
TEMA 4 INTRODUCCIÓ A LA ORIENTACIÓ A OBJECTE

1. Evolució dels llenguatges de programació.
2. Paradigmes de programació.
3. Principis bàsics de l'O.O.
4. Conceptes bàsics de l'O.O.

1

4.1

Evolució dels llenguatges de programació (I)

- Llenguatges de primera generació (1954-1958).
 - Assemblador, en els inicis.
 - FORTRAN I, ALGOL 58, Flowmatic, IPL V, etc.
 - Són llenguatges basats en el procés d'expressions matemàtiques.
- Llenguatges de segona generació (1959-1961).
 - FORTRAN II (subrutines, compilació separada).
 - ALGOL 60 (estructura de blocs, tipus de dades).
 - COBOL (descripció de dades, gestió de fitxers).

2

Evolució dels llenguatges de programació (II)

- Llenguatges de tercera generació (1962-1970).
 - BASIC (poques instruccions, fàcil d'aprendre). Objectius didàctics.
 - PL/1 (FORTRAN + ALGOL + COBOL). Es un llenguatge massa gran.
 - ALGOL 68 (ortogonalitat: cap restricció a les components del llenguatge (instruccions, e.d., etc.). Esdevé inimplementable.
 - Pascal (successor d'ALGOL 60 amb influències d'altres llenguatges). Influït per les noves tècniques aparegudes després de la crisi del software (verificació de programes, programació estructurada, etc.).
 - Simula (classes, abstracció de dades).

Evolució dels llenguatges de programació (III)

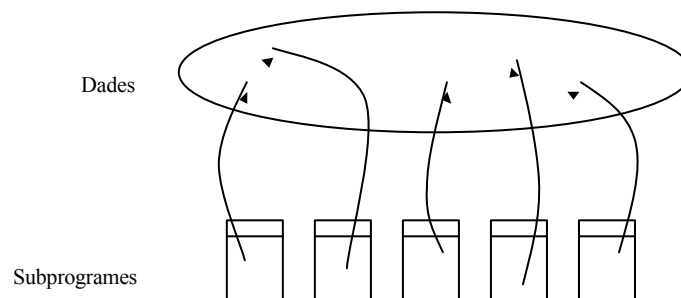
- Generació de successors de la tercera generació (1970-1990).
 - Apareixen molts llenguatges d'alt nivell dissenyats per programar amb un cert estil, però pocs perduren. CLU, ALPHARD, C, MODULA, EUCLID.
 - S'introdueix la idea de programació paral·lela i programes concurrents.
 - ADA (successor d'ALGOL 68 + PASCAL, construccions modulars, registres amb parts variables, apuntadors, concurrència, tractament d'excepcions, sobrecàrrega).
 - CLOS (Lisp OO).
 - C++ (C + Simula).

Llenguatges Orientats a Objecte.

Evolució dels llenguatges de programació (IV)

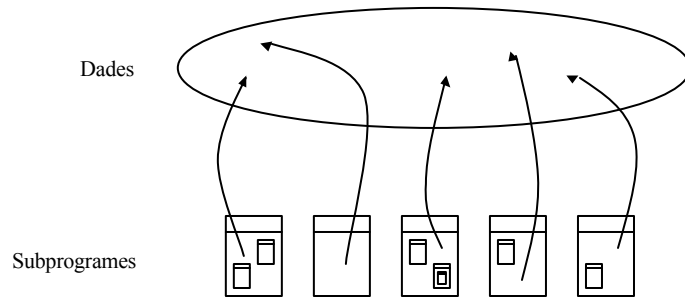
- Quarta generació (1981-...).
 - Acceleren el procés d'automatització d'aplicacions, reduir costos de manteniment, permetre als usuaris finals solucionar els problemes.
 - Poden incloure formats de pantalla, formats d'informes, estructures de diàleg, associacions entre moltes dades, proves de validació, controls de seguretat, etc.
 - Eines CASE, programació visual, llenguatges d'accés a BD, etc.
- Cinquena generació (1960-...).
 - Llenguatges d'intel·ligència artificial: sistemes basats en el coneixement, sistemes experts, mecanismes d'inferència o processament del llenguatge natural.
 - LISP (Llenguatge funcional, llistes, apuntadors). Aparegut al 1960.
 - PROLOG (primer llenguatge de programació lògica). A mitjans dels 70.
 - ML, Miranda, etc. (nous llenguatges funcionals).

