Analiza orientată obiect a sistemelor informatice

În etapa de analiză a sistemului sunt analizate ***specificațiile*** și ***cazurile de utilizare***, identificându-se cele mai importante concepte cu care lucrează sistemul, împreună cu relațiile dintre acestea. Se reprezintă grafic printr-o diagramă structura domeniului claselor pentru sistemul analizat.

1. Se inițiază reprezentarea *diagramei de clase*, care va fi finisată și în etapele următoare. Diagrama claselor reprezintă grafic ***structura statică*** a sistemului, prin includerea claselor identificate, a pachetelor și a relațiilor dintre acestea. **Un pachet** poate conține *clase, interfețe, componente, noduri, colaborări, cazuri de utilizare, diagrame sau alte pachete.*
2. Se inițiază construirea *diagramei obiectuale* care modelează instanțele elementelor conținute în diagramele de clase. Aceste diagrame cuprind un set de obiecte și relațiile dintre acestea la un anumit moment.

Analiza orientată obiect a sistemelor informatice

3. Pentru a evidenția stările prin care poate trece un obiect sau un eveniment (mesaje primite, erori, condiții de realizare) sunt reprezentate *diagramele de stare*, care reprezintă ***ciclul de viață al obiectelor***, subsistemelor și sistemelor. Stările obiectelor se schimbă la recepționarea evenimentelor sau semnalelor.
4. *Diagrama de activitate* este realizată cu scopul de a evidenția acțiunile și rezultatul acestor acțiuni și pentru a scoate în evidență fluxurile de lucru.
5. Pentru a evidenția interacțiunile dintre obiecte se construiesc *diagramele de interacțiune*: diagramele de secvență și, respectiv, diagramele de comunicare.
   - *Diagramele de secvență* descriu modul în care interacționează și comunică obiectele, prin focalizare pe mesajele care sunt transmise și recepționate.
   - *Diagramele de comunicare* permit reprezentarea atât a interacțiunilor, cât și a legăturilor dintre un set de obiecte care colaborează. Se utilizează când este utilă vizualizarea coordonatei spațiale.

Diagrama de clase

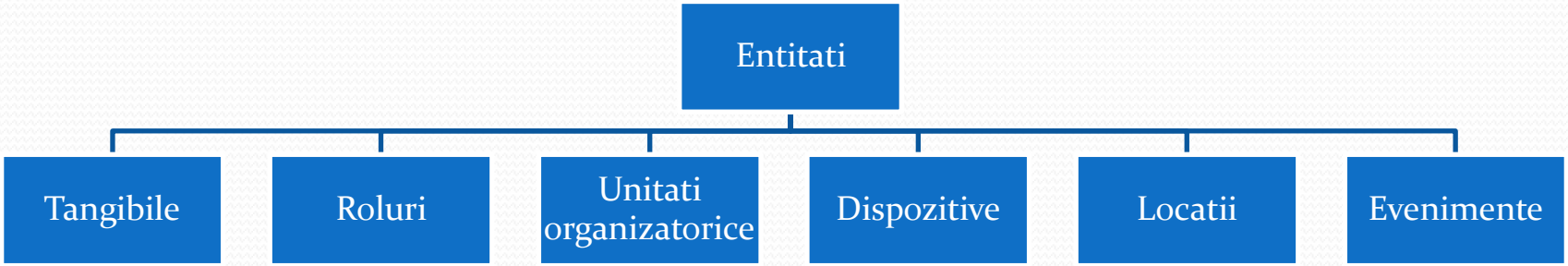
- Diagrama claselor este **cea mai importantă diagramă** în cadrul analizei şi proiectării orientate obiect. Scopul diagramei claselor este de a prezenta natura statică a claselor punand in evidenţă atributele, operaţiile şi asocierile.
- Majoritatea instrumentelor de modelare orientate obiect **generează codul sursă numai din diagrama claselor.**
- Celelalte diagrame UML furnizează diferite **puncte de vedere** din care să fie *identificate atributele, operaţiile şi asocierile* dintre clase. Ele ajută la **validarea diagramei claselor**, putând servi la clarificarea unei probleme specifice .

Tehnici de identificare a claselor

a) **Tehnica brainstorming** – se utilizeaza o lista de control cu toate **tipurile de entitati** care apar in mod frecvent si se realizeaza sedinte de brainstorming pentru a identifica clasele domeniului pentru fiecare dintre tipurile respective
b) **Tehnica substantivelor** – se identifica toate substantivele care apar in descrierea sistemului si sehotaraste daca substantivul este o clasa a domeniului, un atribut sau un element neimportant

a. Tehnica de brainstorming

- Avem vreun element de genul…



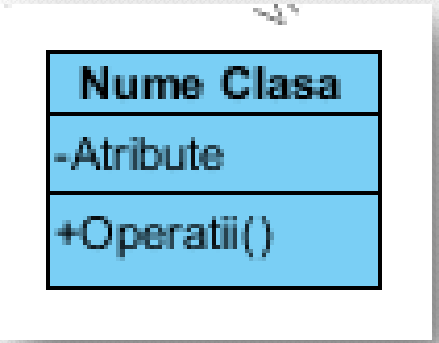
Tangibile	Roluri	Unitati organizatorice	Dispozitive	Locatii	Evenimente
Avion	Angajat	Divizie	Senzor	Depozit	Zbor
Carte	Doctor	Departament	Controler	Fabrica	Logon
Autovehicul	Pacient	Birou	Imprimanta	Magazin	Logoff
Documet	Client	Sectie	Timer	Sucursala	Comanda
Formular	Utilizator	Grup de lucru	Scaner	Desktop	Contract
	Administrator de sistem		Sortator		Achizitie
					Plata

