Intensiu parcial

(Traducció i Disseny de diagrames de seqüència)

Academia ASES

Contacte:

Max Tiessler Aguirre

max.tiessler@estudiantat.upc.edu

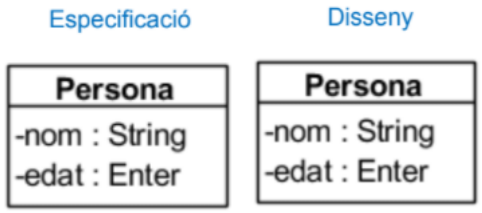
Continguts intensiu FHC 3

1-Traducció d'especificació a disseny

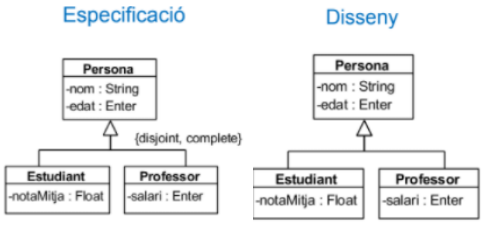
Per tal de poder construir el codi font d'una aplicació s'han de fer uns canvis als diagrames d'especificació, respectant la semàntica del diagrama inicial, haurem de fixar-nos si els elements són **compatibles** o **no compatibles**.

Elements compatibles

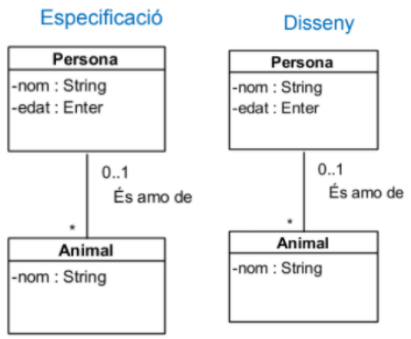
1) Classes no associatives:



2) Herències disjoint

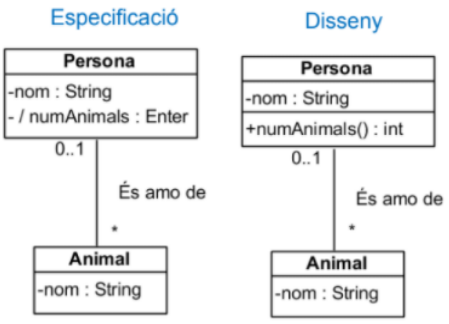


3) Associacions binàries

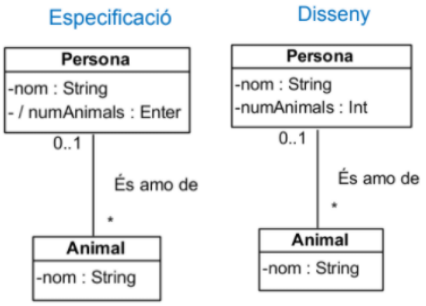


Elements no compatibles

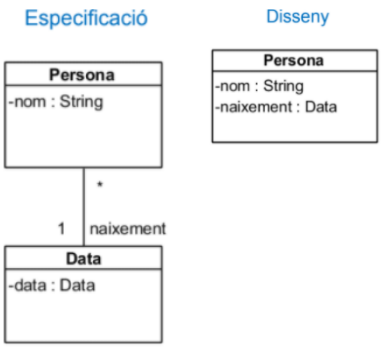
1) Atributs Derivats
    a) Calculats



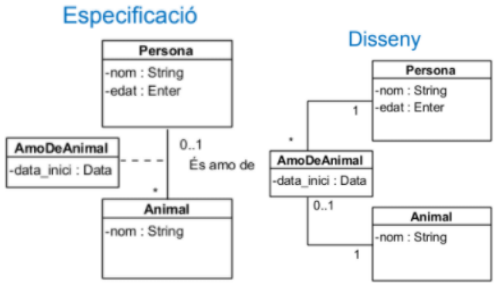
    b) Materialitzats



2) Classes Data / Hora

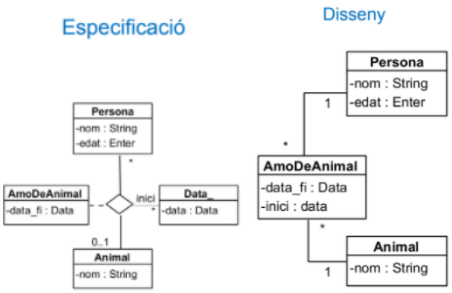


3) Classes Associatives



+ **afegir restricció textual**: *No hi pot haver dos "AmoDeAnimal" amb els mateixos Persona i Animal*

