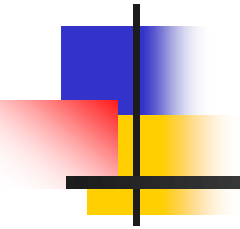
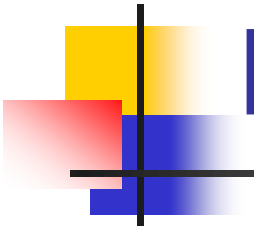


UTM
Software Engineering

Seminar UML



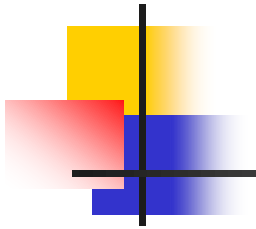
Resources

- https://plantuml.com/ - online ???
- https://www.umlet.com/ - de verificat de catre studenti
- https://medevel.com/open-source-uml-tools/ - de verificat de catre student

- *Object-Oriented and Classical Software Engineering,* Sixth Edition, WCB/McGraw-Hill, 2005 Stephen R. Schach

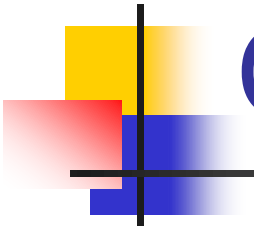
- UML resource page http://www.uml.org/

To send your project



Email to: marius.rogobete@yahoo.com

Subject: [IS - Zi_Modelio key] <*nume, grupa*>



Outline

- Ce este UML și de ce folosim UML?
- Cum să utilizați diagramele UML pentru a proiecta un sistem software?
- Ce instrumente de modelare UML folosim astăzi?

Ce este UML și de ce il folosim

- UML → “Unified Modeling Language”
- Limbajul: exprima o idee, nu o metodologie
- Modelare: Descrierea unui sistem software la un nivel ridicat de abstractizare
- Unificat: UML a devenit un standard mondial

  Object Management Group (OMG): www.omg.org

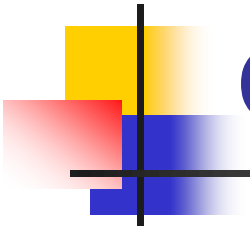
Ce este UML și de ce il folosim

- Mai multe despre UML:

  - Este un limbaj grafic standardizat în industrie pentru specificarea, vizualizarea, construirea și documentarea artefactelor sistemelor software

  - UML folosește în principal notații grafice pentru a exprima analiza OO și proiectarea proiectelor software.

  - Simplifica procesul complex de proiectare software

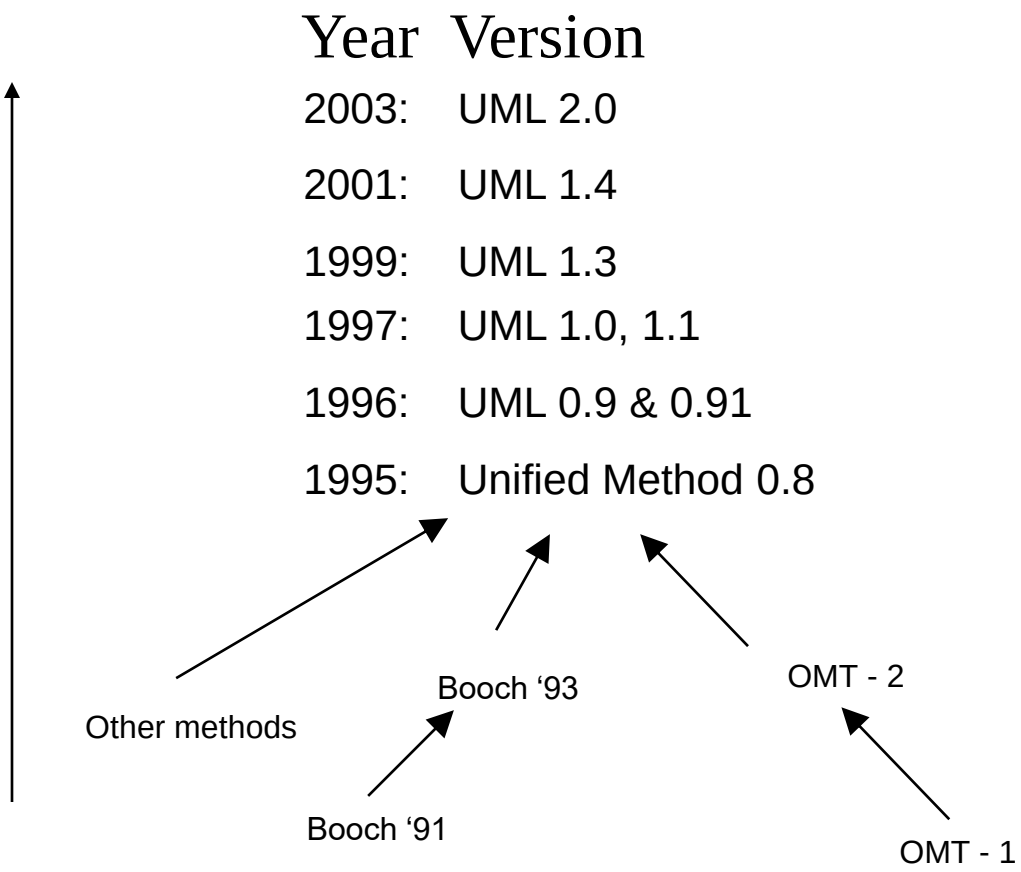


Ce este UML și de ce il folosim

- De ce folosim UML?

  - Utilizeaza notația grafică: mai clar decât limbajul natural (imprecis) și codul (prea detaliat).
  - Ajută la obținerea unei imagini de ansamblu asupra unui sistem.
  - UML nu depinde de niciun limbaj sau tehnologie.
  - UML ne trece de la fragmentare la standardizare.

Ce este UML și de ce il folosim



Cum să utilizați diagramele UML pentru a proiecta un sistem software

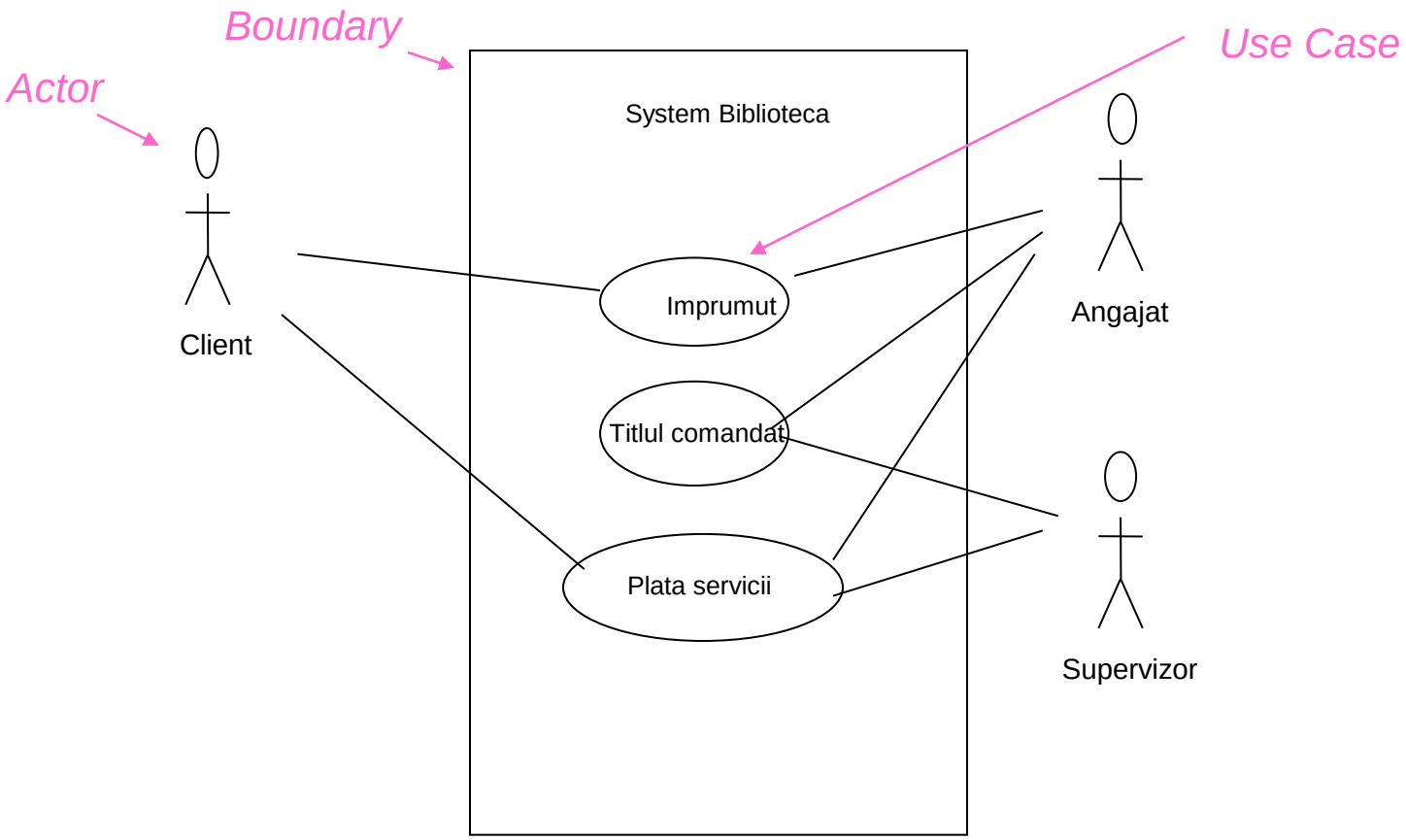
- Tipuri de diagrame UML:
  - Diagrama de caz de utilizare
  - Diagrama de clasă
  - Diagrama de secvența
  - Diagrama de colaborare
  - Diagrama de stare