Topologia de LP de 1a. i inicis de 2a. generació



4.1

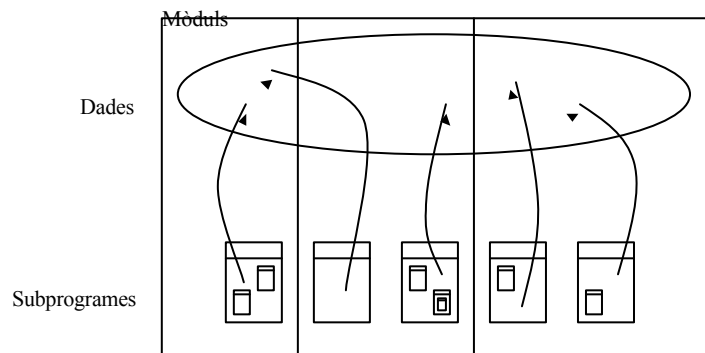
Topologia de LP de finals de 2a. i inicis de 3a. generació



7

4.1

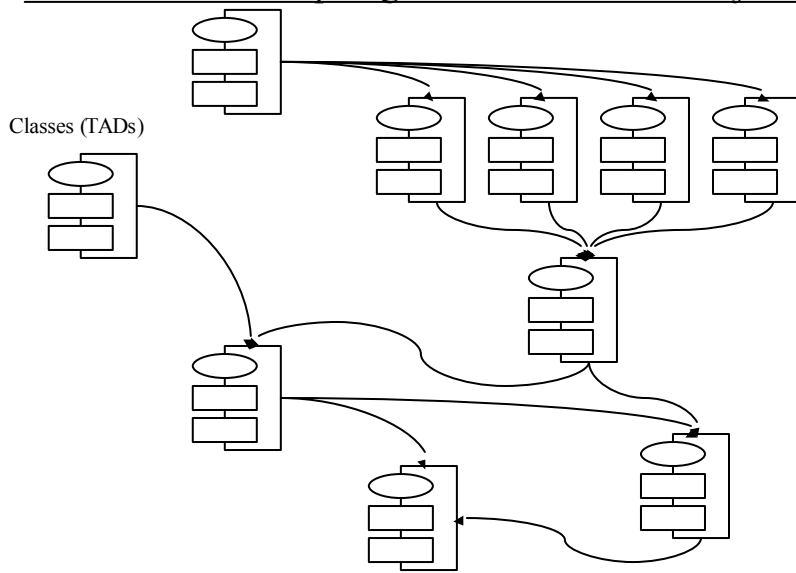
Topologia de LP de finals de 3a. generació