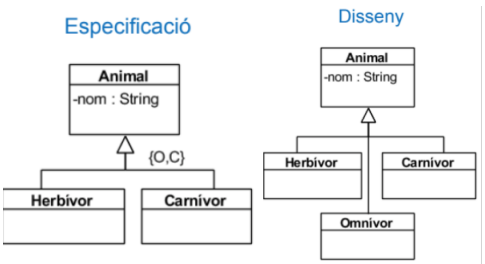
4) Associacions N-àries



+ **afegir restriccions textuals**
    + *RT1) No hi pot haver dos "AmoDeAnimal" amb els mateixos Persona Animal i Inici* → Pel fet que queda igual que si haguéssim transformat una associativa
    + *RT2) Donada una persona i una data, màxim aquesta pot esdevenir amo d'un animal* → Pel fet de mantenir la informació que hi havia a la ternària

Important → RT1 és redundant amb RT2, mantenim RT2.

5) Herència Overlapping



2-Conceptes Disseny

Visibilitat

Defineix quins objectes poden llegir/modificar informació d'altres elements, com poden ser: **Atributs, Operacions o Rols**.

Hi ha tres tipus de visibilitats:

1) Privada (-): Només les pròpies instàncies de la classe poden consultar-ho.
2) Pública (+): Tothom pot accedir-hi.
3) Protegida (#): Només poden accedir-hi instàncies de la mateixa herència.

[IMPORTANT] A IES i en general assumirem:

- Atributs i rols: **privats.**
- Operacions: **públiques.**

Òbviament no serà sempre així fora de l'assignatura.

Àmbit

Determina si els **atributs** o **operacions** són aplicables a objectes individuals o a la classe que defineix aquests objectes.

Per tant, tenim dos tipus d'àmbit:

- De classe (atributEstatic): **estàtic**
- D'instància (atributNoEstatic): **no-estàtic**

Excepcions

A l'hora de dissenyar ja **no tenim cap sistema per assegurar que es compleixin les R.T**, per tant, el concepte de *precondició* perd el sentit, ara ho haurem de garantir nosaltres mitjançant **excepcions**.

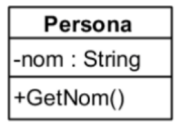
3-Diagrames de seqüència

Representen interaccions entre objectes i la seva temporalitat.

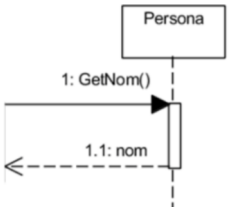
Crides i resultats (getters)

A una instància:

D. Classes

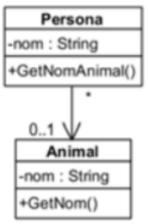


D. Seqüència

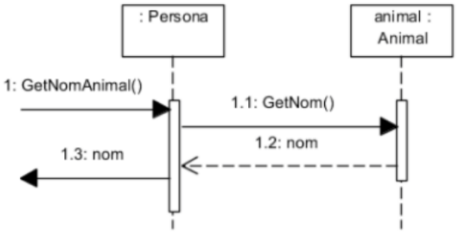


Entre instàncies:

D. Classes



D. Seqüència



Si una classe es comunica amb una altra, ho marquem amb una fletxa, és el que anomenem **navegabilitat.**

Navegabilitat

Serveix per indicar si es pot i en quina direcció travessar una associació binària.

Pot ser:

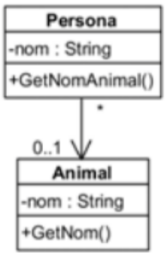
- Unidireccional (exemple)
- Bidireccional
- No Navegable

Opcionals / condicionals

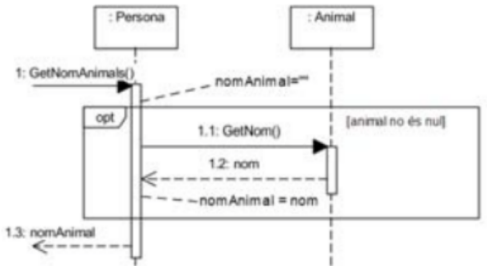
Podem afegir **frames** per tenir estructures opcionals, alternatives o de repetició.