Acesta este doar un subset de diagrame... dar sunt cele mai utilizate pe scară largă

Diagrame de tip Use-Case

- O diagramă de cazuri de utilizare este un set de cazuri de utilizare
- Un caz de utilizare este un model al interacțiunii dintre:
  - Utilizatori externi ai unui produs software (actori) și
  - Produsul software în sine
  - Mai exact, un actor este un utilizator care joacă un anumit rol
- descrierea unui set de scenarii utilizator
- captarea cerințelor utilizatorilor
- contact între utilizatorul final și dezvoltatorii de software

Diagrame de tip Use-Case

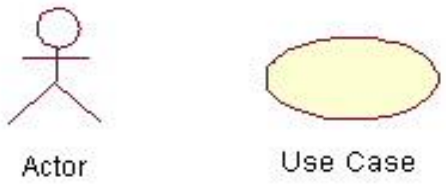


Diagrame de tip Use-Case

- **Actori:** un rol pe care un utilizator îl joacă în ceea ce privește sistemul, inclusiv utilizatorii umani și alte sisteme. de exemplu, obiecte fizice neînsuflețite (de exemplu, robot); un sistem extern care are nevoie de unele informații din sistemul actual.

- **Caz de utilizare:** un set de scenarii care descriu o interacțiune între un utilizator și un sistem, inclusiv alternative.

- **Limita sistemului:** diagramă dreptunghiulară reprezentând granița dintre actori și sistem.



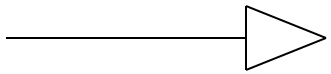
Diagrame de tip Use-Case

- **Asociatie:**

comunicarea dintre un actor și un caz de utilizare; Reprezentat printr-o linie continuă.

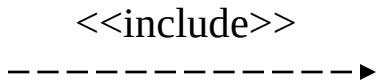
- **Generalizare:**

relație între un caz de utilizare general și un caz de utilizare special (utilizat pentru definirea alternativelor speciale) Reprezentată printr-o linie cu un vârf de săgeată triunghiular către cazul de utilizare părinte.

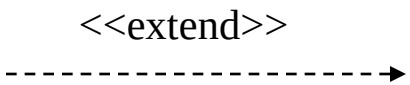


Diagrame de tip Use-Case

Include: o linie punctată etichetată <<include>> care începe cu cazul de utilizare de bază și se termină cu săgeți care indică cazul de utilizare include. Relația de includere apare atunci când o bucată de comportament este similară în mai multe cazuri de utilizare. Folosiți „include” în loc să copiați descrierea acelui comportament.



Extend: o linie punctată etichetată <<extend>> cu o săgeată spre cazul de bază. Extinderea cazului de utilizare poate adăuga comportament la cazul de bază de utilizare. Clasa de bază declară „puncte de extensie”.



Diagrame de tip Use-Case

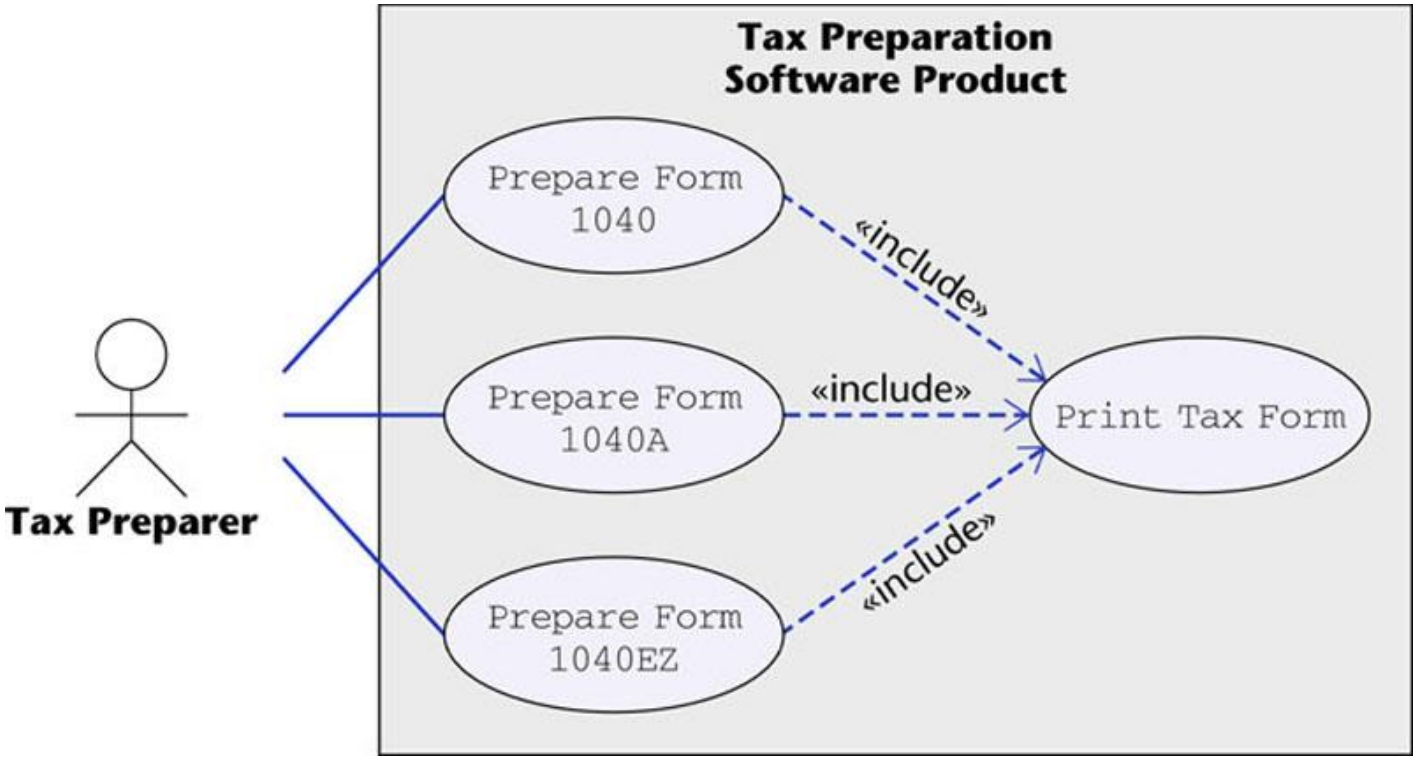
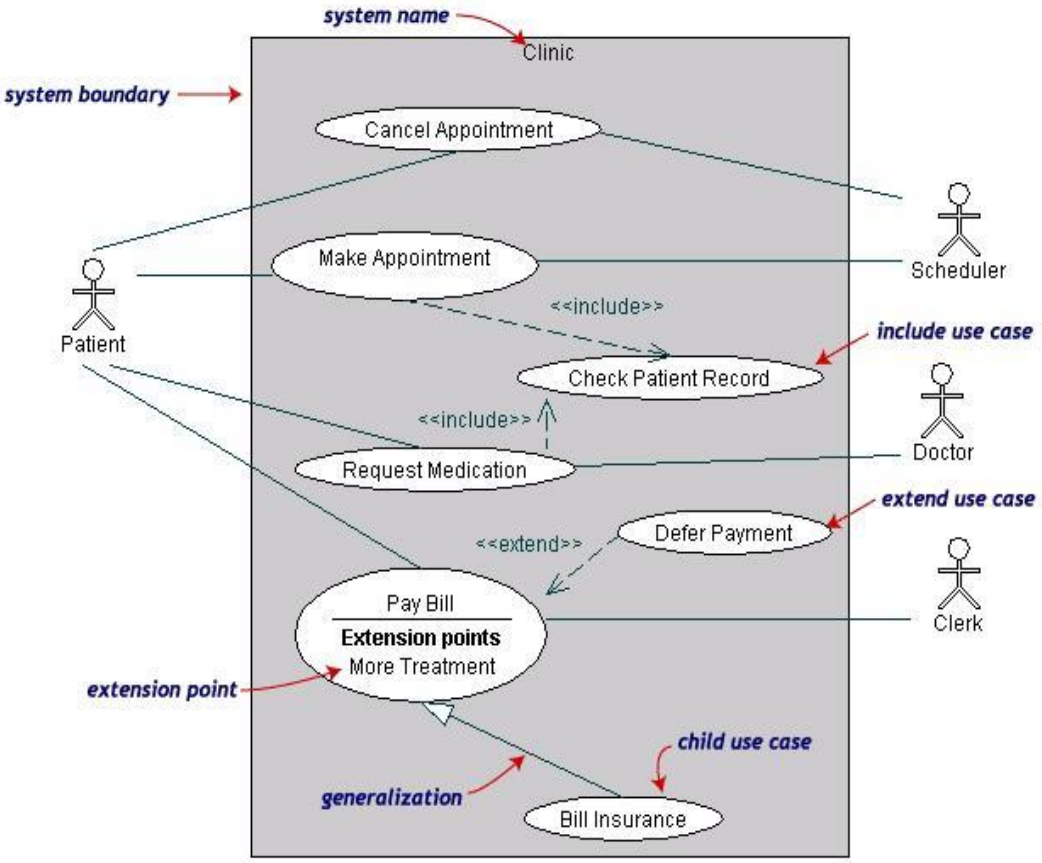