8

4.1

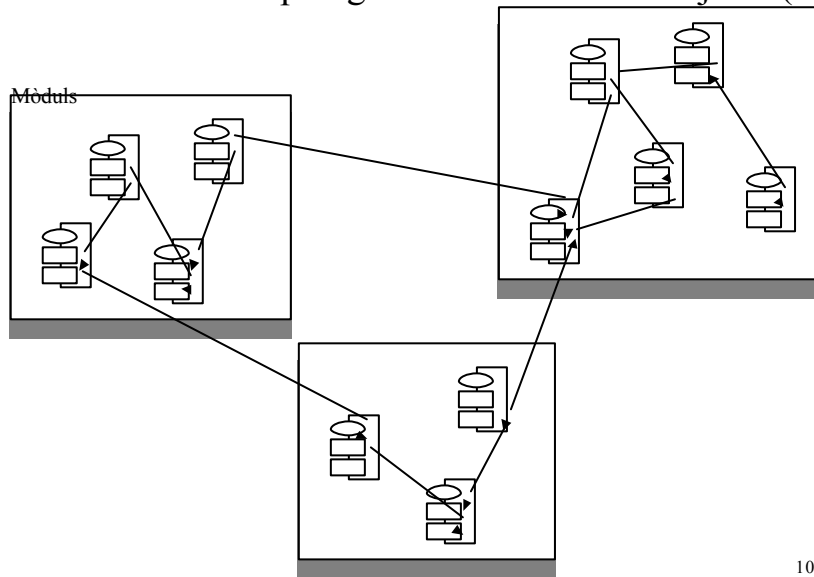
Topologia de LP orientats a objecte (I)



9

4.1

Topologia de LP orientats a objecte (II)



10

Paradigmes de programació

- **Llenguatges imperatius.** Variables, assignació, seqüència, iteració, alternativa. *Algorismes*.
- **Llenguatges declaratius.** *Expressius* (poca dificultat a escriure la descripció de la solució), fort *suport matemàtic*. A mig camí entre el programa i l'especificació.
 - **Llenguatges funcionals.** No tenen instruccions sinó que són *funcions* compostes per altres funcions. Absència d'assignació.
 - **Llenguatges lògics.** Relacions (predicats) entre objectes (dades) => *Regles i fets*. Conjunt d'objectius, sovint expressats en base al càlcul de predicats.
 - **Llenguatges basats en regles.** Regles *if-then*. Sistemes experts.
- **Llenguatges orientats a objecte.** Estil de programació caracteritzat per la manera de gestionar la informació, basant-se en conceptes de *classe, objecte i herència*.
- **Llenguatges de flux de dades.** Execució en paral·lel del màxim nombre de càlculs. S'utilitza un DFD per expressar dependències entre operacions.
- **Llenguatges basats en restriccions.** Relacions invariants.

Principis bàsics de l'orientació a objectes

Paradigma de Orientat a Objecte

Abstracció

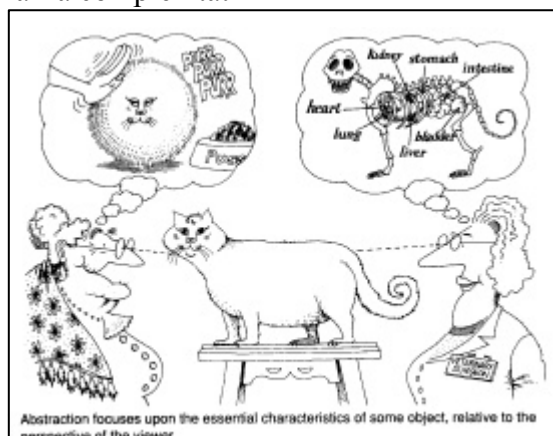
Encapsulació

Modularitat

Jerarquia

- Capacitat per representar (veure), de manera abstracta, les entitats que pertanyen al domini de la solució, i els recursos que ofereixen.
- Una abstracció denota les característiques essencials d'un objecte que permeten diferenciar-lo d'altres tipus d'objectes i, així, defineix els límits conceptuals des del punt de vista de l'usuari.
- Exemple: gestió acadèmica. Entitats *alumne*, *professor*, *PAS* amb les operacions necessàries per donar-los d'alta, validar el password, etc. Potser aquestes entitats es poden concebre com objectes similars (persones universitat). Entitat *matrícula*, amb les seves operacions, etc.

Ens permet gestionar la complexitat



Reproduït de *Object Oriented Design with Applications* (G. Booch), p. 39.

