



Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA



Diagrames de classes UML

Programació Orientada a Objectes (POO)
ETSETB Telecom BCN

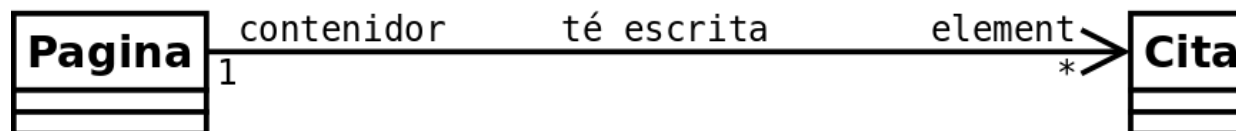
Jordi Perelló Muntan, tardor 2015-16

Unified Modeling Language (UML)

- Llenguatge utilitzat per a representar gràficament dissenys d'aplicacions basats en la metodologia de la programació orientada a objectes
- **UML** com a tal és molt extens, el qual **proposa molts tipus de diagrames diferents**. En aquest tema analitzarem els diagrames de classes UML
- Els **diagrames de classes UML** modelen totes les classes involucrades en l'aplicació que pretenem desenvolupar, així com les relacions entre elles:
 - **Classes**: Tipus d'objectes que tindrem en la nostra aplicació (nom, atributs i mètodes)
 - **Relacions**: Com es relacionen els objectes de les diferents classes?
 - Associacions
 - Relacions de dependència
 - Relacions de generalització/especialització

Associacions (1/3)

- Tipus més freqüent de relació entre classes. Existeix una **associació entre dues classes** quan **els objectes d'una d'aquestes classes poden invocar mètodes sobre els objectes de l'altra**
 - Es representen amb una **línia contínua que uneix les dues classes** involucrades en l'associació
 - Inclouen una sèrie de **descriptors** (nom, rols, navegabilitats, cardinalitats) els quals ens caracteritzen completament l'associació
- **Exemple:** Associació entre les classes Pàgina i Cita



En els diagrames de classes UML les classes es representen amb caixetes, les quals contenen el nom de la classe a dalt de tot

Associacions (2/3)

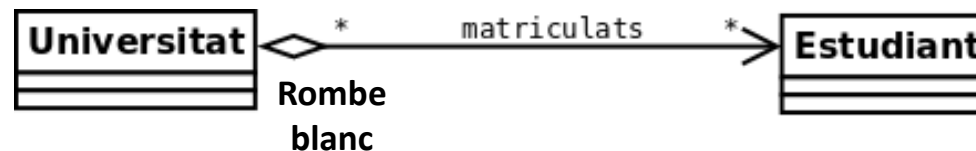
- Descriptors de les associacions:
 - **Nom:** nom de l'associació a nivell conceptual -> “té escrita”
 - **Rols:** funcions de les classes implicades en l'associació -> “contenedor”
“element”
 - **Navegabilitat:** indica el sentit del flux d'invocacions de mètodes. És a dir, quins són els objectes que invoquen mètodes, i quins els que reben les invocacions -> Els objectes Pagina invoquen mètodes sobre els objectes Cita (navegabilitat unidireccional de Pagina -> Cita)
 - **Cardinalitat:** Donat 1 objecte d'una classe, a quants objectes de l'altre classe pot estar enllaçat? Possibles formes d'especificar-ho:
 - Un nombre exacte: 1
 - Un nombre indeterminat d'objectes ≥ 0 : *
 - Un rang (punts suspensius): 1..3, 2..4, 1..*
 - Una sèrie de valors vàlids (separats per comes): 1,2

Associacions (3/3)

- **Important**: La cardinalitat sempre ens l'hem de mirar des del punt de vista de cada una de les classes, i s'indica a l'altre extrem de l'associació:
 - Un objecte Pagina pot estar enllaçat amb un nombre indeterminat ≥ 0 d'objectes Cita (una Pagina pot contenir des de 0 a indeterminades Cites)
 - Un objecte Cita ha d'estar enllaçat sempre amb un únic objecte Pagina (no podem tenir una mateixa Cita en múltiples Paginees)

Tipus especials d'associacions (1/2)