Figure 16.12

The McGraw-Hill Companies, 2005

Diagrame de tip Use-Case

- Atât **Make Appointment** cat si **Request Medication** includ **Check Patient Record** ca o subsarcină (include)
- E**xtension point** este scris în interiorul cazului de bază **Pay bill**; clasa de extindere **Defer payment** adaugă comportamentul acestui punct de extensie. (extend)
- **Pay Bill** este un caz de utilizare pentru părinte, iar **Bill Insurance** este cazul pentru copil. (generalization)



(TogetherSoft, Inc)

Diagrama de clasă

- O diagramă de clasă descrie clasele și interrelațiile lor
- Folosit pentru descrierea structurii și comportamentului în cazurile de utilizare
- Furnizați un model conceptual al sistemului în ceea ce privește entitățile și relațiile acestora
- Folosit pentru captarea cerințelor, interacțiunea cu utilizatorul final
- Diagramele de clasă detaliate sunt utilizate pentru dezvoltatori

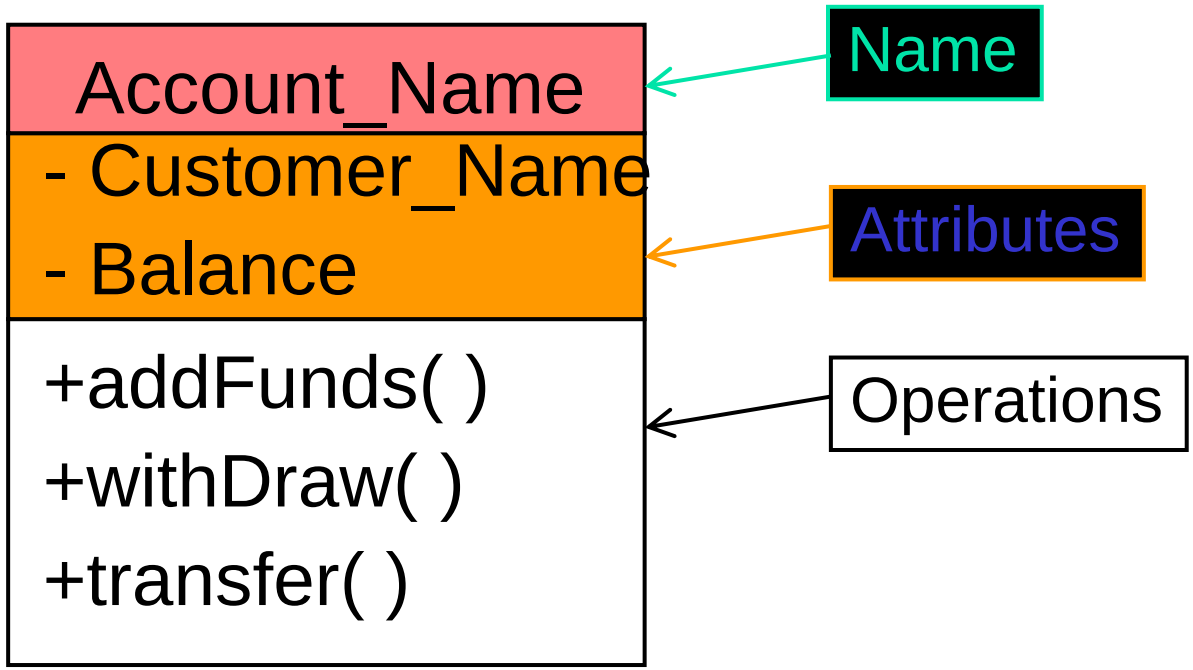
Diagrama de clasă

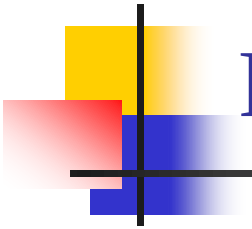
- Fiecare clasă este reprezentată printr-un dreptunghi subdivizat în trei compartimente
  - Name
  - Attributes
  - Operations

- Modificatorii sunt utilizați pentru a indica vizibilitatea atributelor și operațiunilor.
  - '+' este folosit pentru a indica vizibilitatea *Public* (everyone)
  - '#' este folosit pentru a indica vizibilitatea *Protected* (friends and derived)
  - este folosit pentru a indica vizibilitatea *Private* (no one)

- În mod implicit, atributele sunt ascunse și operațiunile sunt vizibile.