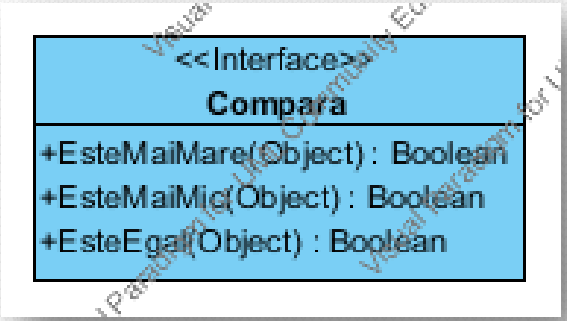
b. Tehnica substantivelor

1. Se identifica **toate substantivele** pe baza cazurilor de utilizare, actorilor si altor informatii despre sistem
2. Se **preiau informatiile** necesare din sistemele, procedurile, formularele si rapoartele existente
3. Se **rafineaza** lista de substantive
   - Apartine ariei de cuprindere a sistemului analizat?
   - Este nevoie sa pastram mai mult de un astfel de element?
   - Este un sinonim pentru un element deja identificat?
   - Este un atribut al altui element inregistrat?
   - Este doar un output al sistemului obtinut pe baza unui alt element deja identificat
   - Este doar un input care rezulta din inregistrarea alttui element identificat?
4. Se creaza o **lista principala de substantive** si se noteaza care trebuie inclus/exclus/ discutat
5. Se prezinta lista utilizatorilor, beneficiarilor, membrilor echipei spre **validare**

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

Operații

- 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

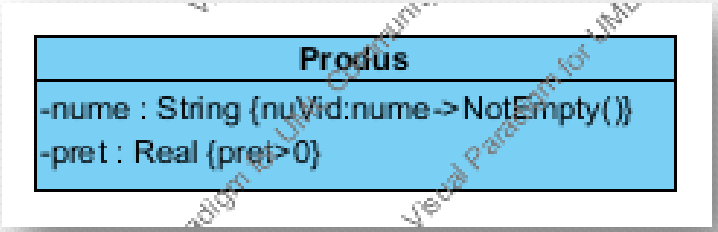
- O constrângere este o expresie care restricţionează un anumit element al diagramei de clase.
- Aceasta poate fi o expresie formală (scrisă în Object Constraint Language - OCL) sau o formulare semi-formală sau informală.
- Acestea sunt reprezentate între acolade.
- Pot fi scrise imediat după definirea 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)



Relații între clase -1

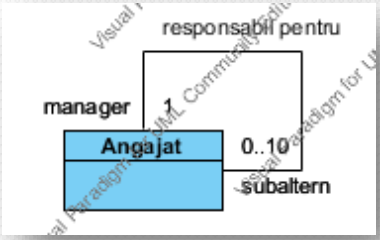
1. Relaţia de asociere implică stabilirea unei relaţii între clase.

- 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

Relații între clase -2

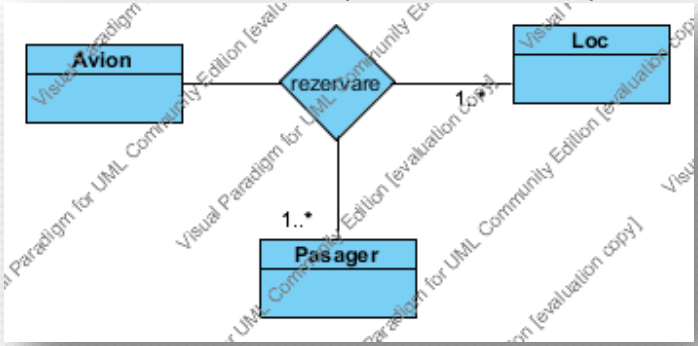
Tipuri de asocieri:

- Unare: conectează o clasă cu sine însăşi.



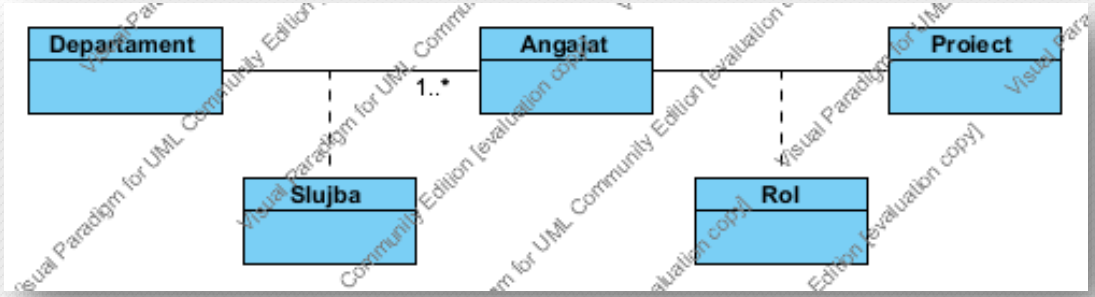
- Binare: se realizează între două clase.