- **Opt:** en el cas que s'hagi de complir una condició concreta (un if)
- **Alt:** si tenim diverses condicions (if..else)

D. Classes

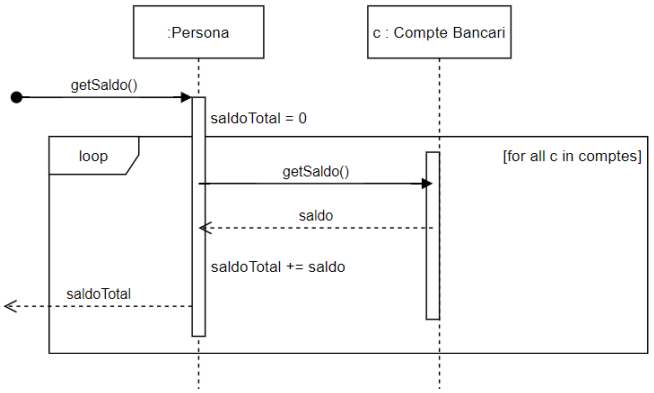


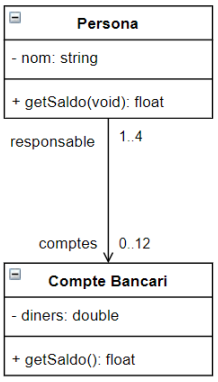
D. Seqüència



Conjunts - "agregats"

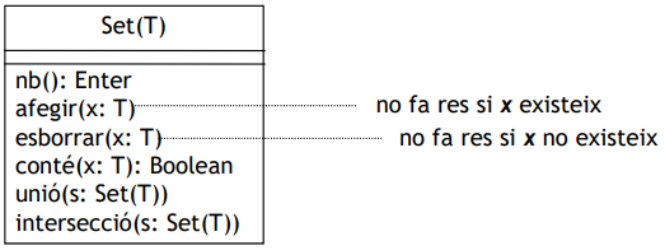
La manera de tenir una col·lecció d'objectes es fa mitjançant agregats. Aquest sets es poden recorre mitjançant loops:





Els sets apareixen en

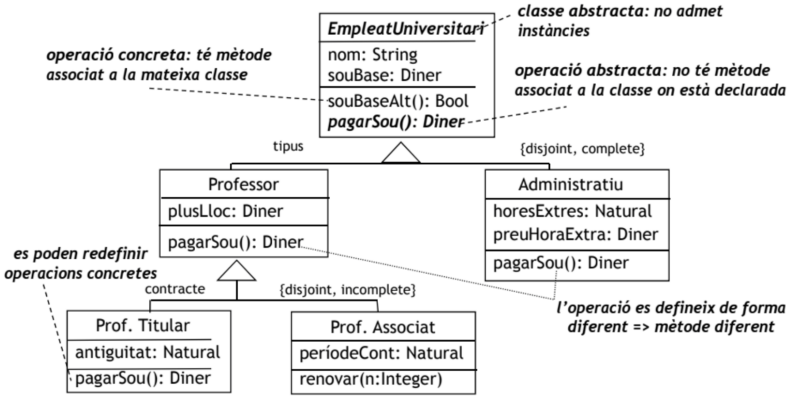
- relacions amb alguna cardinalitat major que 1.
- Operacions que retornen una col·lecció de valors.
- Atributs multivaluats.



Herència i polimorfisme

Capacitat de poder redefinir el comportament de les funcions, tenint en compte el tipus de subclasse. Als diagrames de classes es pot usar el polimorfisme creant operacions amb el mateix nom a les subclasses.

