# Elements d'UML propis de l'etapa de disseny


## Elements d'UML propis de l'etapa de disseny

## Conceptes generals: visibilitat

La visibilitat *defineix quins objectes tenen dret a consultar i eventualment modificar informació declarada en un diagrama de classes*

La visibilitat dels elements d'un diagrama de classes pot influir en els factors de qualitat de l'arquitectura, per això cal establir-la en la seva vista lògica

En UML, la visibilitat s'estableix sobre:

- Atributs d'una classe
- Operacions d'una classe
- Rols d'associacions accessibles des d'una classe

En UML, l'element *x* definit a la classe *C* pot ser:

- *Public* (+): qualsevol classe que veu *C*, veu *x*
- *Private* (-): *x* només és visible des de *C*
- *Protected* (#): *x* és visible des de *C* i des de tots els descendents de *C* (si n'hi ha)



## Conceptes generals: àmbit

L'àmbit *defineix si determinats elements de disseny són aplicables a objectes individuals o a la classe* que defineix els elements

En UML, l'àmbit s'estableix sobre:

- Atributs d'una classe
- Operacions d'una classe

En UML, l'element *x* definit a la classe *C* pot ser d'àmbit:

- D'instància (no estàtic): *x* està associat als objectes de *C*
  referència: obj.x, essent *obj* una instància de la classe *C*
- De classe (estàtic): *x* està associat a *C*
  referència: C.x

## Atributs: sintaxi completa

*[visibilitat] nom [: tipus] [multiplicitat] [= valor-inicial] [{propietats}]*

**Univaluat** (per defecte)
**Admet valors nuls**: [0..1]
**Multivaluat**: [1..*]
etc.

*No les usem en aquesta assignatura*

**Public** (+): qualsevol classe que veu la classe, veu l'atribut
**Protected** (#): només la pròpia classe i els seus descendents veuen l'atribut
**Private** (-): només la pròpia classe veu l'atribut

*Suposarem que els atributs són privats, a no ser que especifiquem una altra visibilitat*

| Persona |
| --- |
| + dni: String<br># nom: String<br>- adreça: String = "C/Nou"<br>telèfon : Integer [0..1] |

**operacions** *getter* **i** *setter*

obtéAdreça(): String
posaAdreça(n: String)



## Atributs: Àmbit

Atribut d'instància:

- representa una propietat aplicable a tots els objectes d'una classe.
- cada objecte pot tenir un valor diferent d'aquest atribut.

Atribut de classe:

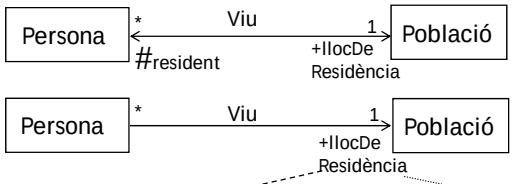
- propietat aplicable a la classe d'objectes com a tal.
- no és aplicable a les instàncies de la classe.

| Persona |
| --- |
| dni: String<br>nom: String<br>adreça: String<br>edat: Natural<br>mitjanaEdats: Real |

*atribut d'instància*

*atribut de classe*

## Associacions




Navegabilitat

Persona a Població
Població a Persona

Persona a Població

*visibilitat*: public, protected, private (com abans)

*Suposarem que els rols són privats, a no ser que especifiquem el contrari*

**Navegabilitat:** *resultat del procés de disseny*

- Indica si és possible o no travessar una associació binària d'una classe a una altra:
  - si *A* és navegable cap a *B*, des d'un objecte d'*A* es poden obtenir els objectes de *B* amb els què està relacionat
  - efecte: *A* té un pseudo-atribut amb la mateixa multiplicitat del rol
- Les associacions no navegables en cap sentit podrien desaparèixer de la vista lògica
  - però les podem deixar si el disseny és incomplet o per motius de canviabilitat (extensibilitat)
- La navegabilitat té un fort impacte en l'avaluació de l'arquitectura



## Operacions: sintaxi completa

*No les usem a aquesta assignatura*

Signatura d'una operació:

*[visibilitat] nom [(llista-paràmetres)] [: tipus-retorn] [{propietats}]*

**Public** (+): l'operació pot ser invocada des de qualsevol objecte
**Protected** (#): pot ser invocada pels objectes de la classe i els seus descendents
**Private** (-): només els objectes de la pròpia classe poden invocar l'operació

*Suposarem que les operacions són públiques, a no ser que especifiquem una altra visibilitat*

Paràmetres d'una operació:

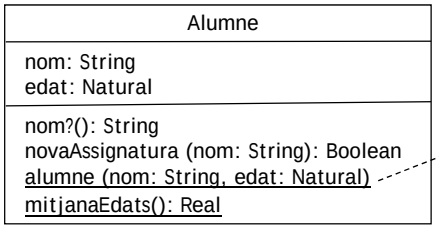
*[direcció] nom: tipus [multiplicitat] [= valor-per-defecte]*

