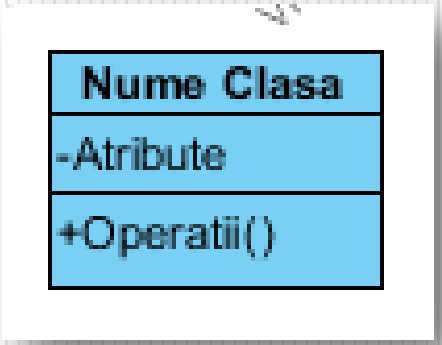
Seminar 4

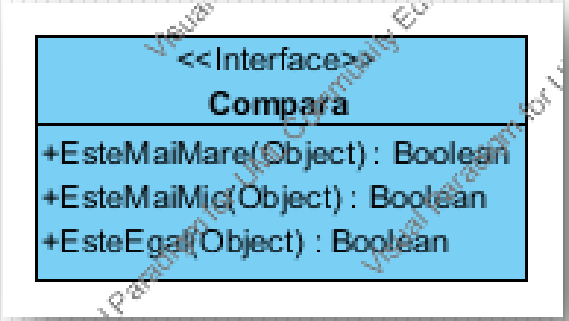
Realizarea sistemelor informatice pentru management

Diagrama claselor UML

Definirea unei clase

- Ansamblu de obiecte care au aceleaşi caracteristici şi contrângeri.
- Caracteristicile unei clase sunt atributele şi operaţiile.
- Clasele *abstracte* nu pot fi instanţiate. Rolul lor este de a permite altor clase să le moştenească, în vederea reutilizării caracteristicilor.
- O interfaţă descrie un set de caracteristici şi obligaţii publice. Specifică, de fapt, un contract. Orice instanţă care implementează interfaţa trebuie să ofere serviciile furnizate prin contract.





Exemple de clase stereotipe uzuale

- entitate (<<entity>>) – o clasă pasivă, care nu iniţiază interacţiuni;
- control (<<control>>) – iniţiază interacţiuni, conţine o componentă tranzacţională şi este separator între entităţi şi limite;
- limită (<<boundary>>) – este aflată la periferia sistemului, dar în interiorul său. Reprezintă limita de legătură cu actorul sau cu alte sisteme informatice;
- enumerare (<<enumeration>>) - este folosită pentru definirea tipurilor de date ale căror valori sunt enumerate.
- primitivă (<<primitive>>) - o formă de clasă care reprezintă tipuri de date predefinite, cum ar fi tipul Boolean.

Atribute -1

- Fiecare atribut este descris cel puţin prin numele său.
- Se pot adăuga şi informaţii adiţionale, iar forma generală a unui atribut este:

[vizibilitate][/]nume[:tip][multiplicitate][=valoare implicită] [{proprietate}]

- Vizibilitatea poate fi:
  - + public: poate fi văzută şi folosită de oricine
  - - private: numai clasa însăşi poate avea acces
  - # protected: au acces clasa şi subclasele acesteia
  - ~ package: numai clasele din acelaşi pachet pot avea acces
- / simbolizează un atribut derivat

Atribute -2

- UML permite specificare multiplicităţilor pentru atribute, atunci când dorim să definim mai mult de o valoare pentru un atribut. Au următoarea semnificaţie:

Multiplicitate	Sens
1	Exact 1 (implicit)
2	Exact 2
1..4	De la 1 la 4 (inclusiv)
3, 5	3 sau 5
1..*	Cel puţin unul sau mai mulţi
*	Nelimitat (inclusive 0)
0..1	0 sau 1

- Proprietate indică o proprietate suplimentară care se aplică atributului:
  - {readonly}: atributul poate fi citit, dar nu modificat
  - {ordered}, {unordered}: o mulţime ordonată sau neordonată
  - {unique}, {nonunique}: mulţimea poate conţine sau nu elemente identice

Operaţii

- Forma generală a unei operaţii este:

[vizibilitate] nume ([direcţie] lista parametri) [:tip returnat] [{proprietate}]

- Vizibilitatea – aceeaşi ca şi la clase
- Direcţie - 'in' | 'out' | 'inout' | 'return'
- Tip returnat – dacă operaţia returnează ceva, adică este o funcţie
- Un exemplu de proprietate a unei operaţii: {query} - nu are efecte secundare, nu schimbă starea unui obiect sau a altor obiecte, exemplu operaţiile de tip "get".