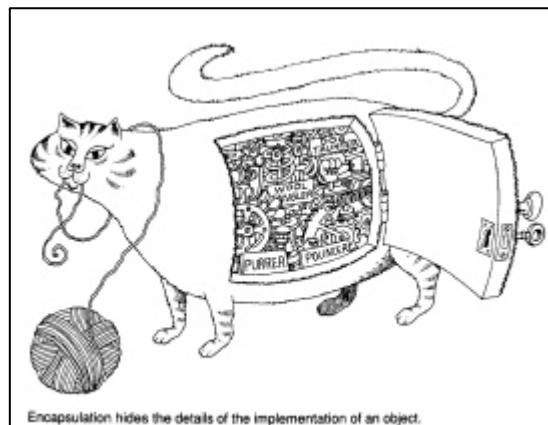
Encapsulament

- L'objectiu de l'encapsulament és ocultar els detalls procedimentals (implementació de l'objecte).
Complementa el concepte d'abstracció en el sentit que l'abstracció es preocupa de la visió externa d'un objecte i l'encapsulament evita que els clients en vegin els detalls interns.
- Mitjançant l'encapsulament oferim:
 - Una *frontera* clara que delimita l'àmbit de tots els objectes interns del software. Frontera entre els diferents nivells d'abstracció.
 - Una *interfície* que descriu com els objectes interactuen amb altres objectes.
 - Una *implementació interna protegida* que dona els detalls de la funcionalitat que ofereixen els diferents objectes.
- Exemple: separació entre representació física de les dades i esquema en una aplicació de bases de dades.

Encapsulació

Ocultar la implementació als usuaris

- Els usuaris depenen de la interfície



Reproduït de *Object Oriented Design with Applications* (G. Booch), p. 46.

4.4

Conceptes bàsics de la orientació a objectes

- Objecte
- Classe
- Atribut
- Operació
- Interfície (Polimorfisme)
- Component
- Paquet
- Subsistema
- Relacions

21

4.3

Objecte

De manera informal, un objecte representa una entitat física, conceptual o software.

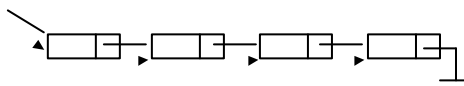
- Entitat física



- Entitat conceptual

Procés Industrial

- Entitat software



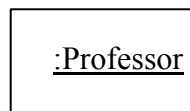
22

Un objecte és un concepte, abstracció o element amb límits definits i amb significat per una aplicació

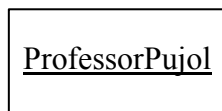
Un objecte te:

- Estat
- Comportament
- Identitat

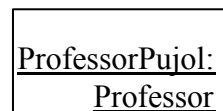
Un objecte es representa amb un rectangle amb el nom subrallat



Nom de la classe



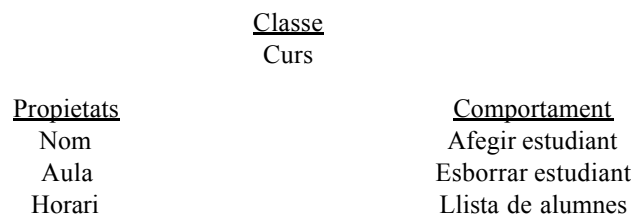
Nom de l'objecte



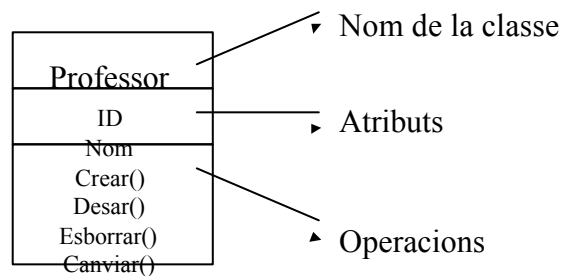
Nom de l'objecte-classe

Una classe es una descripció d'un grup de objectes amb propietats (atributs), comportament (operacions), relacions.