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



Relații între clase -3

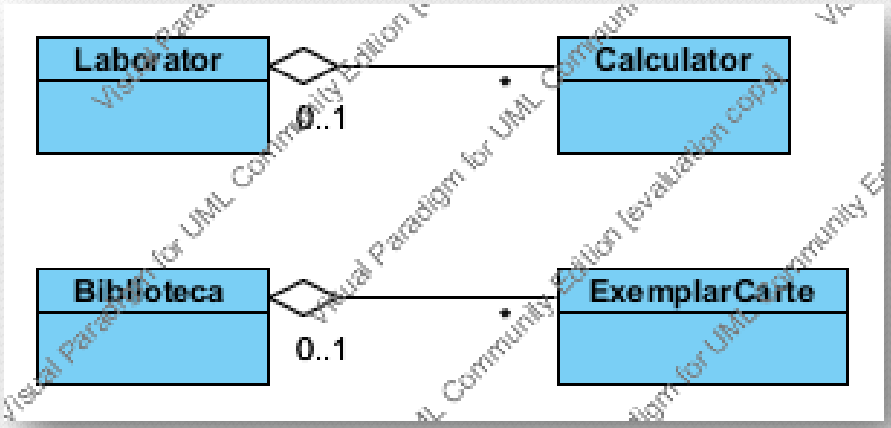
- Asocierea modelată ca o clasă permite relaţiei de asociere să aibă artibute şi operaţii.



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)

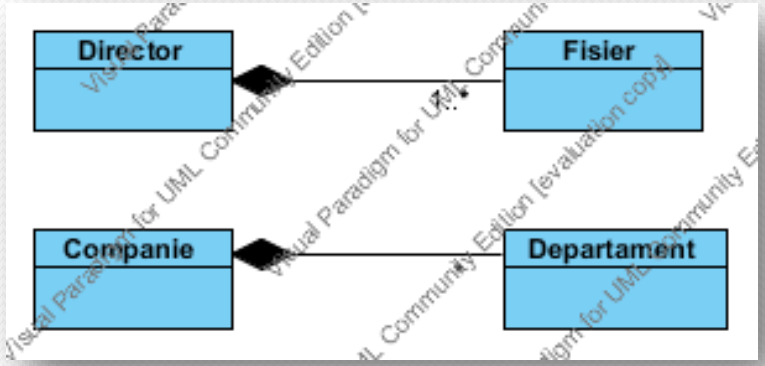
Relații între clase -4

- *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.



Relații 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.



Relații între clase -6

Asociere

Obiectele știu 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

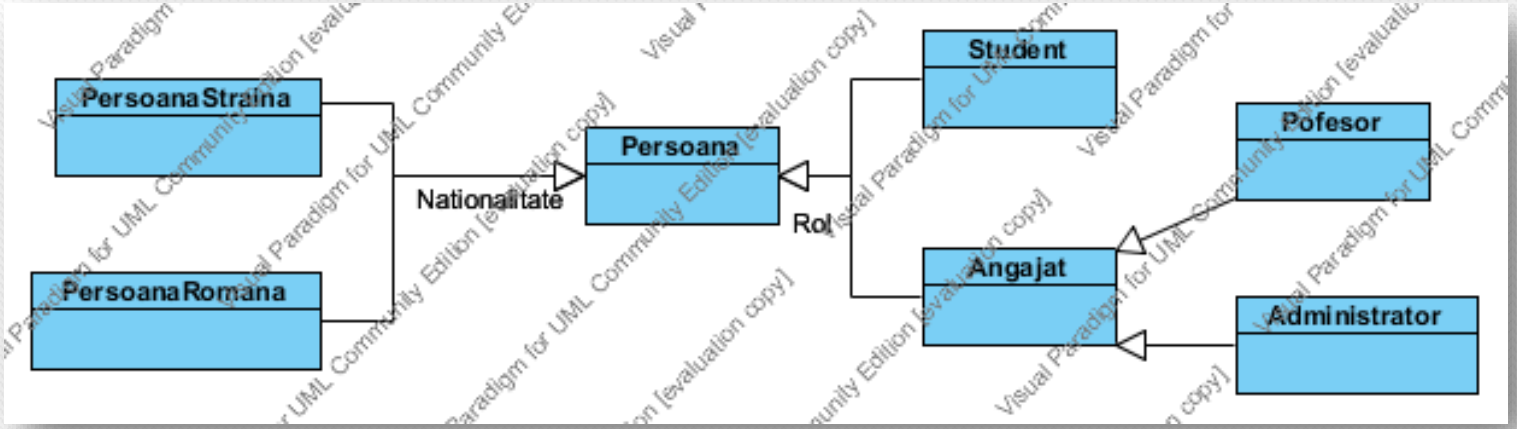
Fiecare parte poate fi membră a unui singur obiect agregat.

Relaţii între asociere, agregare şi compunere

Relații 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.



