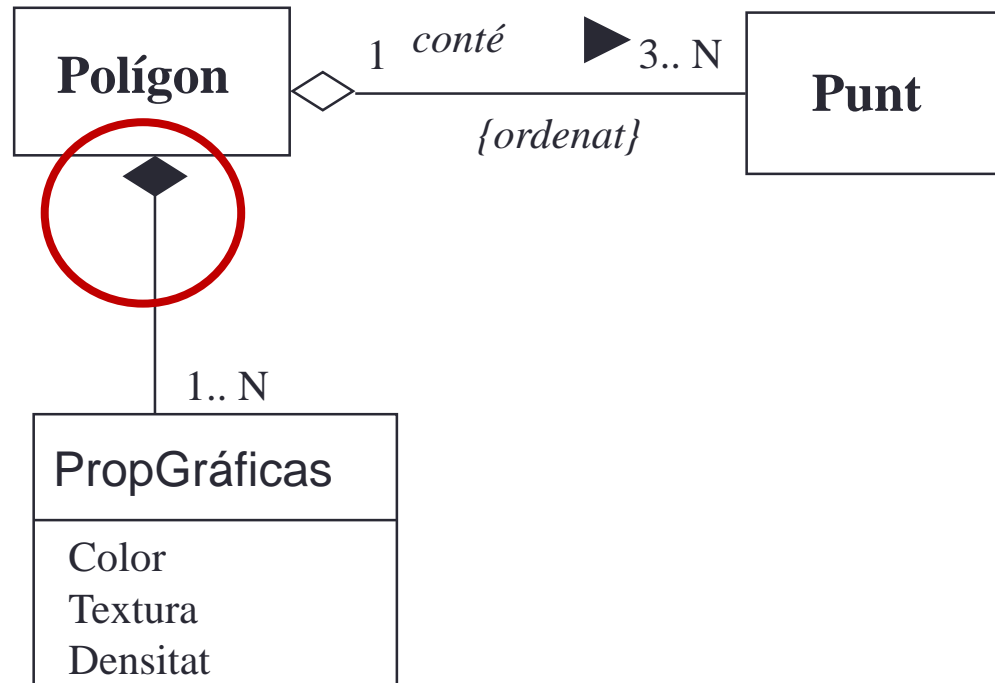
- **Tipus d'agregació:**

***Inclusiva o física:*** cada component pot pertànyer com a màxim a un compost. La destrucció del compost implica la destrucció de les parts.

***Referencial o de catàleg:*** els components són reutilitzables al llarg de diferents compostos. No estan relacionats els temps de vida.

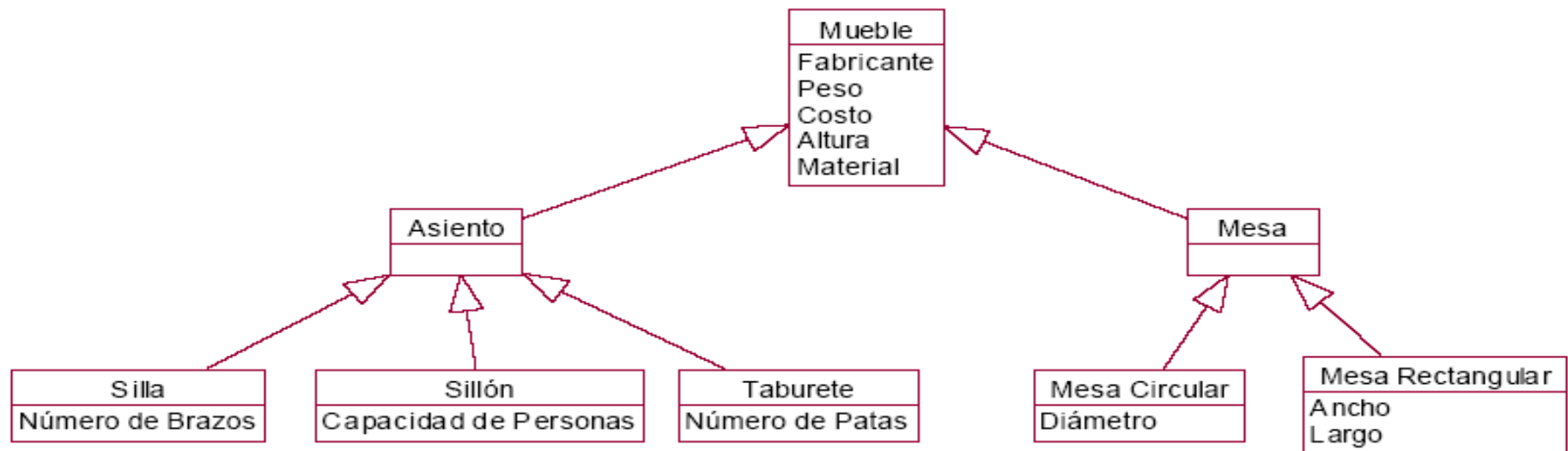
# Composició

- És una agregació inclusiva o física
- “*està compost per*”