Exemple de atribute:

- - varsta: Integer {varsta>18}
- # nume:String[1..2]="Ioana"
- ~ Id:String {unique}
- / sumaTotala:Real=0

Exemple de operaţii:

- + setVarsta (out varsta: Integer)
- + getVarsta(in Id:String): Integer {query}
- - schimbaNume(inout nume:String)

Constrângeri

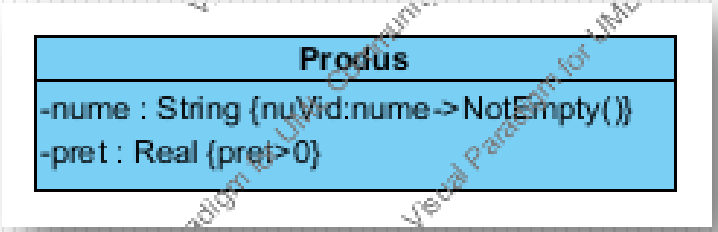
- O constrângere este o expresie care restricţionează un anumit element al diagramei de clase.
- Aceasta poate fi o expresie formală (scrisă în Object Constraint Language - OCL) sau o formulare semi-formală sau informală.
- Acestea sunt reprezentate între acolade.
- Pot fi scrise imediat după definirea elementului sau ca un comentariu.
- O constrângere poate avea şi un nume, astfel:
- **{nume : expresie booleană }**

Exemple de constrângeri OCL:

context Organizatie

inv: self. departamente→isUnique (nume)

inv: departamante.angajati→isUnique (cod)



Relaţii între clase -1

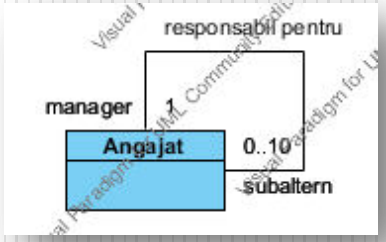
1. Relaţia de asociere implică stabilirea unei relaţii între clase.

- Este caracterizată prin:
  - denumire (opţională)
  - multiplicităţi – se trec la cele 2 capete ale asocierii;
  - roluri ale asocierii: se trec la fiecare capăt al asocierii şi conţin o descriere scurtă şi reprezentativă de (1 – 2 substantive)
  - direcţie de navigare
  - tipuri de asocieri:
    - unare
    - binare
    - ternare

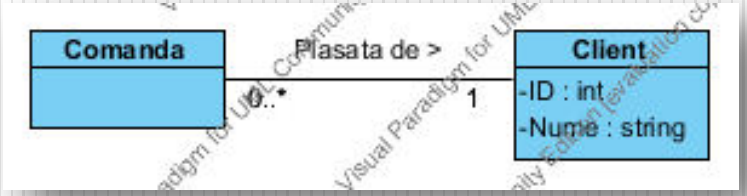
Relaţii între clase -2

Tipuri de asocieri:

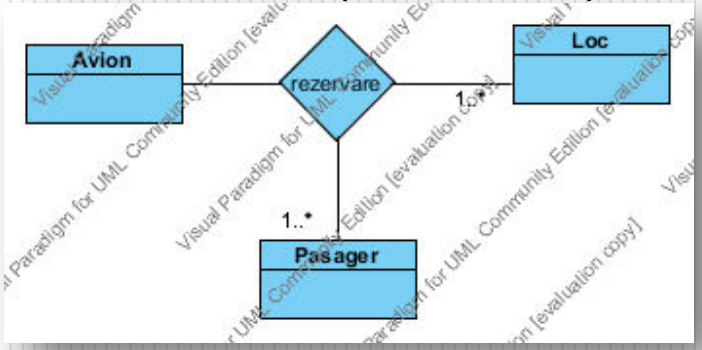
- Unare: conectează o clasă cu sine însăşi.



- Binare: se realizează între două clase.