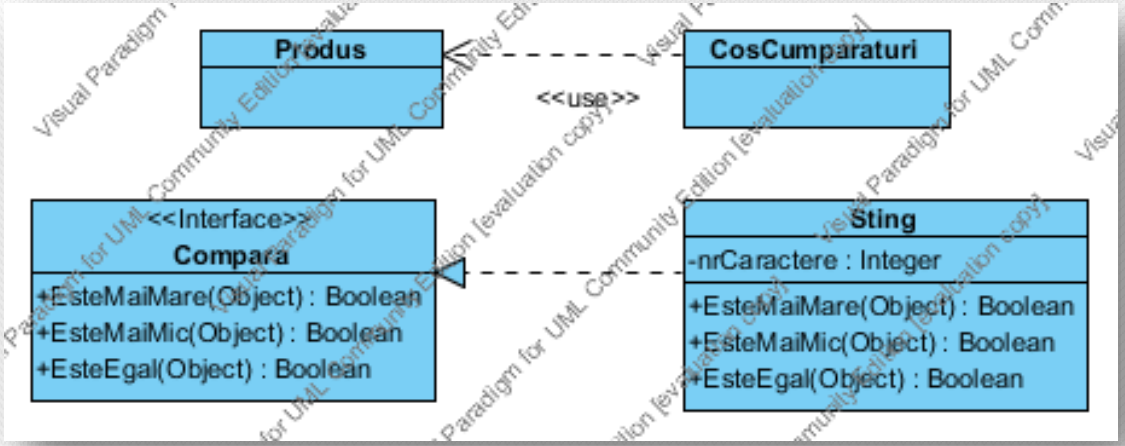
Relații între clase -8

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

- În etapa 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**.

Relații 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.



Exemplu de diagramă de clase

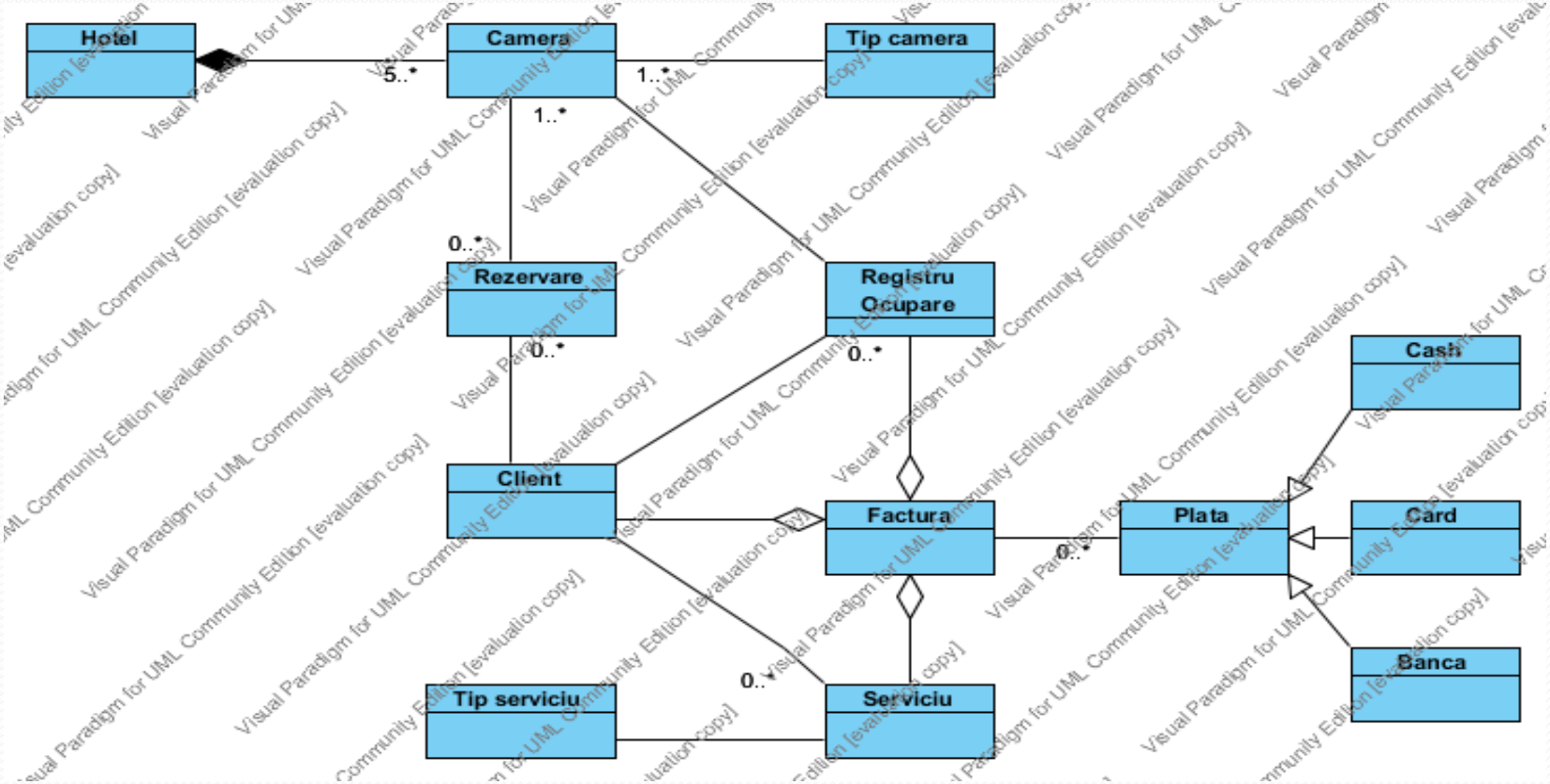
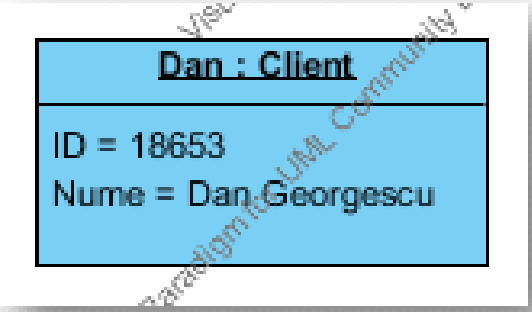