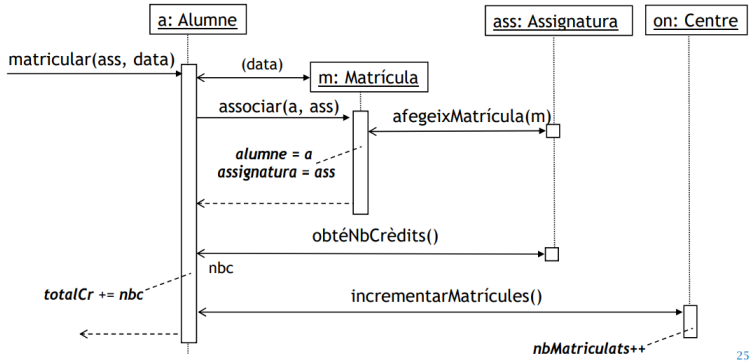
A més, podem tenir també operacions **abstractes,** aquest fet obliga a tots els descendents a definir aquesta funció. Una classe només pot ser abstracta si la herència és de tipus *complete*. **Als diagrames de IES es poden indicar les classes abstractes amb un comentari (o escrivint en cursiva).**



Creadores

Creen nous objectes amb els paràmetres necessaris. A una jerarquia, estem **obligats** sempre a cridar a la constructora de la superclasse. A IES existeix *per se* una creadora amb tots els atributs com a paràmetre.

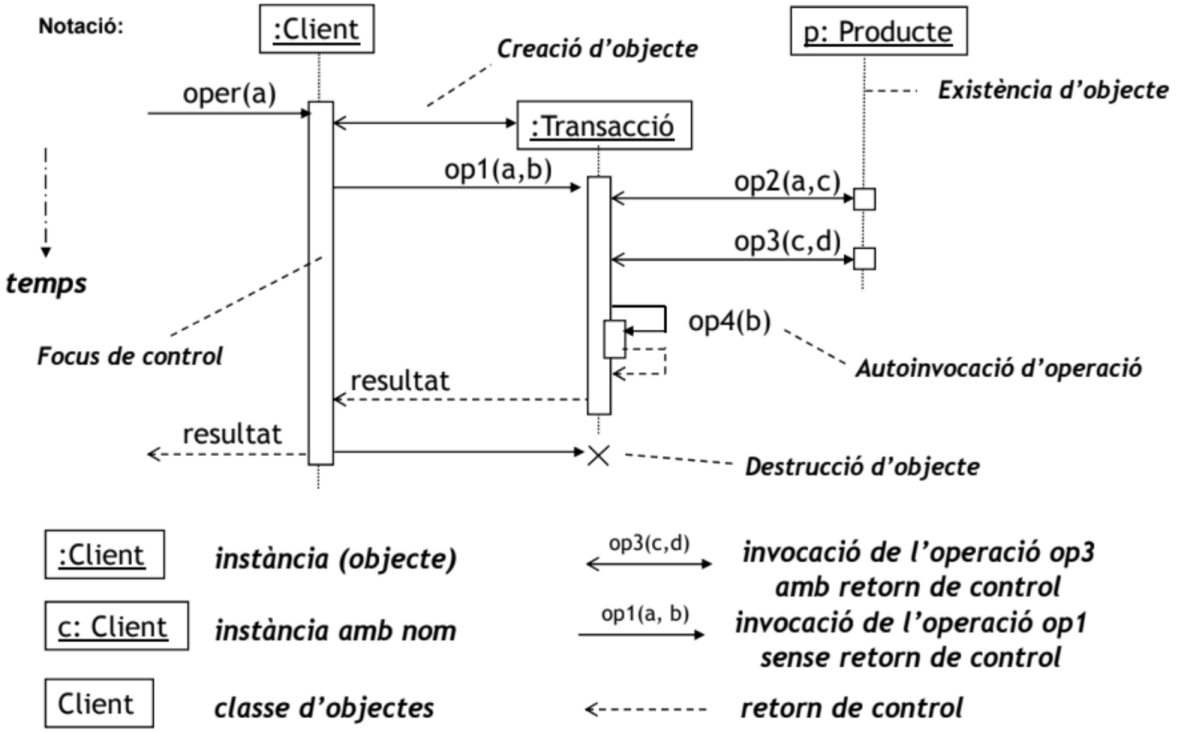
El objectiu és que totes les navegabilitats que **surten** de la classe quedin assignats a la nova instància.



Destructores

L'objectiu és que totes les instàncies que tenen navegabilitat que **entra** a la instància que volem destruir, n'eliminin la seva associació.