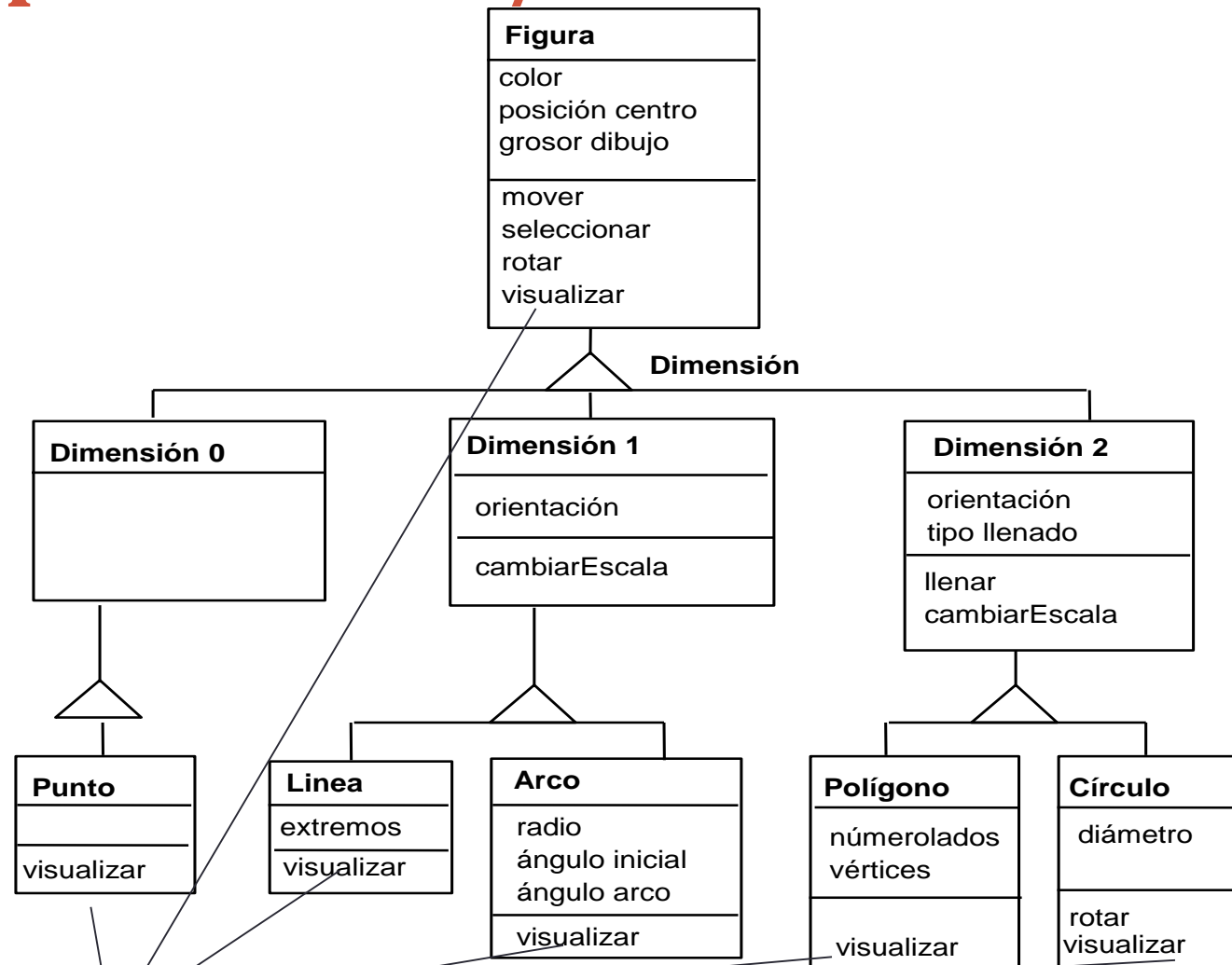


# Especialització / Generalització

- Permeten definir jerarquies de classes
- Generalització: Donat un conjunt de classes, si tenen en comú atributs i mètodes, es pot crear una classe més general (superclasse) a partir de les inicials (subclasses)
- Especialització: És la relació contrària.
- “és un”