Diagrama de obiecte

- Constă din obiecte şi legăturile dintre acestea.
- Are rolul de a valida diagrama de clase.
- O legătură reprezintă o relaţie între două obiecte.

Diagrama de obiecte	Diagrama de clase
Modelează fapte despre anumite entităţi.	Modelează reguli pentru tipuri de entităţi.
Reprezintă obiecte reale.	Reprezintă abstractizări ale conceptelor.
Leagă între ele obiecte.	Asociază entităţi.

- Un obiect este denumit folosind numele acestuia, semnul ":" urmat de numele clasei căreia îi aparţine: *nume obiect : nume clasa .*
- Pot exista şi obiecte anonime, denumite doar prin numele clasei.

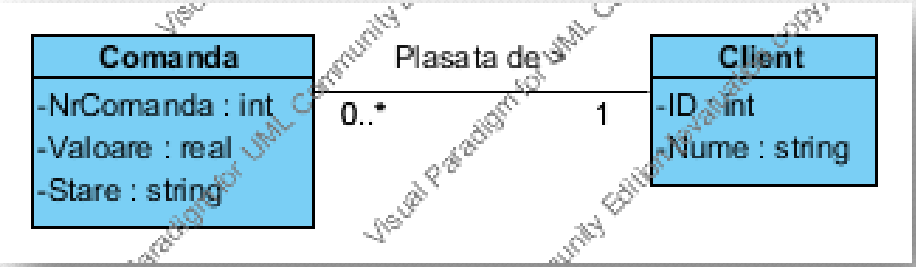


Notațiile diagramei de clase şi obiecte - comparație

Diagrama de clase	Diagrama de obiecte
Clasa are trei compartimente: nume, atribute şi operaţii.	Obiectul are numai două compatrimente: nume şi atribute.
Numele clasei este specificat singur în primul compartiment.	Formatul numelui unui obiect include şi numele clasei, toată expresia fiind subliniată. Aceste notaţii vor fi întâlnite şi în alte diagrame care reprezintă obiecte.
Al doilea compartiment descrie proprietăţi sub forma atributelor.	Al doilea compartiment defineşte valori pentru fiecare atribut, pentru testarea modelului.
Operaţiile apar în descrierea clasei.	Operaţiile nu sunt incluse în obiecte, deoarece ele sunt identice pentru fiecare obiect al clasei.
Clasele sunt conectate prin asocieri, având un nume, multiplicitate, constrângeri şi roluri. Clasele sunt o abstractizare a obiectelor, deci este necesar să specificăm câte clase participă într-o asociere.	Obiectele sunt conectate printr-o legătură, care poate avea un nume, roluri, dar nu şi multiplicităţi. Obiectele reprezintă entităţi singulare, toate legăturile sunt unu-la-unu, iar multiplicităţile sunt irelevante.

Diagrama de obiecte în Visual Paradigm

- Se definește diagrama de clase în care clasele au specificate atribute.



- Se definește un obiect în diagrama de obiecte (Instance Specification).
- Se selectează clasa căreia îi aparţine obiectul: Click dreapta pe obiect -> Select Classifier-> se bifează şi selectează clasa corespunzătoare
- Opțional, se dă un nume obiectului.
- Se definesc valorile pentru atribute: Click dreapta pe obiect -> Slots, Define Slots (pentru atributele cărora vrem să le dăm valori) ->Edit Values-> Add -> Text (se introduce valoarea dorită).
- Se creează legături (Link) între obiecte.