Implementa el principi d'Abstracció



Una classe es representa utilitzant un rectangle dividit en compartiments

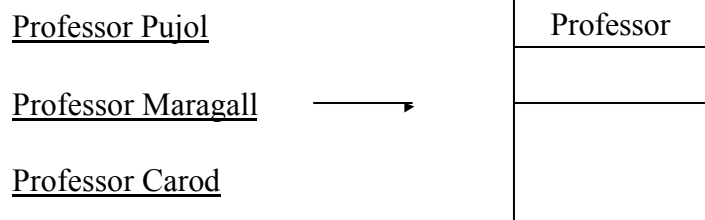


Relacions entre Classes i Objectes

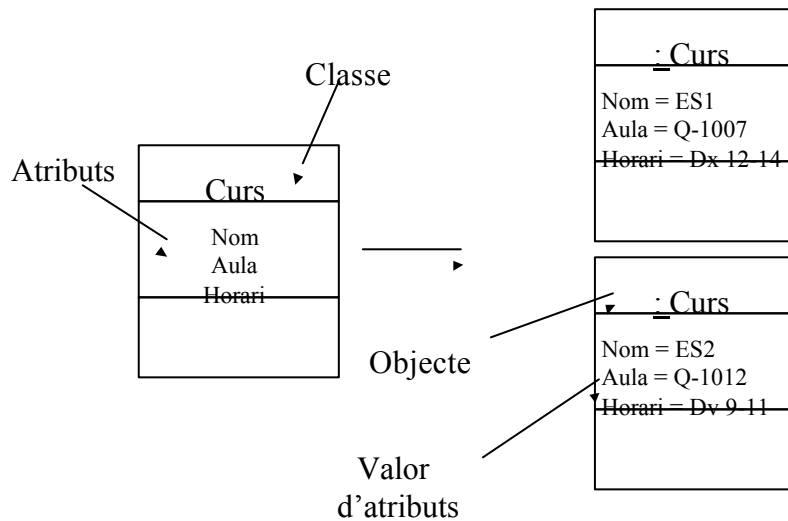
Una classe es una representació abstracta d'un objecte:

- Defineix l'estructura i el comportament de cada objecte en la classe
- Serveix com a patró per la creació d'objectes

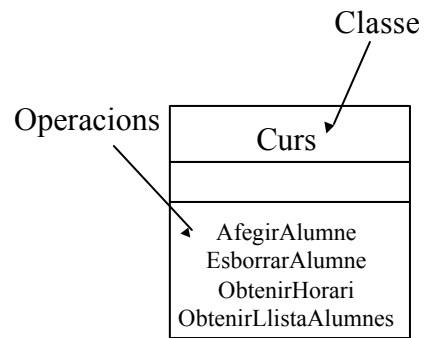
Els objectes s'agrupen en classes



Atributs



Operacions



29

Polimorfisme

L'habilitat d'ocultar diferents implementacions darrere una única interfície.



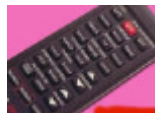
Fabricant A



Fabricant B



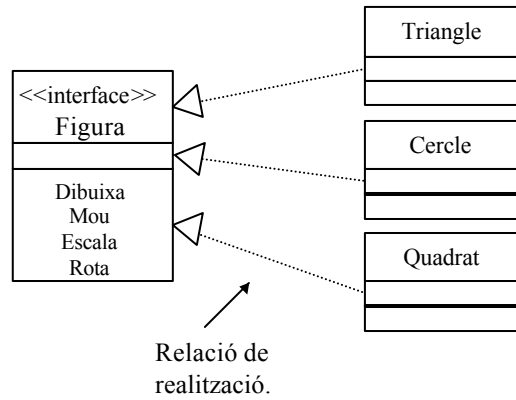
Fabricant C



Principi OO:
Encapsulament.