*in*, out, inout

*Valor per defecte*

## Operacions: àmbit

- Operació d'instància:
  - l'operació és invocada sobre objectes individuals
- Operació de classe:
  - l'operació s'aplica a la classe pròpiament dita
  - exemple: operacions constructores que serveixen per donar d'alta noves instàncies d'una classe

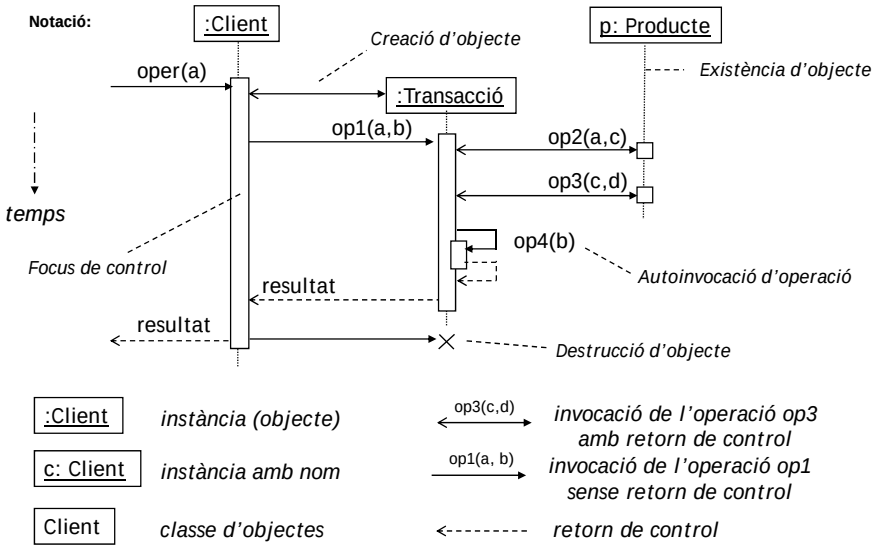



*Operació constructora:*

*Suposarem que, si no s'indica el contrari, cada classe d'objectes té una operació constructora amb tants paràmetres com atributs té la classe. Si es consideren altres constructores, caldrà definir-ne la seva signatura.*



## Diagrames de seqüència: sintaxi completa



## Diagrames de seqüència: convencions (1)

Noms dels objectes: batejar-los quan calgui per identificar el seu origen
- si són resultat d'una operació, usar el mateix nom en el resultat i en l'objecte
- si s'obtenen recorrent una associació amb multiplicitat 1, usar el nom del rol
- si han arribat com a paràmetres, usar el nom dels paràmetres

Paràmetres de les operacions:
- cal indicar explícitament amb quins paràmetres s'invoquen les operacions

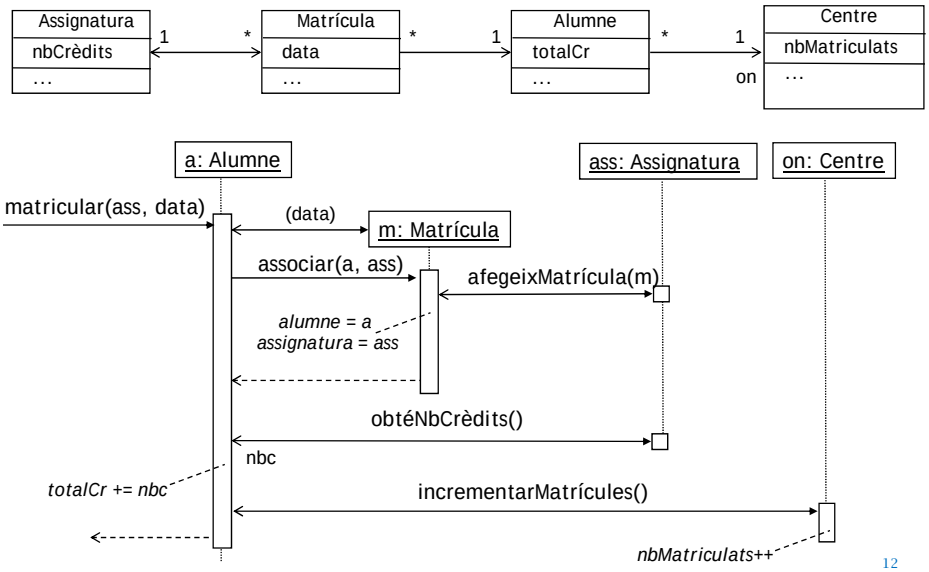
Resultats de les operacions:
- donar nom al resultat si surt en algun altre lloc del diagrama
- també als valors retornats en els paràmetres

Comentaris:
- no deixar cap aspecte rellevant sense comentar
- en particular, ha de quedar clar:
  - ✓ com es calculen els resultats de les operacions
  - ✓ Quins valors es passen com a paràmetres a les operacions creadores
  - ✓ com es modifiquen els valors dels atributs i pseudo-atributs de les classes
- usar noms d'atributs, associacions, etc.
  - ✓ usar sintaxi Java per a operacions aritmètiques