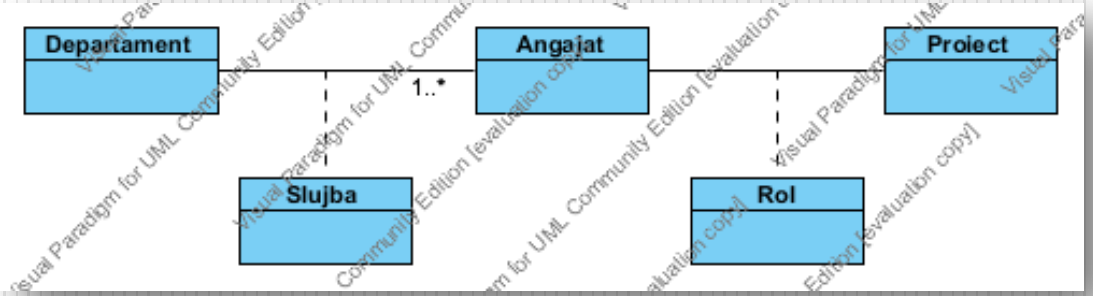


- Ternare: sunt transformate, de obicei, în asocieri binare.



Relații între clase -3

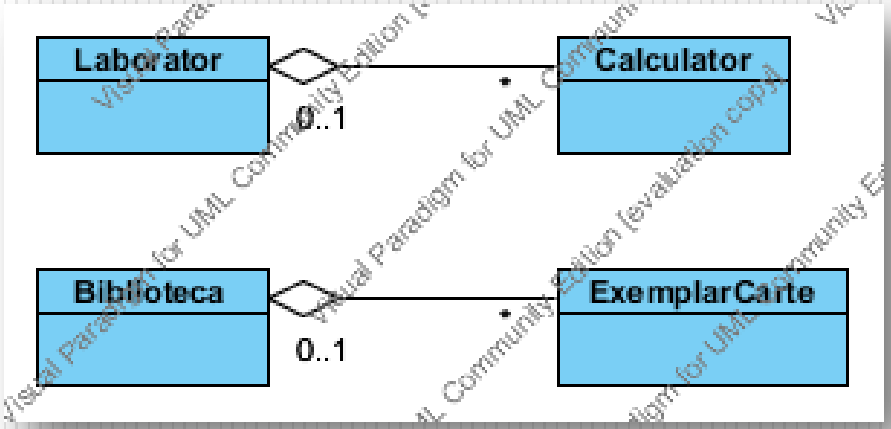
- Asocierea modelată ca o clasă permite relaţiei de asociere să aibă artibute şi operaţii.



2. *Relaţia de agregare* este o formă de asociere binară reprezentând o relaţie de tip parte/întreg.
- Poate fi de doua tipuri:
  - Agregare partajată (agregare)
  - Agregare compusă (compunere)

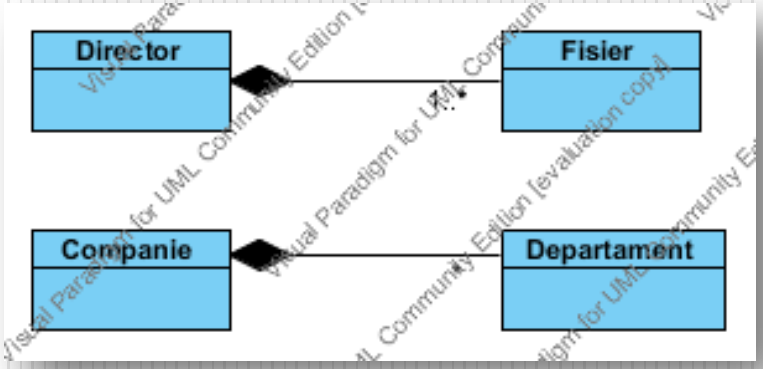
Relaţii între clase -4

- *Agregarea partajată* este o formă slabă de agregare în care instanţele părţilor sunt independente de întreg, astfel:
  - Aceleaşi părţi partajate pot fi incluse în mai multe clase întreg.
  - Dacă clasa întreg se şterge, clasele parte vor exista în continuare.
  - Se reprezintă sub forma unui romb gol plasat la capătul clasei întreg.



Relaţii între clase -5

- *Agregarea compusă* este o formă puternică de agregare în care instanţele părţilor sunt independente de întreg, astfel:
  - Dacă clasa întreg se şterge, clasele parte vor vor fi şterse şi ele.
  - Se reprezintă sub forma unui romb plin plasat la capătul clasei întreg.
  - Atunci când se foloseşte pentru modelarea obiectelor dintr-un anumit domeniu, ştergerea poate fi interpretată la figurativ, ca “terminare”, şi nu ca o distrugere fizică.