Diagrama de obiecte în Visual Paradigm - exemplu

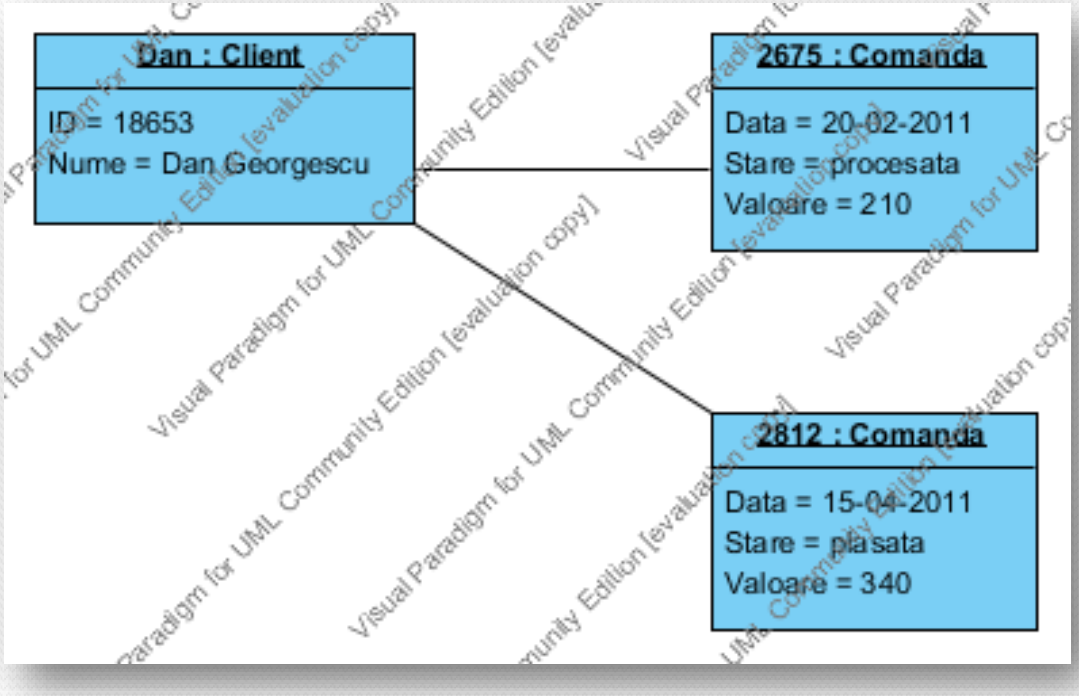
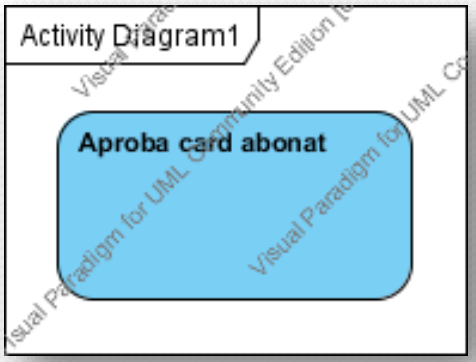


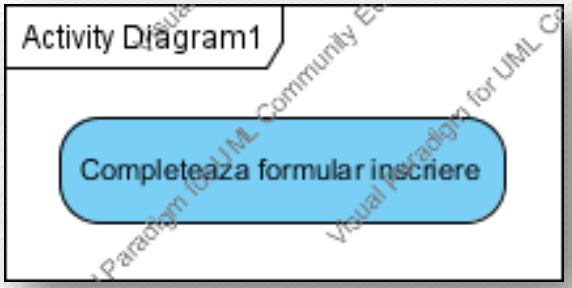
Diagrama de activitate

- Ajută la reprezentarea vizuală a **secvenţelor de acţiuni** prin care se doreşte obţinerea unui rezultat.
- Se poate realiza **pentru unul sau mai multe cazuri de utilizare** sau pentru descrierea unor **operaţii complexe**.
- Nu se construieşte pentru fiecare caz de utilizare şi scenariu, deoarece nu este necesar, ci numai pentru cele importante.
- Descrie fluxul de lucru dintr-un punct de plecare până într-un punct de terminare, detaliind **căile de decizie** care pot apărea într-o activitate.
- Poate fi folosită pentru a descrie **procesare paralelă**.
- Este importantă în modelarea proceselor de afaceri.

Activitate/ Acțiune

- **Activitatea** - un comportament parametrizat reprezentat sub forma unui flux coordonat de acţiuni.
- **Acţiunea** – reprezintă un singur pas în cadrul unei activităţi.
  - Acţiunea poate fi **fizică**, realizată de un factor uman sau **electronică**.
- Activitate/acţiune, reprezentată printr-un dreptunghi cu margini rotunjite.



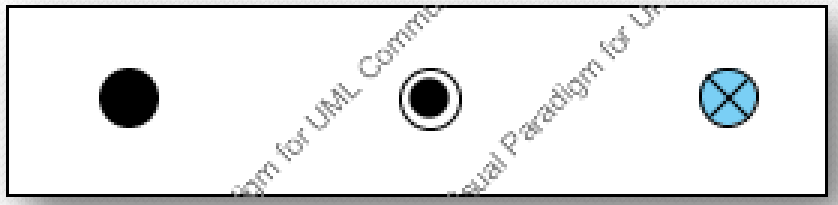


Constrângeri

- Constrângerile pot fi ataşate unei acţiuni, spre exemplu, sub forma unor **pre- şi post-condiţii**.
- Se folosesc cuvintele cheie **<<precondition>>** şi **<<postcondition>>.**