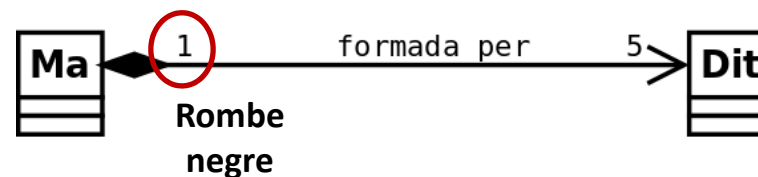
- **Agregació/Composició**: mostren explícitament que existeix una relació tot-part entre els objectes de les dues classes involucrades
 - **Agregació (agregació dèbil)**: Relació tot-part on les parts tenen “vida pròpia”. Si es destrueix el tot, les parts romanen vives (poden formar part de diversos agregats)



Navegabilitat SEMPRES
unidireccional del tot
cap a les parts*

- **Composició (agregació forta)**: Relació tot-part on les parts no tenen sentit sense el tot. Si es destrueix el tot, les parts també

En les relacions de
composició les parts
només poden
pertànyer a 1 sol tot.
Sempre cardinalitat
1 a la banda del tot

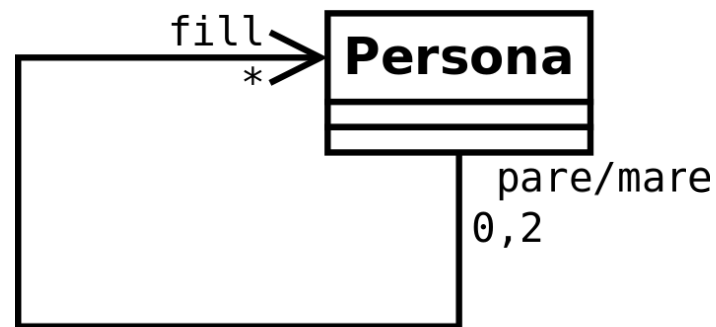


Navegabilitat SEMPRES
unidireccional del tot
cap a les parts*

*Les parts MAI poden invocar mètodes sobre el tot

Tipus especials d'associacions (2/2)

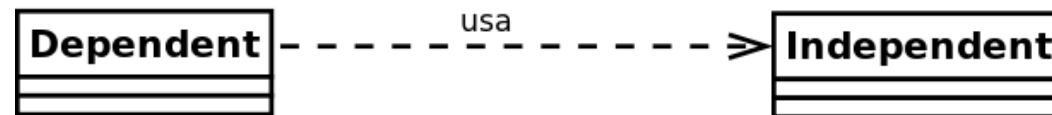
- **Associació reflexiva**: tipus d'associació entre dues classes, on la classe origen i la classe destí són la mateixa



- Tant els pares com els fills són objectes de tipus Persona
- Els pares poden tenir un nombre indeterminat de fills. En canvi, un fill sempre estarà enllaçat amb 2 objectes Persona, el seu pare i la seva mare
- Aleshores, perquè cal introduir també la cardinalitat 0 al cantó de pare/mare (0, 2)?

Relacions de dependència

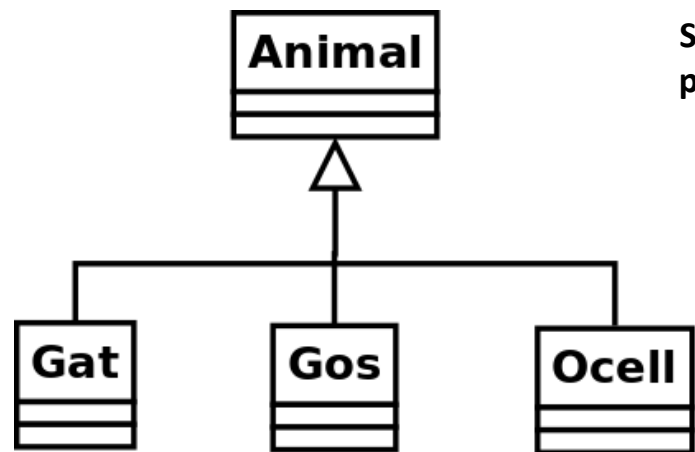
- S'estableix una **relació de dependència** entre dues classes quan una d'elles (la classe dependent) **utilitza l'altre** (classe independent)
- Diferència molt subtil enfront de les associacions:
 - Com veurem, les associacions s'acaben implementant mitjançant la definició d'atributs a les classes involucrades
 - Les relacions de dependència són més dèbils que l'associació: indiquen simplement que **la classe dependent utilitza mètodes de la classe independent, o objectes d'aquesta com a variables locals en els seus mètodes** (no requereix la definició d'atributs)