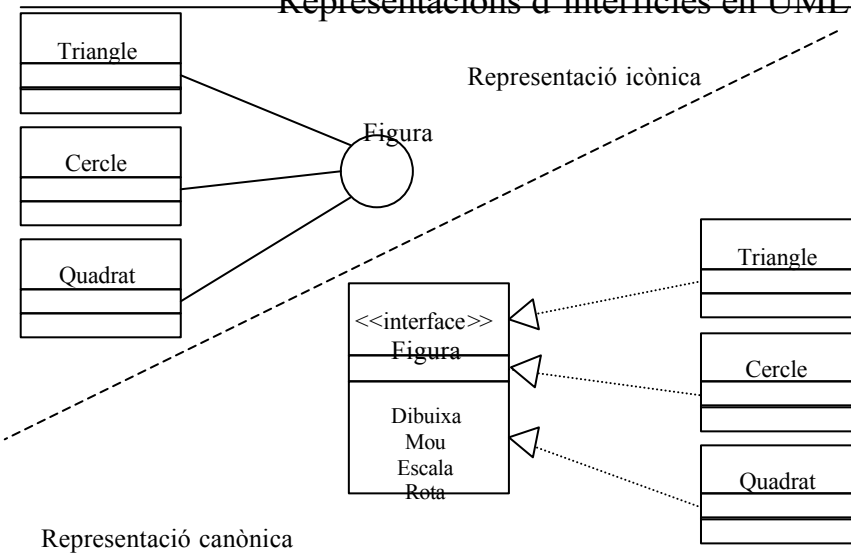
30

- Les interfícies formalitzen el polimorfisme.
- Les interfícies suporten arquitectures “plug&play”.



31

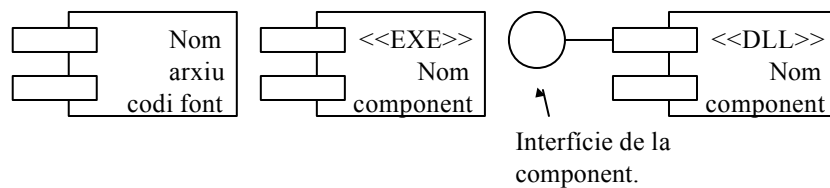
Representacions d'interfícies en UML



32

Component

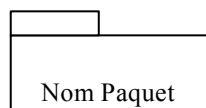
- Una part física i reemplaçable del sistema que s'adapta i proporciona la realització d'un conjunt d'interfícies.
- Una component pot ser:
 - Una component de codi font.
 - Llibreries dinàmiques (DLL).
 - Components executables.



Principi OO: Encapsulament.

Paquet

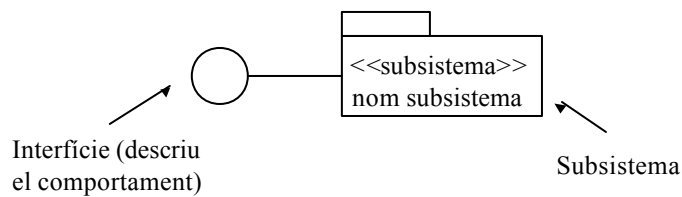
- Un paquet és un mecanisme de propòsit general que permet organitzar elements en grups.
- És un element de modelat que pot contenir altres elements de modelat.
- Usos:
 - Organitzar el model.
 - Una unitat de gestió de la configuració.



Principi OO: Modularitat.

Subsistema

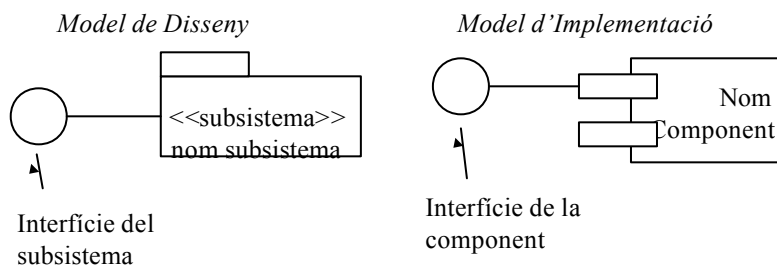
Una combinació d'un paquet (pot contenir altres elements de modelat) i una classe/interfície que defineix el seu comportament. Ho podem entendre com una abstracció a més alt nivell que el que representa una classe.