Relaţii între clase -6

Asociere

Obiectele ştiu unele de existaţa celorlalte şi pot lucra împreună.

Agregare

1. Protejează integritatea configuraţiei.
2. Funcţionează ca un tot unitar.
3. Control prin intermediul unui singur obiect.

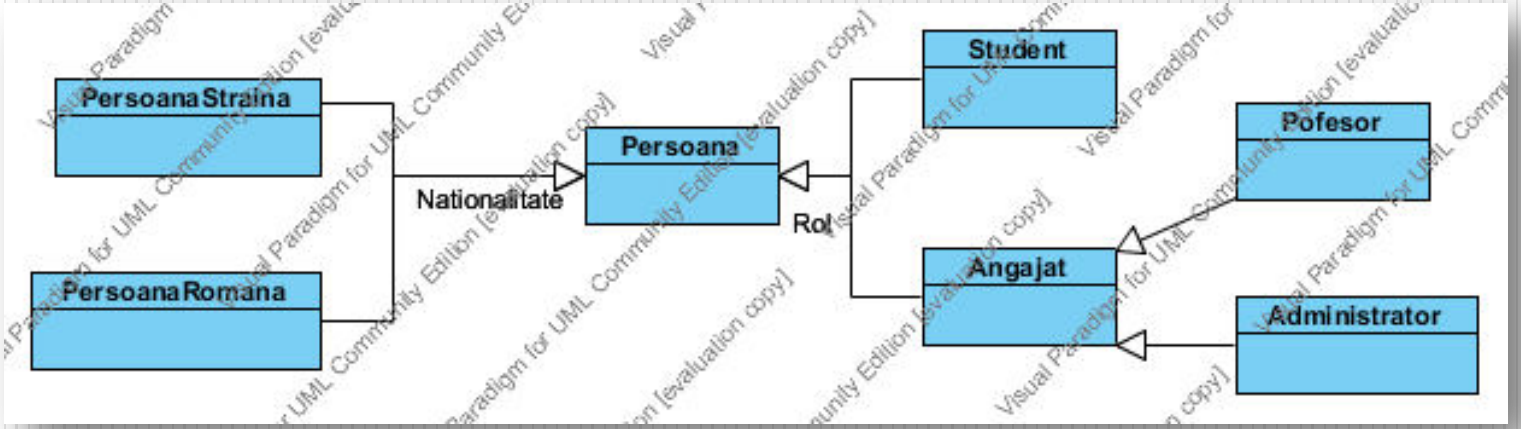
Compunere

Fiecare parte poate fi membră a unui singur obiect agregat.

Relaţii între asociere, agregare şi compunere

Relaţii între clase -7

3. *Relaţia de generalizare* este folosită pentru a indica moştenirea dintre o clasă generală (superclasă) şi o clasă specifică (subclasă).
- Se mai numeşte informal şi relaţie de genul “este un tip de”.
- Se reprezintă sub forma unui tringhi gol plasat la capătul superclasei.
- Subclasele moştenesc caracteristicile şi constrângerile superclasei.
- Este permisă moştenirea multiplă.



Relaţii între clase -8

4. Relaţia de dependenţă este folosită pentru a arăta o gamă largă de dependenţe între elementele unui model.

- În atapa de analiză, tipul de dependenţă poate să nu fie specificat.
- În proiectare, dependenţele vor fi personalizate cu stereotipuri sau vor fi înlocuite cu conectori specifici tehnologiei folosite.
- Se reprezintă sub forma unei linii punctate de la clasa dependentă “client,” pană la clasa “furnizor”, cu o săgeată la capătul clasei “furnizor”.
- În diagramele de clase, cele mai importante dependenţe sunt relaţiile de **utilizare** şi de **abstractizare**.

Relaţii între clase -9

- Dependenţa de utilizare (<<use>>, <<create>>, <<call>> etc.) este o relaţie în care clasa client are nevoie de altă clasă sau set de clase (furnizor) pentru a funcţiona.
- Dependenţa de abstractizare pune în relaţie două elemente sau seturi de elemente (numite client şi furnizor), reprezentând acelaţi concept, dar la niveluri diferite de abstractizare sau din puncte de vedere diferite.
  - Relaţia de **realizare** este o formă de abstractizare, în care un element de modelare (furnizorul) reprezintă specificaţia, iar celălat element (clientul) reprezintă implementarea specificaţiei. Se reprezintă sub forma unei linii punctate cu o săgeată la capătul clasei furnizor.