Diagrama de clasă

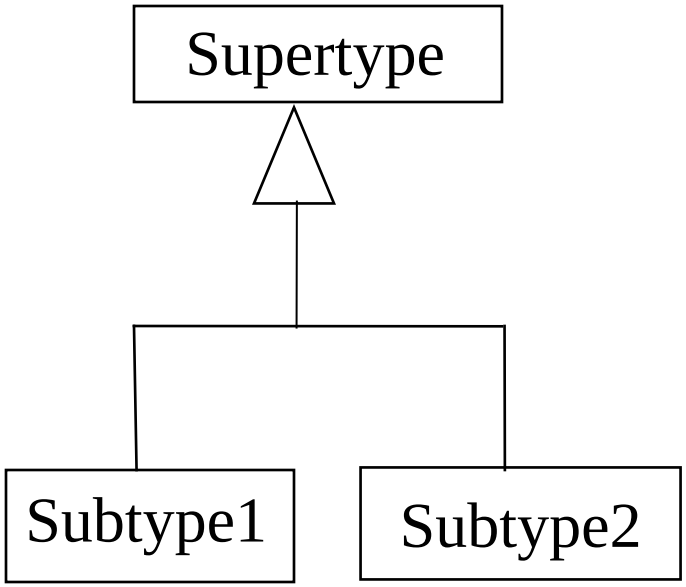




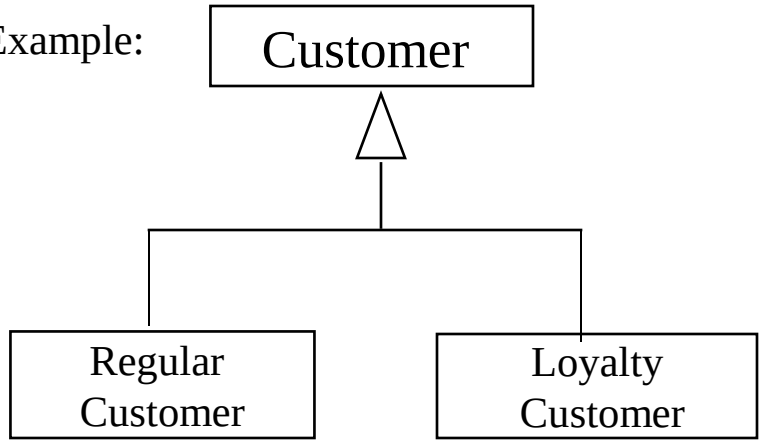
Relatii OO

- Există două tipuri de relații
  - Generalization (relația părinte-copil)
  - Association (studentul se înscrie la curs)
- Asociațiile pot fi clasificate în continuare ca
  - Aggregation
  - Composition

Relatii OO: **Generalizarea**



Example:



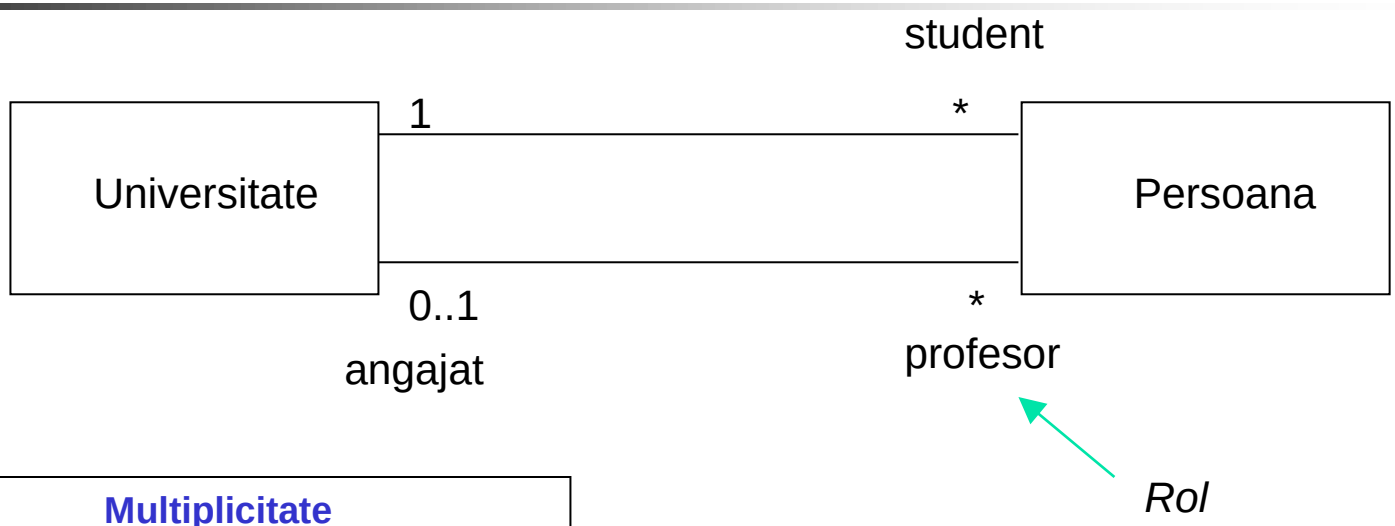
- Moștenirea este o caracteristică necesară a OO
- Generalizarea exprimă o relație părinte/copil între clasele înrudite.
- Folosit pentru abstractizarea detaliilor în diferite layere

Relatii OO: Asocierea

- Reprezinta relaționarea dintre instanțe de clase
  - Reprezinta relaționarea dintre instanțe de clase
  - Studentul se înscrie la un curs
  - Cursurile au studenți
  - Cursurile au examene
  - Etc.

- Asocierea are două capete
  - Nume de roluri (de exemplu, înscrieri)
  - Multiplicitate (de exemplu, un curs poate avea mai mulți studenți)
  - Navigabilitate (unidirecțională, bidirecțională)

Asocierea: Multiplicitate si Roluri

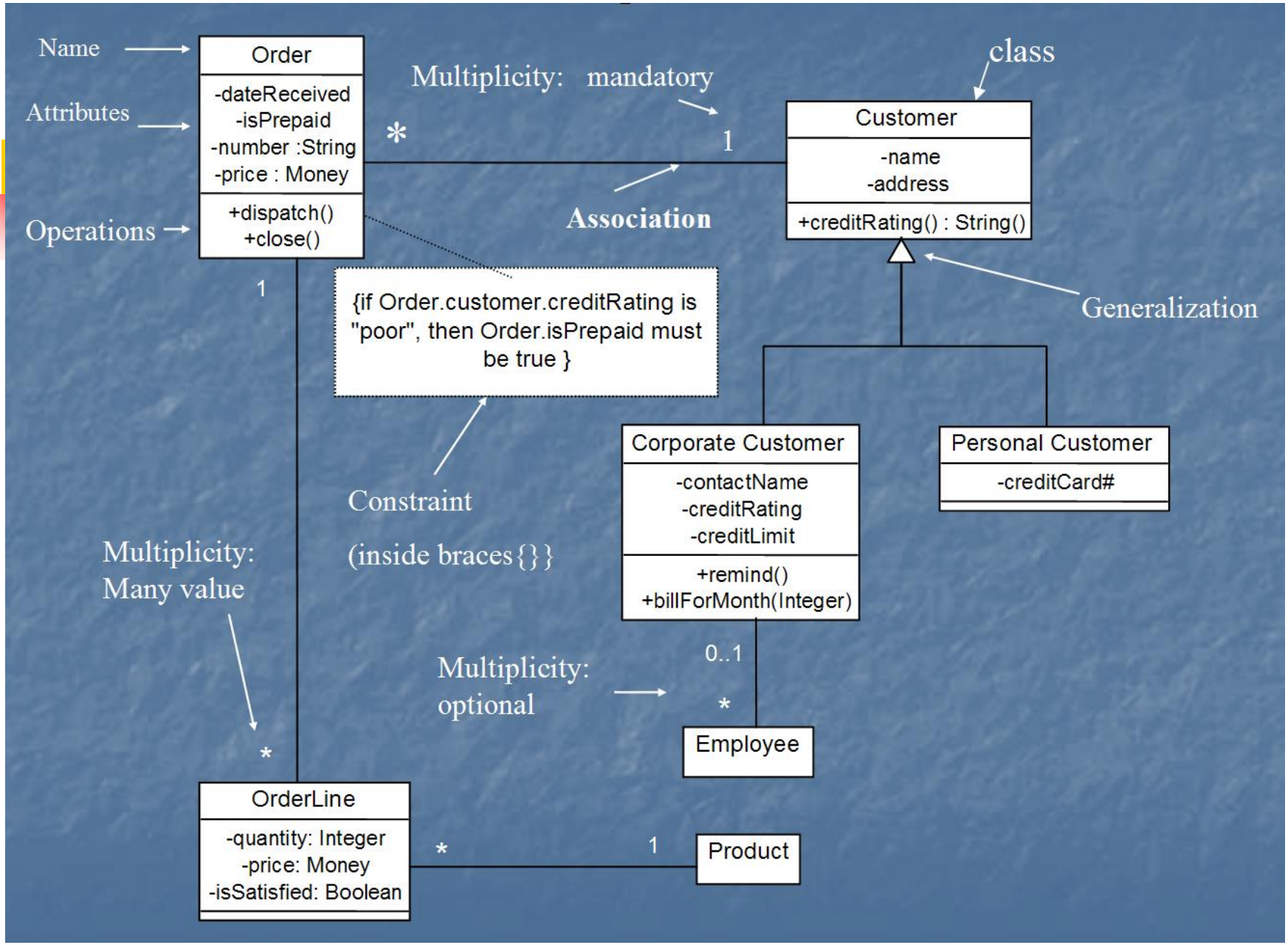


Multiplicitate

Simbol	Insemnatate
1	Unul si numai unul
0..1	Zero or unu
M..N	De la M la N (limbaj natural)
*	De la zero la orice intreg pozitiv
0..*	De la zero la orice intreg pozitiv
1..*	De la one la orice intreg pozitiv