Fletxa discontinua des de la classe dependent cap a la classe independent, normalment amb el nom “usa”

Relacions de generalització/especialització

- S'utilitzen per representar relacions d'herència entre classes. **Relacionen una classe general** (superclasse) **amb una o més classes derivades** (subclasses) **que representen especialitzacions de la primera**
 - Les subclasses són especialitzacions de la superclasse: hereten totes les propietats i el comportament de la superclasse automàticament, proporcionant-ne d'addicionals (estenen la superclasse)

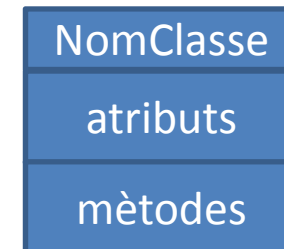


Superclasse que defineix totes aquelles propietats i comportament comú a tots els animals

Subclasses que hereten totes aquestes propietats i comportament comú a tots els animals, especialitzant-los per a cada tipus específic d'animal

Representació de les classes (1/2)

- En els diagrames de classes, les classes es representen de forma completa mitjançant un rectangle amb 3 compartiments:



- Representació d'atributs:

opcional
↓

 visibilitat nomAtribut : tipus [=valor inicial]

Visibilitat:

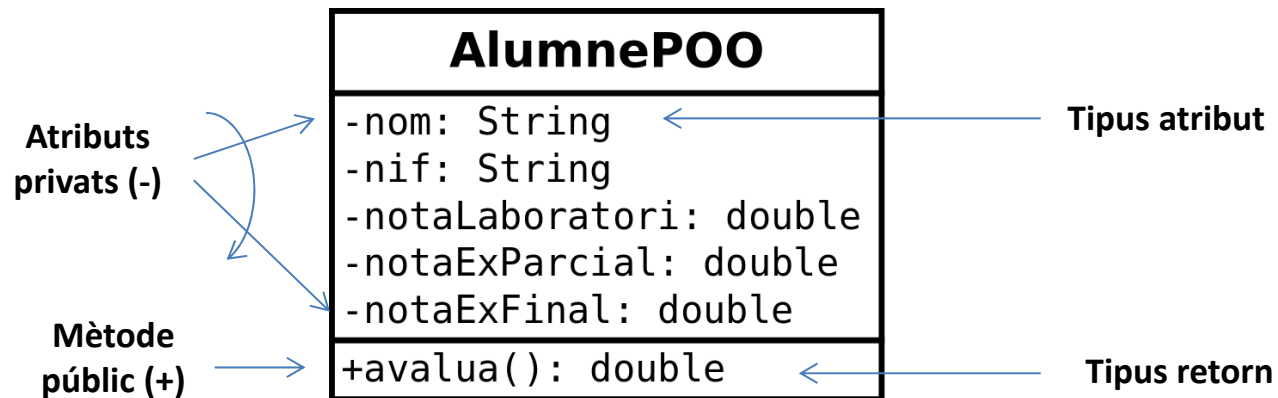
public (+)
private (-)
protected (#)

- Representació de mètodes:

visibilitat nomMetode (nomParam1:tipus, nomParam2:tipus,...) : tipusRetorn
- Els atributs i mètodes estàtics es subratllen en els diagrames de classes UML.
- A més, les constants s'escriuen en MAJÚSCULES (ja ho havíem comentat!)

Representació de les classes (2/2)

- **EXAMPLE**: Classe AlumnePOO en UML



- Els mètodes constructors i accessors (getters/setters) normalment NO es mostren en els diagrames de classes UML

Implementació de classes Java a partir de diagrames de classes UML

- Com a programadors, hem de ser capaços d'extreure l'estructura de les classes de la nostra aplicació a partir del diagrama de classes UML
- Hem de traduir a codi Java:
 - Contingut de les caixetes que representen les classes de l'aplicació
 - Relacions entre classes (**IMPORTANT**)
- Traduir el contingut de les caixetes és senzill (el que hem estat fent fins ara)
- Les associacions s'implementen mitjançant la definició d'atributs a les classes involucrades:
 - Per saber on hem de definir aquests atributs, i de quin tipus han de ser, ens fixarem en les cardinalitats i les navegabilitats de l'associació

Implementació de classes Java a partir de diagrames de classes UML

- Passos a seguir:

1. Navegabilitats: Hem de definir atributs en la classe origen en el sentit de la navegabilitat, del tipus de la classe destí