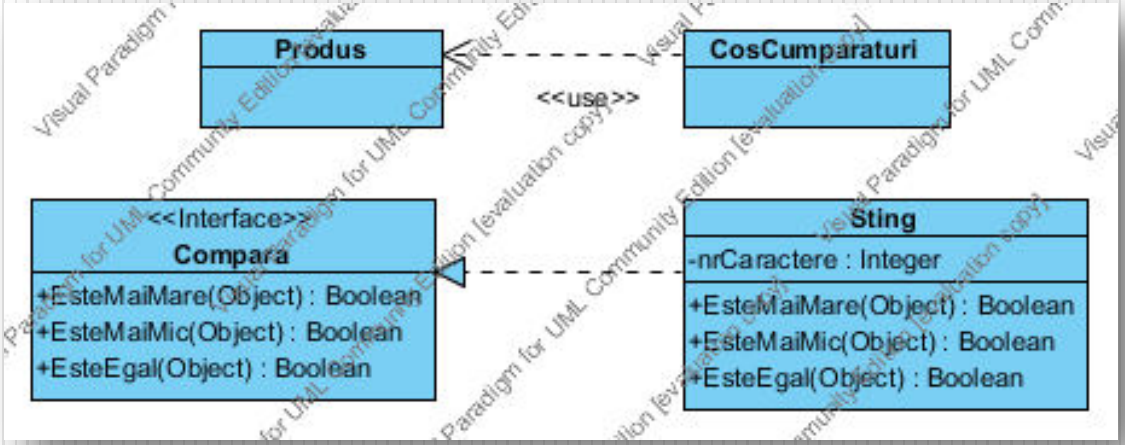


Diagrama de clase UML

- Ce este şi cum se reprezintă multiplicitatea?
- Denumiţi domeniile de vizibilitate în UML.
- Care este rolul moştenirii şi cum se identifică?
- Clasele abstracte pot fi instanţiate?
- De ce sunt importante numele de rol?
- Care este diferenţa dintre agregare şi compunere?

Lucru la seminar

Să se întocmească diagrama de clase pentru scenariul de mai jos.

Scopul proiectului este realizarea aplicaţiei informatice pentru gestiunea activităţii unei unităţi hoteliere. În vederea cazării, un client poate solicita rezervarea uneia sau mai multor camere prin e-mail sau telefonic. Pentru aceasta furnizează recepţionerului informaţii privind perioada de cazare şi tipurile de camere solicitate. Clienţii vor beneficia de reduceri dacă rezervă cel puţin 3 camere sau dacă perioada de cazare depăşeşte 5 zile. Recepţionerul verifică disponibilitatea camerelor şi îl înştiinţează pe client de acest lucru precum şi de costul estimat al cazării. Dacă nu există camere disponibile conform solicitării, recepţionerul poate oferi clientului alternative. De asemenea, clientul poate solicita un discount (suplimentar sau nu), iar recepţionerul va decide fezabilitatea discountului, fiind asistat obligatoriu de managerul hotelului. În situaţia în care clientul este de acord cu preţul propus, se va proceda la realizarea rezervării. Pentru clienţii noi, recepţionerul solicită datele de identificare, pe care le introduce în aplicaţie.

Odată ajuns la hotel, şi dacă a făcut în prealabil o rezervare, clientul va furniza datele de identificare ale sale şi/sau ale rezervării şi se face cazarea. Dacă nu există o rezervare, se va verifica disponibilitatea camerelor pentru perioada cerută. Atunci când se găseşte o astfel de cameră, se face cazarea. La finalul sejurului, recepţionerul întocmeşte o listă cu toate serviciile solicitate de client şi preţul acestora. Lista trebuie validată de client, după care se întocmeşte factura finală. Factura poate fi plătită parţial sau integral, prin transfer bancar, numerar sau folosind un card bancar. Totodată, înainte de a părăsi hotelul, clientul este rugat să completeze un formular prin care să evalueze serviciile oferite de unitatea hotelieră.