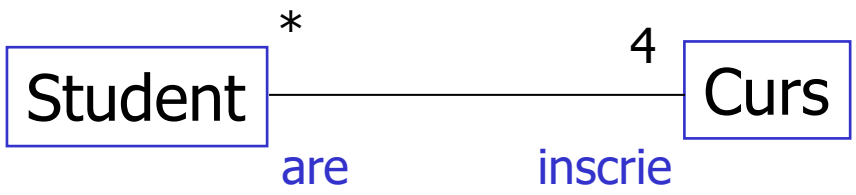
Rol

„O anumită universitate grupează mulți oameni; unii acționează ca studenți, alții ca profesori. Un anumit student aparține unei singure universități; un anumit profesor poate sau nu să lucreze pentru universitate la un anumit moment.”



[from *UML Distilled Third Edition*]

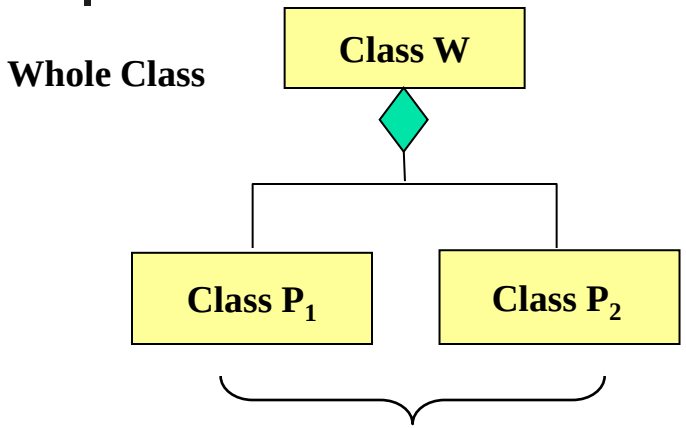
Association: de la Model la Implementare




```
Class Student {
    Course enrolls[4];
}

Class Course {
    Student have[];
}
```

Relatii OO : Compoziţie



[From Dr.David A. Workman]

Example

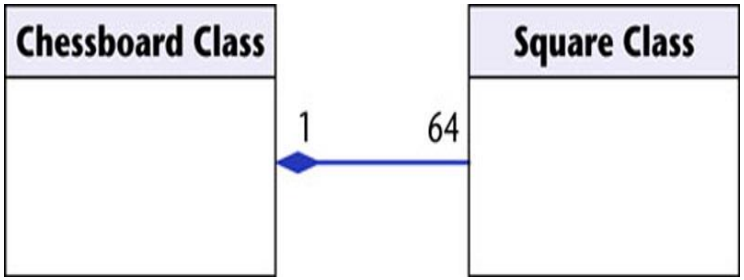


Figure 16.7

Asociere

Modelează relația parte-intreg

Compoziţie

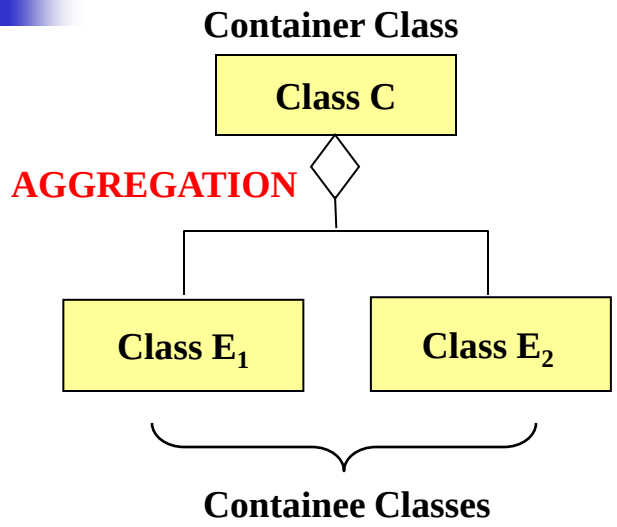
De asemenea, modelează relația parte-intreg, dar, în plus, fiecare parte poate aparține unui singur întreg, iar dacă întregul este șters, la fel și părțile.

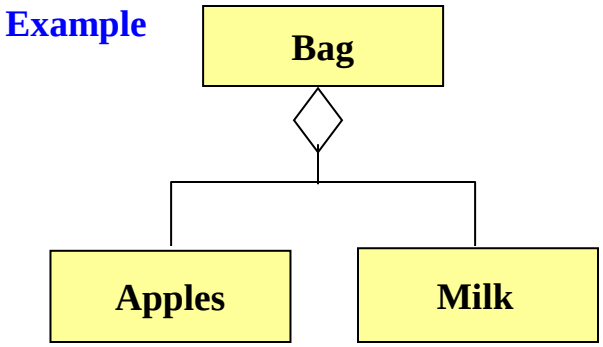
Exemplu:

Un număr de table de șah diferite: Fiecare pătrat aparține unei singure table. Dacă o tablă de șah este aruncată, toate cele 64 de pătrate de pe acea tablă merg și ele.
odels the part–whole relationship

The McGraw-Hill Companies, 2005

Relatii OO : **Agregare**





[From Dr.David A. Workman]

Agregarea:

exprimă o relație între instanțe de clase înrudite. Este un tip specific de relație Container-Container.

exprimă o relație mai informală decât exprimă compoziția.

Agregarea este adecvată atunci când Container și Containees nu au privilegii speciale de acces unul la celălalt.

Agregare vs. Compositie

- **Compoziția** este într-adevăr o formă puternică de **asociere**
  - componentele au un singur proprietar
  - componentele nu pot exista independent de proprietarul lor
  - componentele trăiesc sau mor împreună cu proprietarul lor
  - de exemplu. Fiecare mașină are un motor care nu poate fi partajat cu alte mașini.

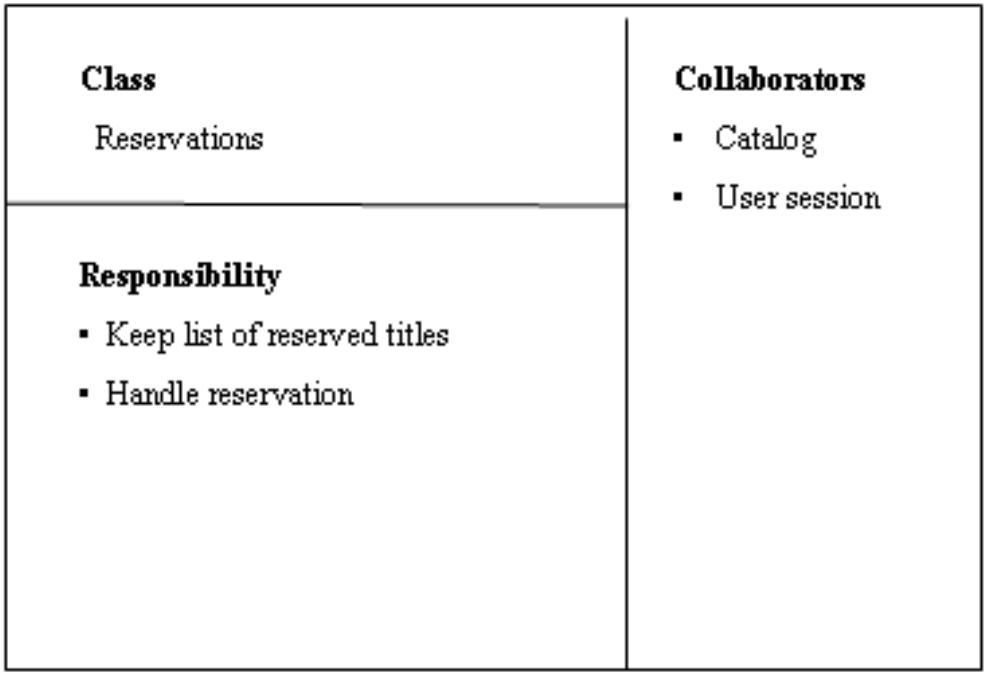
- **Agregările**

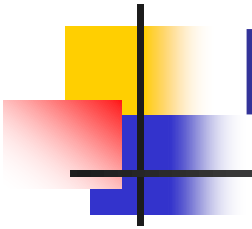
poate face „parte din” asociație, dar poate să nu fie esențială pentru aceasta. Ele pot exista, de asemenea, independent de agregat. de exemplu. Merele pot exista independent de pungă.