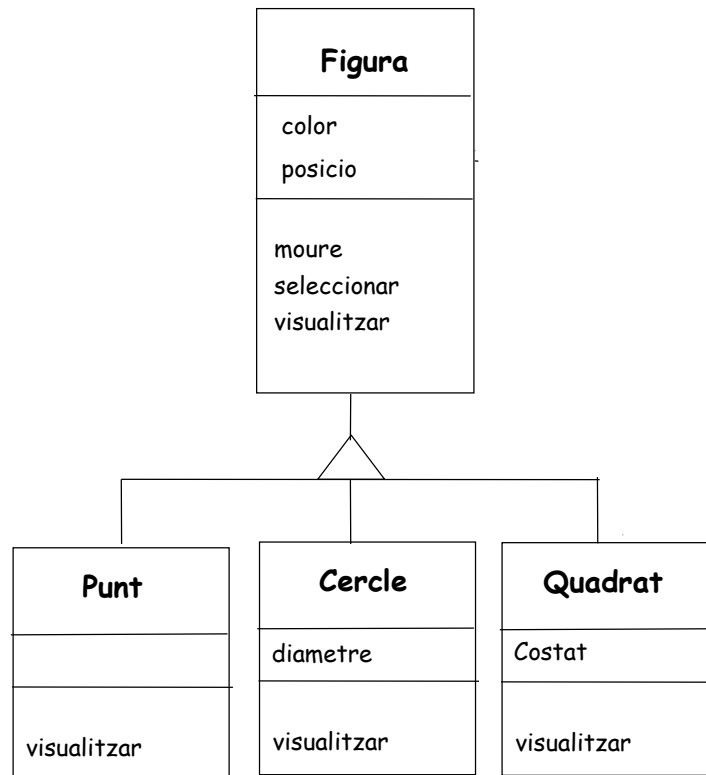
# Especialització / Generalització



Quan en una jerarquia d'especialització es repeteix una característica d'una classe (atribut o mètode) estem **redefinint** la característica heretada

# Especialització / Generalització

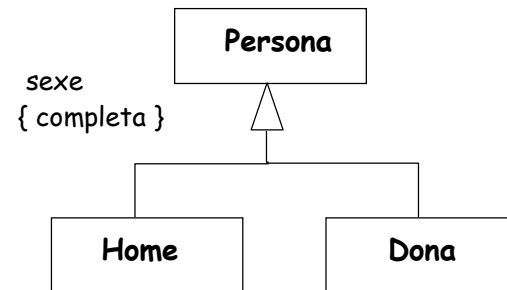
- La relació d'especialització es gesta en la fase de modelat d'un sistema, mentre que la relació d'herència es veu com un mecanisme de reutilització de codi en la fase d'implementació o disseny.



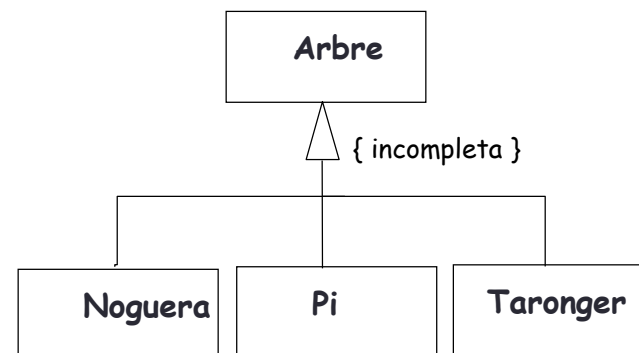
En Implementació, l'herència permet **reutilitzar** els atributs i operacions de la classe **Figura**

# Especialització / Generalització

- Dos tipus de **restriccions**:
  - **Completa**: Tots els fills de la generalització s'han especificat en el model



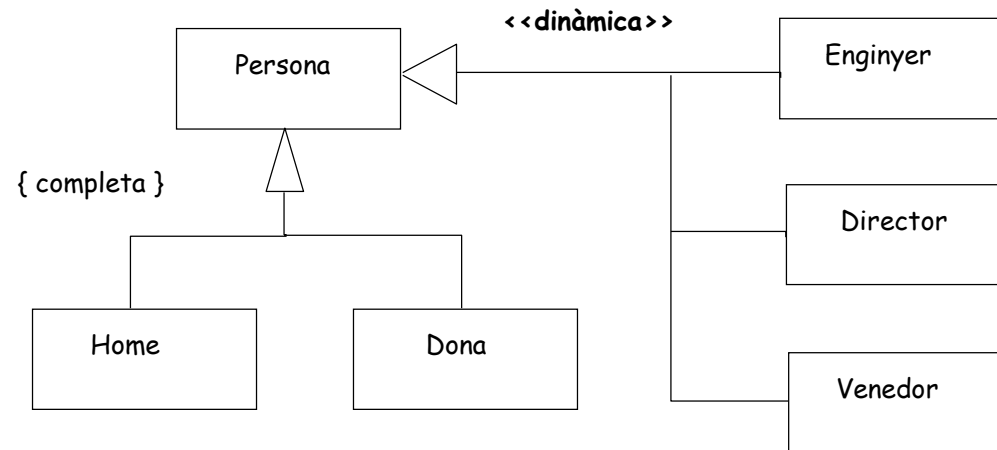
- **Incompleta**: No s'han especificat tots els fills i es permeten fills addicionals