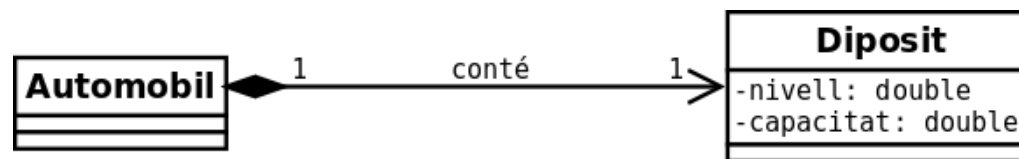
- Els objectes de la classe origen definiran atributs de la classe destí, sobre els quals invocaran mètodes (navegabilitat -> sentit flux invocacions de mètodes)

2. Cardinalitats: Ens indiquen la multiplicitat d'aquests atributs

- Si cardinalitat = 1: Un únic objecte de la classe destí
- Si cardinalitat determinada > 1 (p. ex. ,10): Array estàtic d'objectes de la classe destí (mida = cardinalitat)
- Si cardinalitat indeterminada (p. ex., *, 1..*): Com que de bon principi no sabem a quants objectes de la classe destí estarà enllaçat un objecte de la classe origen: Llista/Mapa/Conjunt d'objectes de la classe destí

Implementació de classes Java a partir de diagrames de classes UML

- Pregunta: quan usarem una Llista, Mapa o Conjunt per implementar una cardinalitat indeterminada?
 - Si l'ordre dels objectes de la classe destí importa: **Llista**
 - Si objectes de la classe destí unívocament identificables pel valor d'un dels seus atributs: **Mapa**
 - Si volem evitar objectes de la classe destí duplicats i l'ordre no importa: **Conjunt**
- **EXEMPLE 7.1:**

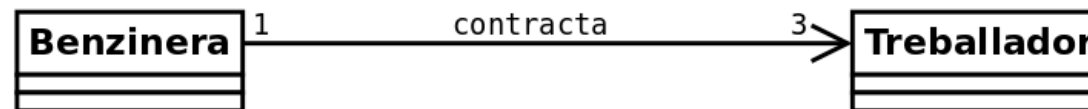


```
public class Automobil {
    private Diposit diposit;
}
```

```
public class Diposit {
    private double nivell;
    private double capacitat;
}
```

Implementació de classes Java a partir de diagrames de classes UML

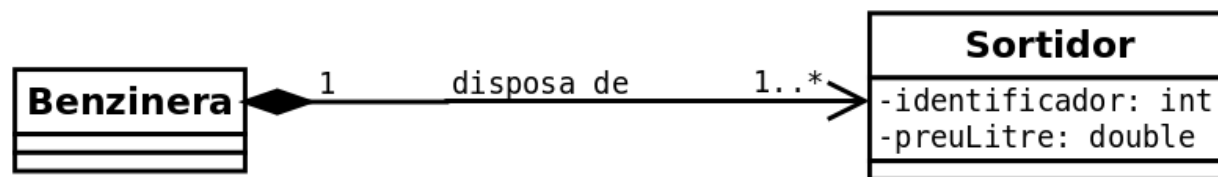
- **EXEMPLE 7.2:**



```
public class Benzinera {
    private Treballador[] treballadors;
}
```

```
public class Treballador{
}
```

- **EXEMPLE 7.3:**

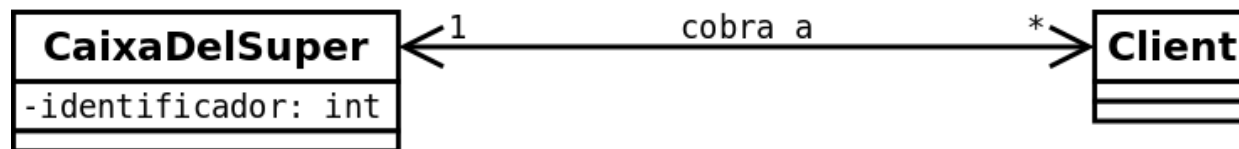


```
public class Benzinera {
    private HashMap<Integer,Sortidor> sortidors;
}
```

```
public class Sortidor{
    private int identificador;
    private double preuLitre;
}
```

Implementació de classes Java a partir de diagrames de classes UML

- **EXAMPLE 7.4:**



```

public class CaixaDelSuper{
    private int identificador;
    private ArrayList<Client> cua;
}
  
```

```

public class Client {
    private CaixaDelSuper caixa;
}
  
```