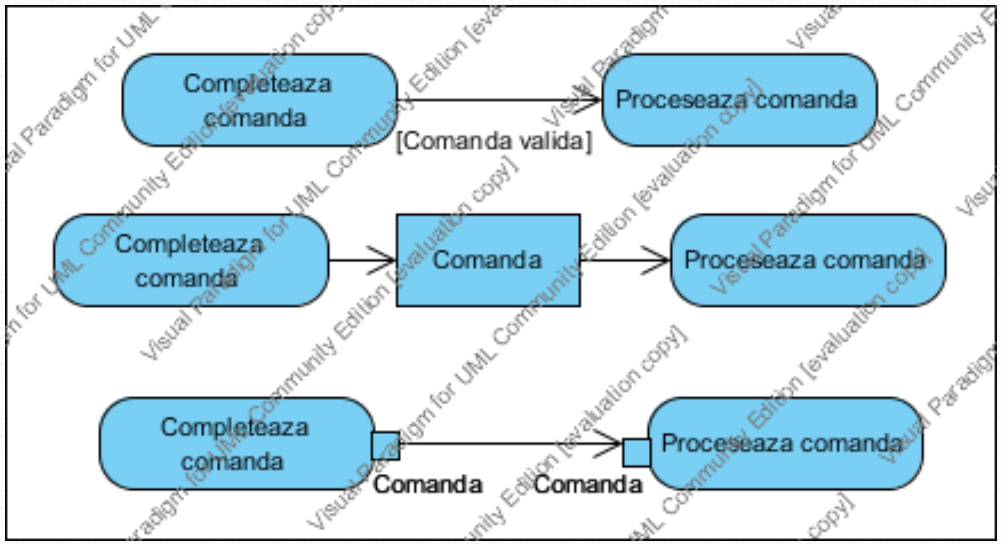
Noduri

- **Nod iniţial** - reprezintă punctul de început al diagramei.
- **Nodul final** - există două tipuri de noduri finale:
  - ***Nod final al activităţii***: reprezintă sfârşitul tuturor fluxurilor de control dintr-o diagramă.
  - ***Nod final al fluxului***: arată că procesul se opreşte în acel punct. Acesta denotă sfârşitul unui singur flux de control.



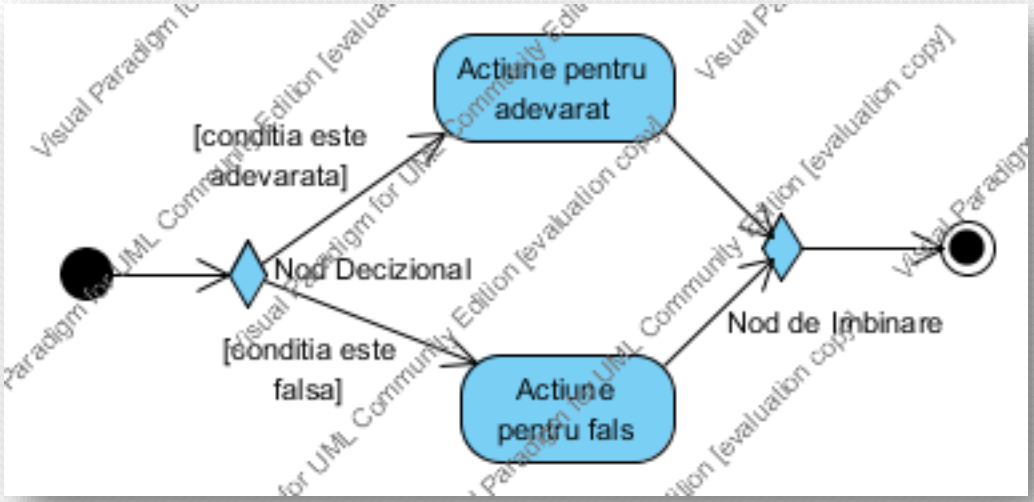
Fluxuri şi obiecte

- **Flux de control** – este un arc pe diagramă care descrie modul de transfer al controlului de la o acţiune la alta.
- **Flux de obiecte** - este un flux de-a lungul căruia sunt transferate obiecte sau date.
  - Trebuie să aibă un obiect la cel puţin unul din capete.
  - Există şi o notaţie prescurtată în care se pot folosi **calificatori** (engl. pins) de intrare şi de ieşire.
- **Condiţie tranzitorie** - un text pe un flux ce defineşte **o condiţie** care trebuie să fie adevărată pentru a produce tranziţia către următoarea acţiune.



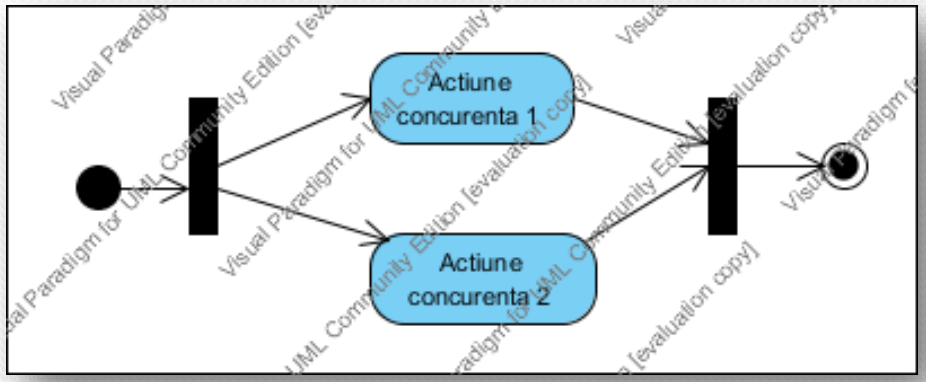
Noduri decizionale şi de îmbinare

- Ambele se reprezintă sub forma unui romb şi pot fi denumite.
- **Nod decizional** (decision):
  - nod în care intră un flux şi ies mai multe.
  - fluxurile de ieşire trebuie să fie însoţite de ***condiţii mutual exclusive.***
- **Nod de imbinare** (merge):
  - nod în care intră mai multe fluxuri şi iese unul singur.