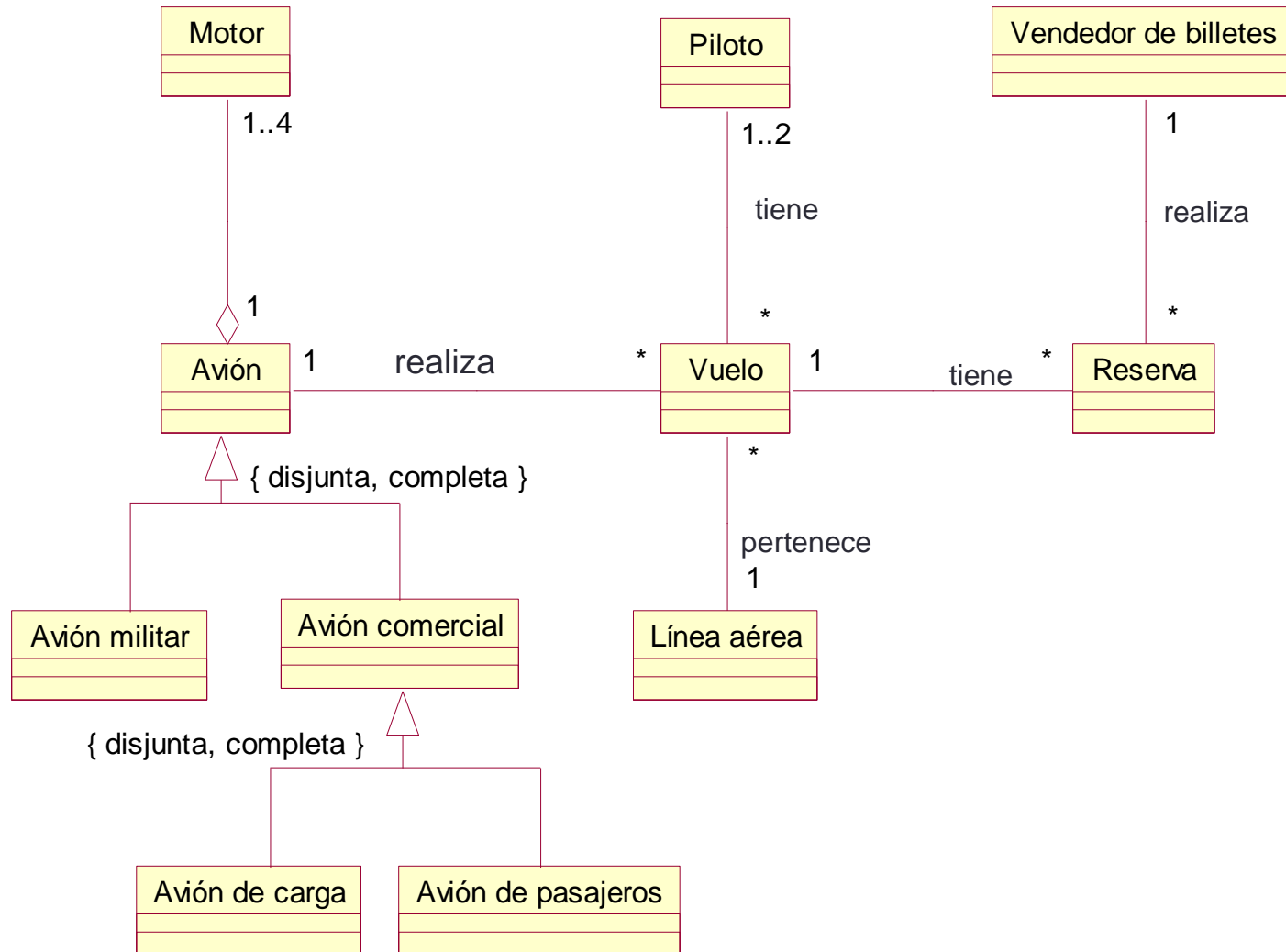


# Especialització / Generalització

- Es parla d'especialització **dinàmica** quan un objecte pot canviar de classe dins d'una jerarquia de subclasses.



# Exemples



# Exemples