Principi OO: Encapsulament i modularitat.

Subsistemes i components

- Les components són la realització física d'una abstracció en el disseny.
- Els subsistemes poden ser utilitzats per representar les components en el disseny.

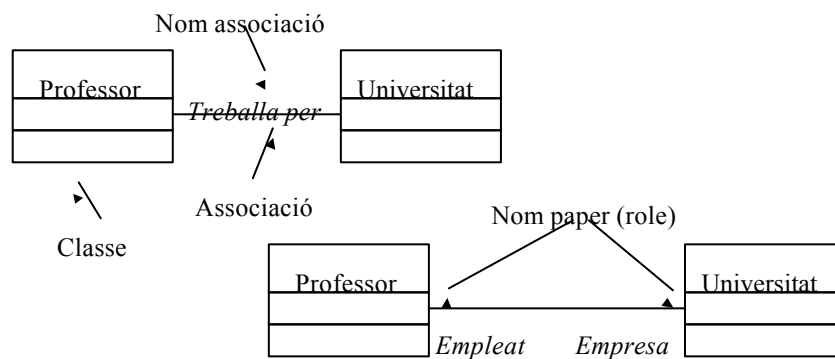


Principi OO: Encapsulament i modularitat.

- Associació
- Agregació / Composició
- Dependència
- Generalització

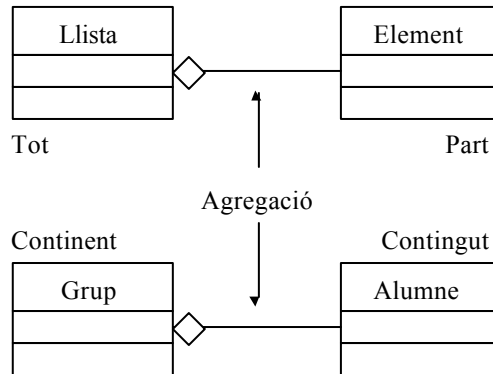
Relacions: Associació

Modela una connexió semàntica entre classes. És una relació estructural que especifica que els objectes d'una classe estan connectats als objectes d'un altre. Es pot navegar des d'un objecte d'una classe a un de l'altra.



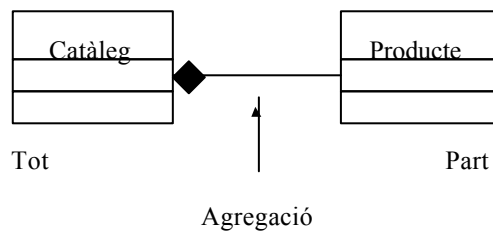
Relacions: Agregació

Una forma especial d'associació que modela una relació “tot-part” o de “continent-contingut” entre un agregat (el tot) i les seves parts.



Relacions: Composició

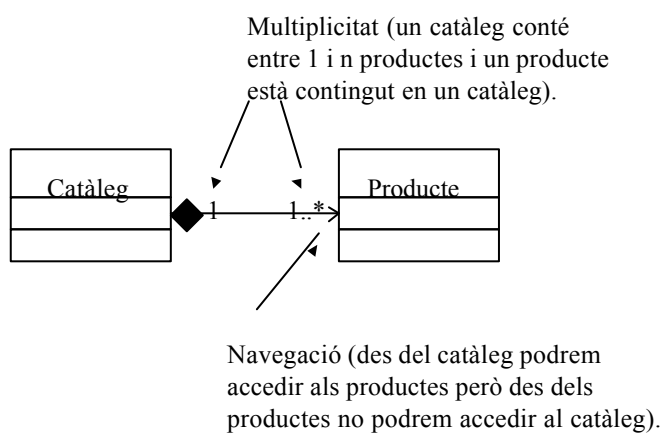
Una forma d'agregació amb un propietari fort i temps de vida coincidents. Les parts no poden sobreviure al tot (agregació).