Bună practică: Card CRC

Class Responsibility Collaborator

- Descrierea clara a funcționarii claselor prin mutarea cardurilor; permite luarea rapida în considerare a alternativelor.





Diagrame de interacțiune

- Arată cum interacționează obiectele între ele
- UML acceptă două tipuri de diagrame de interacțiune
  - Diagrame de succesiune (sequence)
  - Diagrame de colaborare

Diagrama de secvența (efectuare apel telefonic)

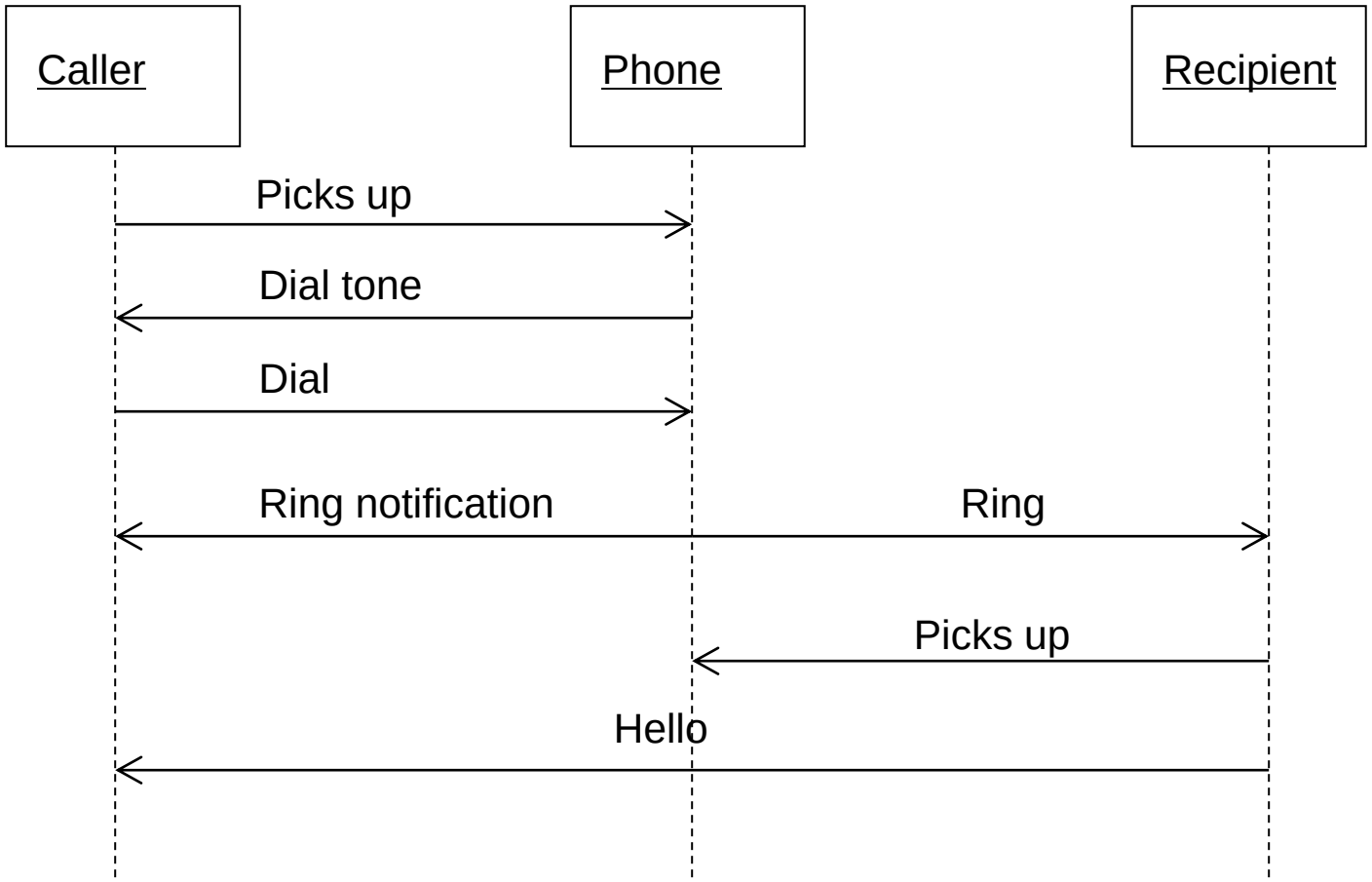
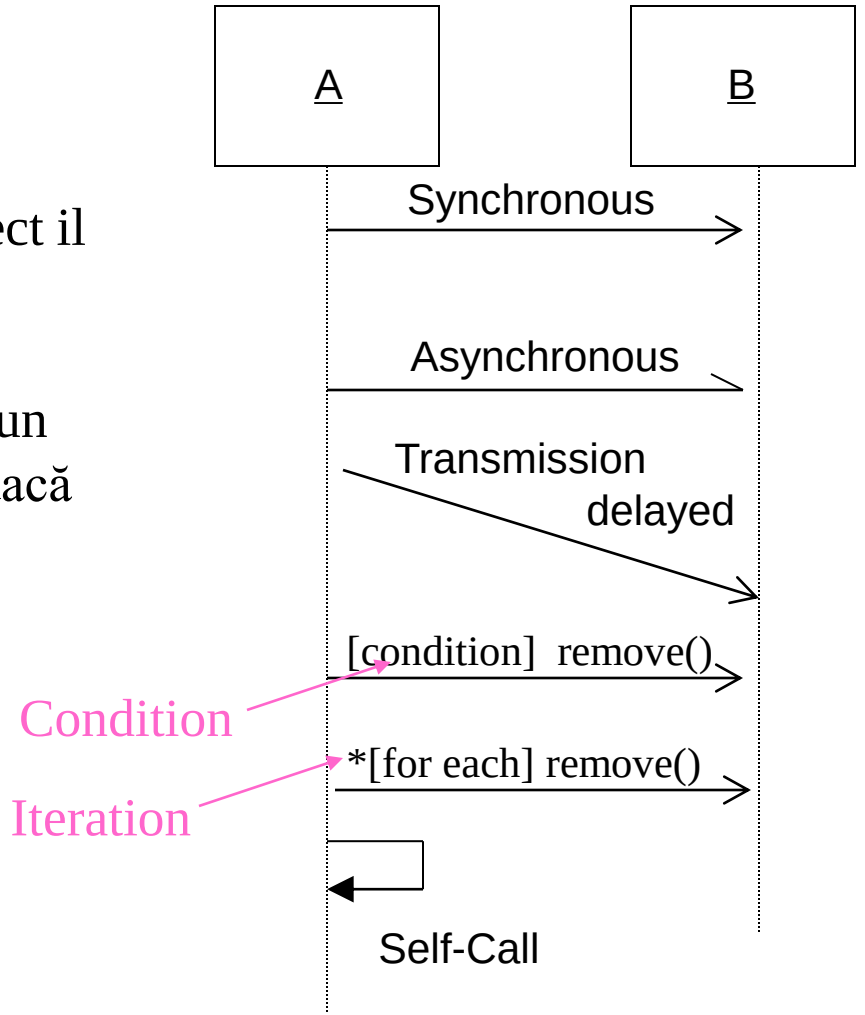


Diagrama de secvența: interacțiunea obiectelor

Self-Call: un mesaj pe care un obiect il trimite la sine insusi.

Condition: indică când este trimis un mesaj. Mesajul este trimis numai dacă condiția este indeplinita.