Resum



4-Conceptes Arquitectura del Software

Busquem una arquitectura de qualitat, a IES considerarem dos propietats que són indicadores d'una bona arquitectura:

1) Acoblament

Indica el grau de connexió i dependència d'una classe.
Una classe està acoblada amb una altra si existeix alguna relació (navegabilitat, atribut d'un tipus d'una altra classe, referència a una operació d'una altre classe) que en fa dependre'n.
Un baix acoblament farà que les classes siguin més reusables.
Per tant, busquem un **baix acoblament**.
Com a regla general, una classe només ha de "parlar" amb objectes i mètodes que coneix (llei de Demèter).

2) Cohesió

Indica com de concentrades i especialitzades estan les classes i funcions.
Busquem una **cohesió alta**, volem que les classes representin només un concepte i les funcions només facin una cosa i la facin bé.

Principi Obert - Tancat

Els mòduls (classes, funcions…) han d'estar **oberts** a extensions però **tancats** a modificacions

- **Oberts per a l'extensió**. El comportament del mòdul es pot estendre per tal de satisfer nous requisits.
- **Tancats per a la modificació**. L'extensió no implica canvis en el codi del mòdul. No s'ha de tocar la versió executable del mòdul.

Arquitectura 3 Capes

Tota arquitectura ha d'estar dividida en 3 capes:

- **Capa de presentació:**

  S'encarrega de rebre inputs i extreure els outputs.
- **Capa de Domini:**

  S'encarrega de la lògica de negoci.
- **Capa de Dades:**

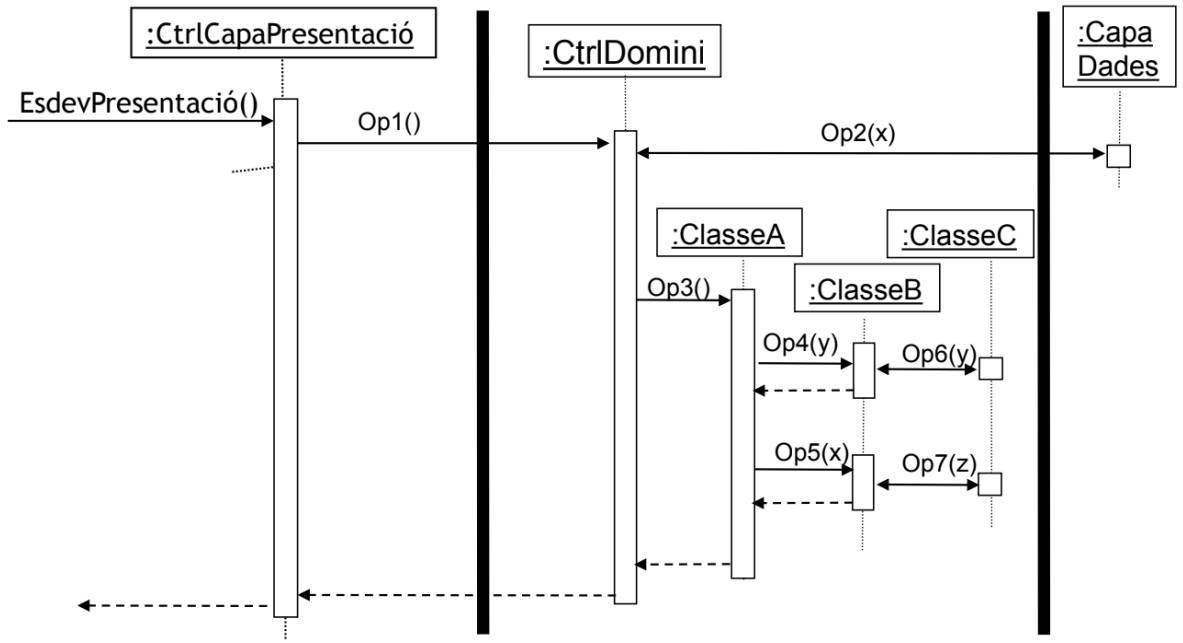
  S'encarrega de gestionar les dades que hi ha dins de l'execució del programa, a més de gestionar la persistència.

Patrons de disseny

Solució estandarditzada per a un problema que es repeteix en el temps.
A IES hem de tenir en compte els següents patrons:

Patró *Domain Model*

La lògica de l'aplicació resideix a la capa de domini.



Patró transacció

S'introdueix una classe abstracta que actua de superclasse de tots els controladors transacció del sistema.

Una classe transacció representa una operació que s'efectuarà en el sistema.