Associació: Multiplicitat i Navegació

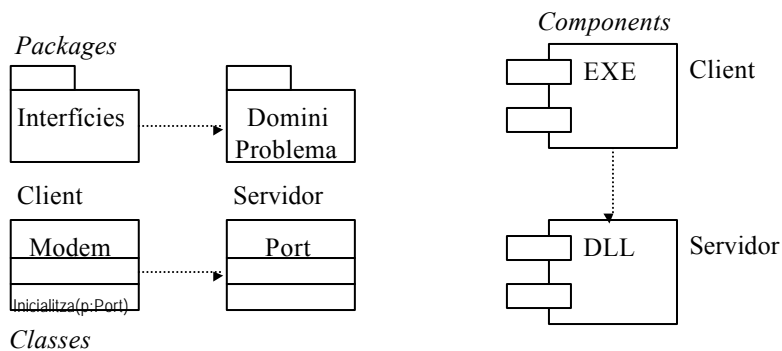
- **Multiplicitat.** La multiplicitat defineix quants objectes participen en una relació.
 - El número d'instàncies d'una classe relacionades amb UNA instància de l'altra classe.
 - S'especifiquen amb un rang en cada final de l'associació.
- **Navegació.** Les associacions i agregacions són bidireccionals per defecte, però de vegades és desitjable restringir la navegació a una direcció.
 - Si la navegació és restringida, s'afegeix una línia amb fletxa per indicar la direcció de la navegació, això vol dir, des de quin objecte es pot accedir a l'altre.

Associació: Multiplicitat i Navegació. Exemple



Relacions: Dependència

Una relació entre dos elements del model on un és client i l'altre servidor. Un canvi en l'especificació d'un element pot afectar l'altre. Un objecte és una variable local o paràmetre d'un mètode de l'altre.

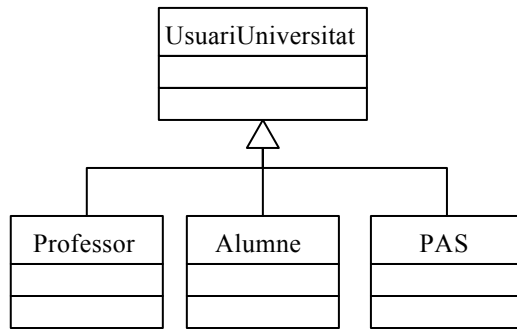


Relacions: Generalització

- Representa la relació “*ser un tipus de*”. És una relació entre classes on la classe especialitzada comparteix l'estructura i/o el comportament de la classe més general.
- Defineix una jerarquia d'abstraccions on la sub-classe **hereta** les propietats d'una o més super-classes:
 - Herència simple.
 - Herència múltiple.

4.4

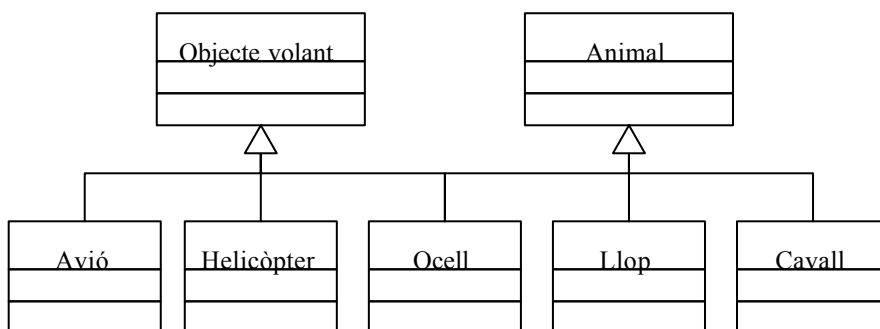
Exemple: Herència simple



45

4.4

Exemple: Herència múltiple



46