Diagrame de secvență – Durata de viață a obiectelor

- Creare
  - Mesaj de creare obiect
  - Existenta obiectului începe în acel moment
- Activare
  - Simbolizata de un dreptunghi
  - Pozitionat pe Lifeline acolo unde obiectul este activat.
  - Dreptunghiul indică, de asemenea, când obiectul este dezactivat.
- Stergere
  - Plasarea unui „X” pe Lifeline
  - Viața obiectului se termină în acel moment

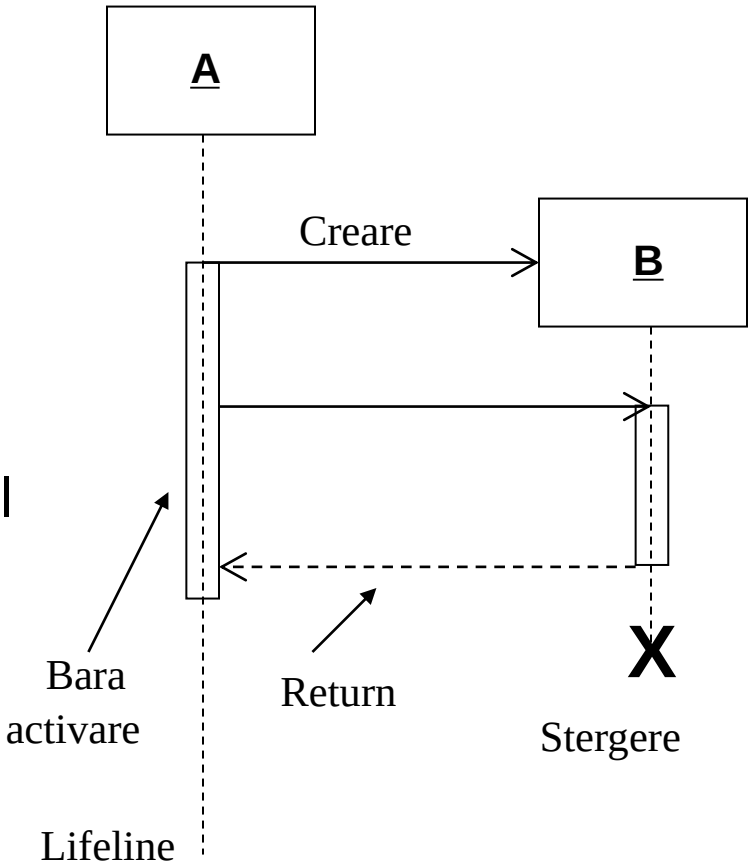
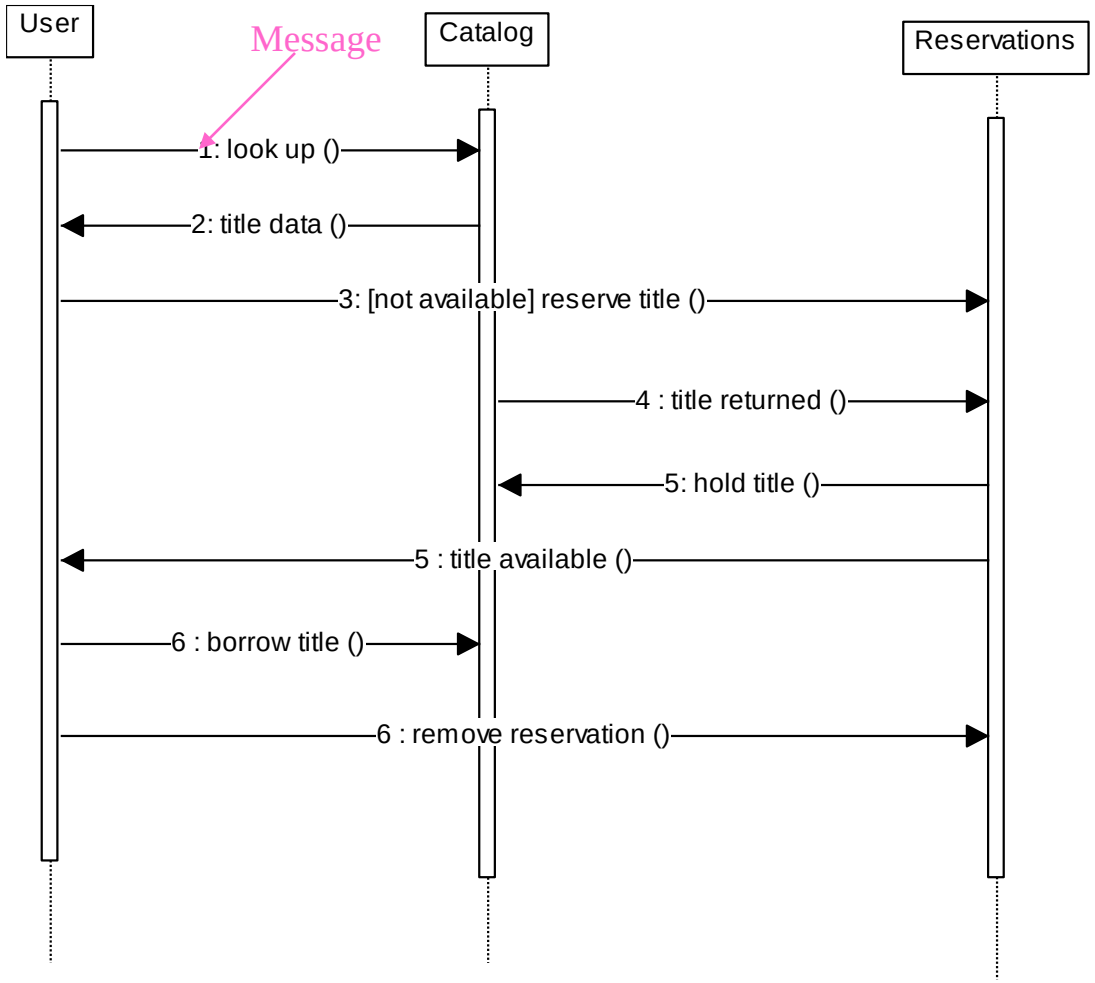
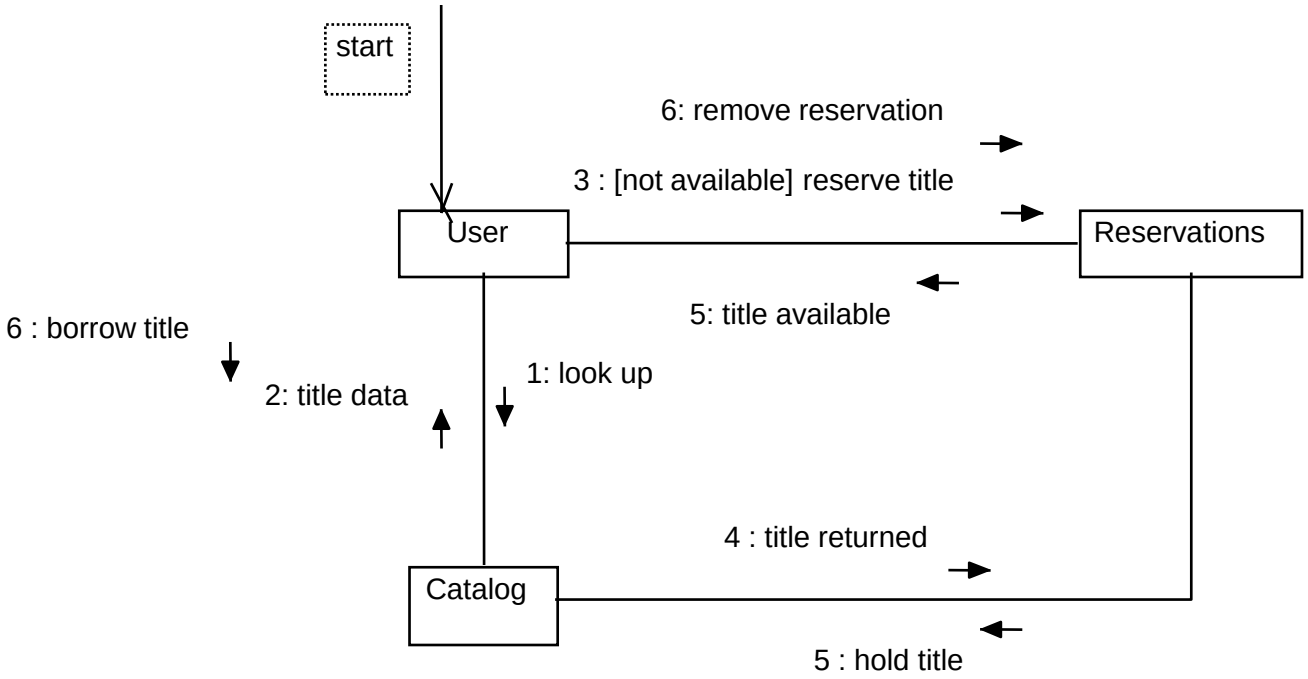