No cal especificar el comportament de les operacions *getter* i *setter*



## Diagrames de seqüència: convencions (2)



## Agregats

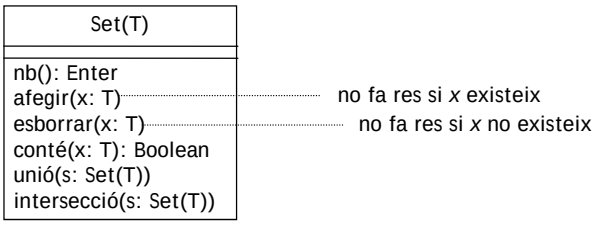
Un agregat és una col·lecció d'objectes

Sorgeixen en diversos contexts:

- Rols navegables amb multiplicitat més gran que 1
- Operacions que reben o retornen una col·lecció de valors
- Atributs multivaluats

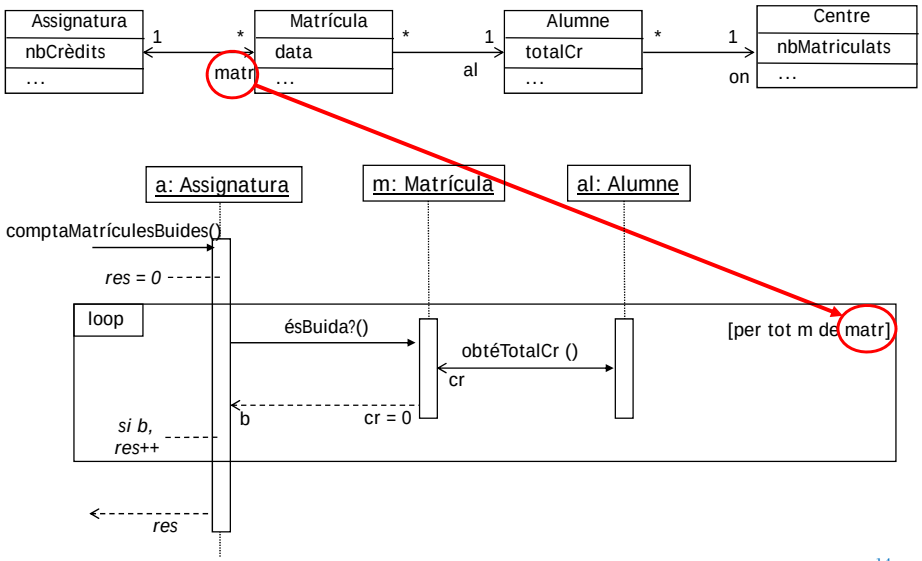
En tots aquests casos, considerem l'agregat com un conjunt (Set)

- S'hi poden aplicar operacions següents (i només aquestes):

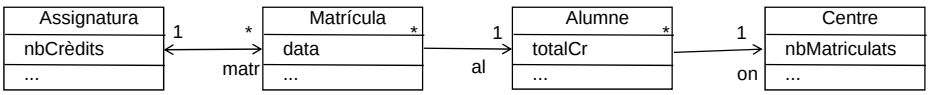


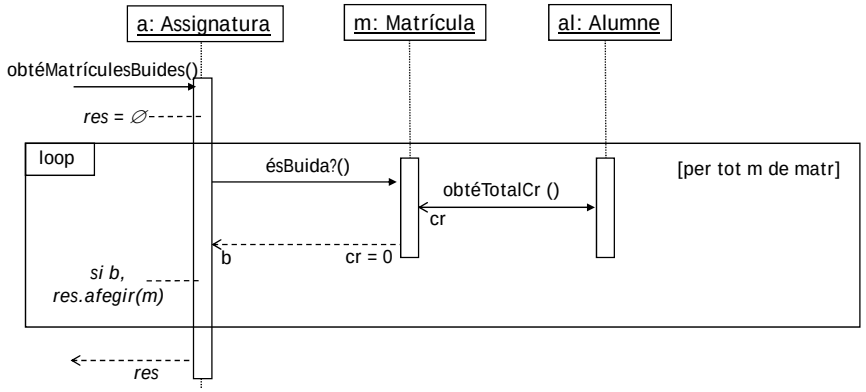



## Diagrames de seqüència amb agregats (1)



## Diagrames de seqüència amb agregats (2)






## Bibliografia


- Larman, C. "*Applying UML and Patterns. An Introduction to Object-oriented Analysis and Design*", Prentice Hall, 2005, (3ª edició).
- http://www.uml.org/#UML2.3
- Meyer, B. "*Object-Oriented Software Construction*", Prentice Hall, 1997, cap. 3
- Martin, R.C., "*Agile Software Development: Principles, Patterns and Practices* ", Prentice Hall, 2003, caps. 7 i 9.