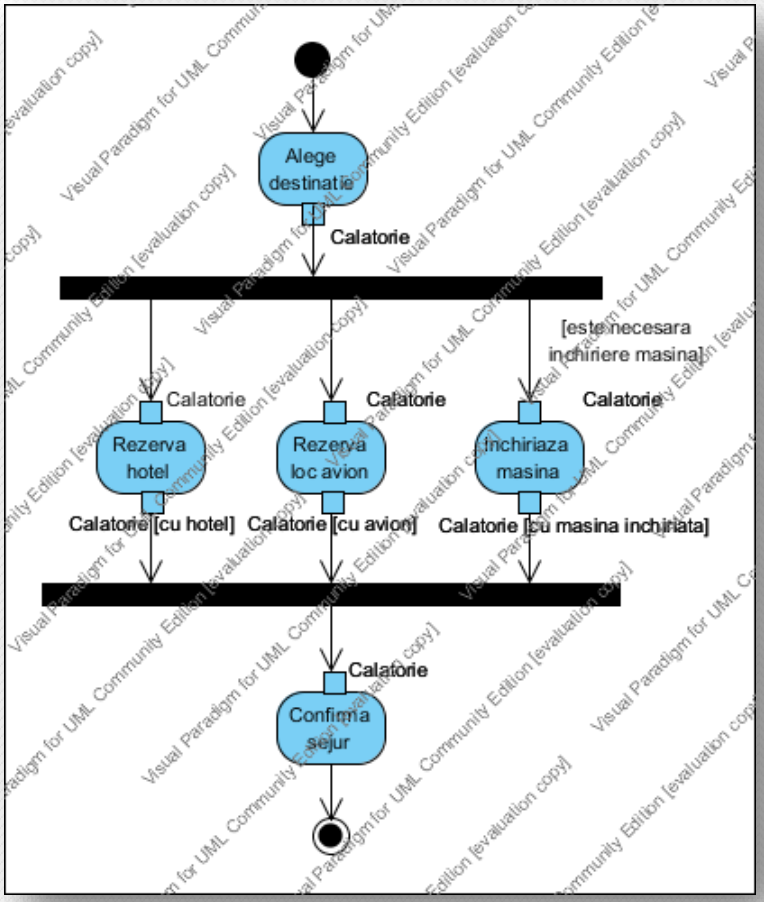


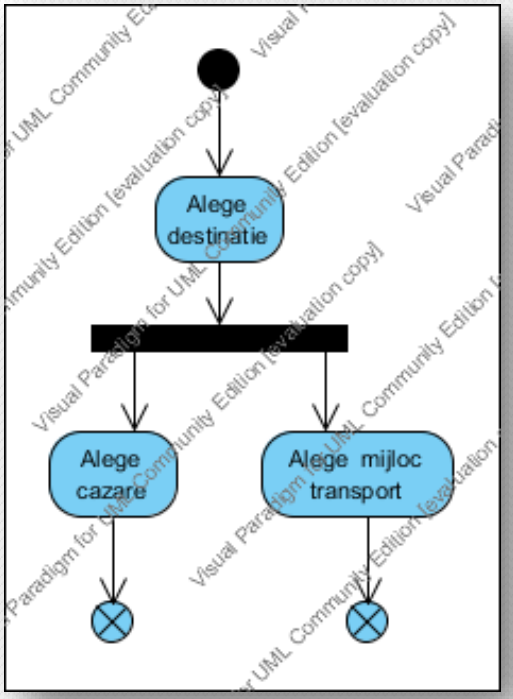
Noduri de bifurcație și joncțiune

- Ambele se reprezintă printr-o linie neagră îngroşată.
- **Nod de bifurcaţie** (fork):
  - nod în care intră unul singur flux şi ies mai multe.
  - denotă **începutul** unor **acţiuni paralele**.
- **Nod de joncţine** (join):
  - nod în care intră mai multe fluxuri şi iese doar unul singur.
  - toate fluxurile care intră în joncţiune trebuie să ajungă în punctul de joncţiune înainte ca procesarea să continue.
  - denotă **sfârşitul** unei **procesări paralele.**



Noduri - exemple





Un nod de joncţiune este diferit de un nod de îmbinare deoarece sincronizează două fluxuri de intrare şi produce un singur flux de ieşire. Un un nod de îmbinare transmite mai departe orice flux de control ajunge la el.

Partiții

- Sunt **culoare** care arată **cine sau ce execută acţiunile** într-o diagramă de activitate.
- Pot fi **orizontale** sau **verticale**.
- Separarea pe partiţii poate fi făcută în funcţie de *unitaţile organizaţionale*, *responsabilităţi* etc.