Diagrama de secvență

- Diagramele de secvență demonstrează comportamentul obiectelor într-un caz de utilizare prin descrierea obiectelor și a mesajelor pe care le transmit.
- Dimensiunea orizontală arată obiectele care participă la interacțiune.
- Dispunerea verticală a mesajelor indică ordinea acestora.
- Etichetele pot conține # secv. pentru a indica concurența.



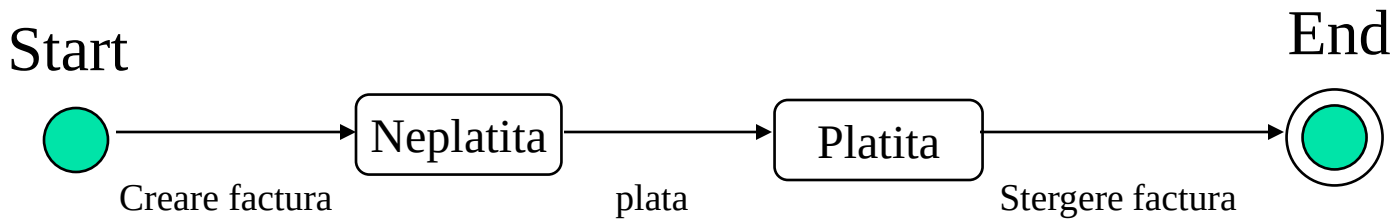
Diagrame de Interactiune: Diagrame de colaborare



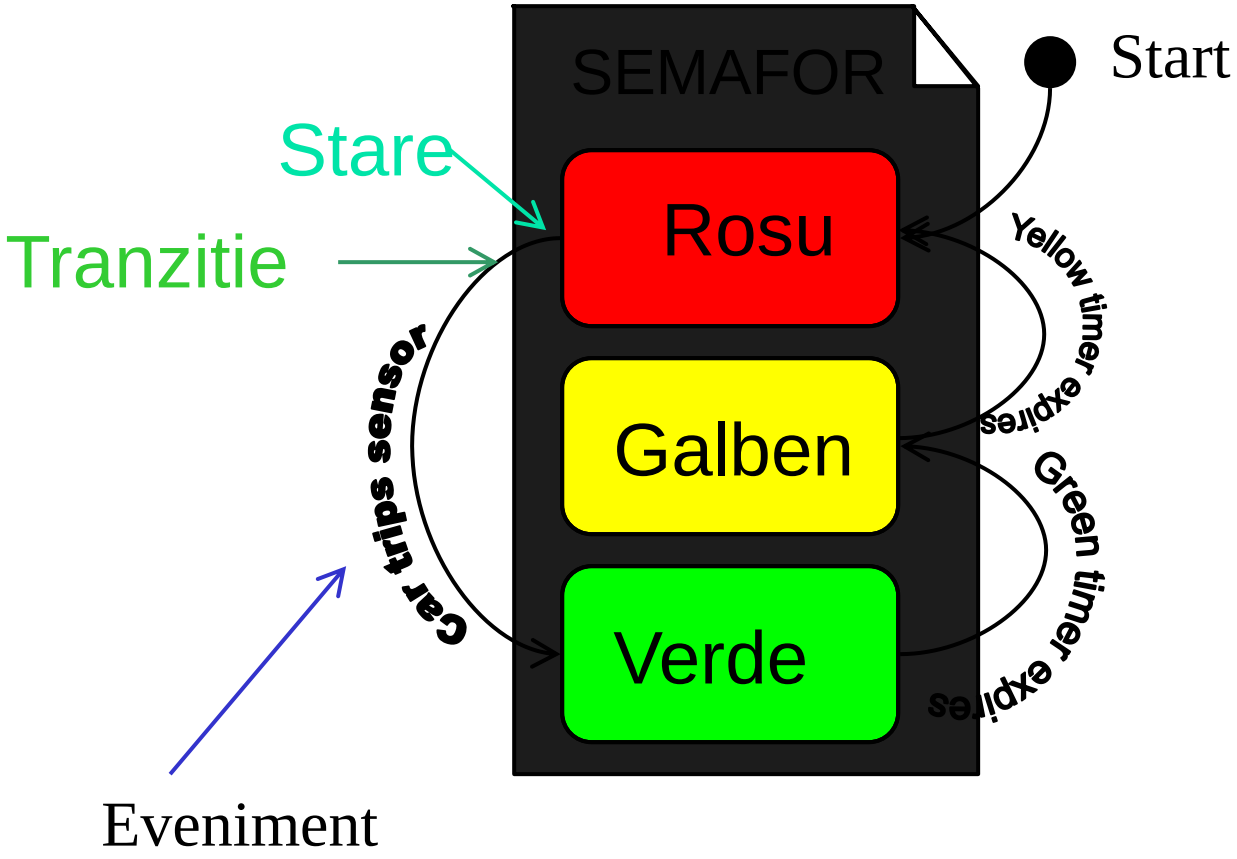
- Diagramele de colaborare sunt echivalente cu diagramele de secvență. Toate caracteristicile diagramelor de secvență sunt aplicabile în mod egal și diagramelor de colaborare
- Utilizați o diagramă de secvență atunci când transferul de informații este în centrul atenției
- Utilizați o diagramă de colaborare atunci când concentrarea este pe clase

Diagrame de Stare (Exemplu de facturare)

Diagramele de stări arată secvențele de stare prin care trece un obiect în timpul ciclului său de viață ca răspuns la stimuli, împreună cu răspunsurile și acțiunile sale; o abstractizare a tuturor comportamentelor posibile.

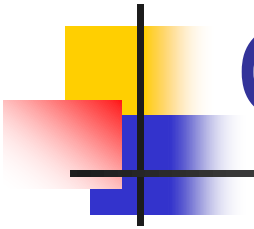


Diagrame de Stări (Exemplu: semafor)



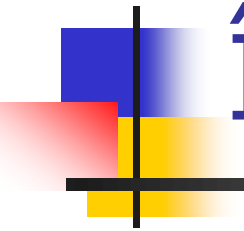
What UML Modeling tools we use today?

- List of UML tools http://en.wikipedia.org/wiki/List_of_UML_tools
- ArgoUML: https://argouml-tigris-org.github.io/tigris/argouml/
- Rational Rose (www.rational.com) by IBM
- UML Studio 7.1 (http://www.pragsoft.com/) by Pragsoft Corporation: Capable of handling very large models (tens of thousands of classes). Educational License US$ 125.00; Freeware version.
- Enterprise Architect by Sparx System and its UML Tutorial



Concluzii

- UML este un limbaj de specificare standardizat pentru modelarea obiectelor
- Cateva diagrame UML:
  - *Diagrama cazurilor de utilizare*: un număr de cazuri de utilizare (modele de caz de utilizare interacțiunea dintre actori și software)
  - *Diagrama de clase*: un model de clase care arată relațiile statice dintre ele, inclusiv asociere și generalizare.
  - *Diagrama de secvențe*: arată modul în care obiectele interacționează unele cu altele pe măsură ce mesajele sunt transmise între ele. Model dinamic
  - *Diagrama de stare*: arată stări, evenimente care provoacă tranziții între stări. Un alt model dinamic care reflectă comportamentul obiectelor și modul în care acestea reacționează la un anumit eveniment
- Există mai multe instrumente UML disponibile



Întrebări?