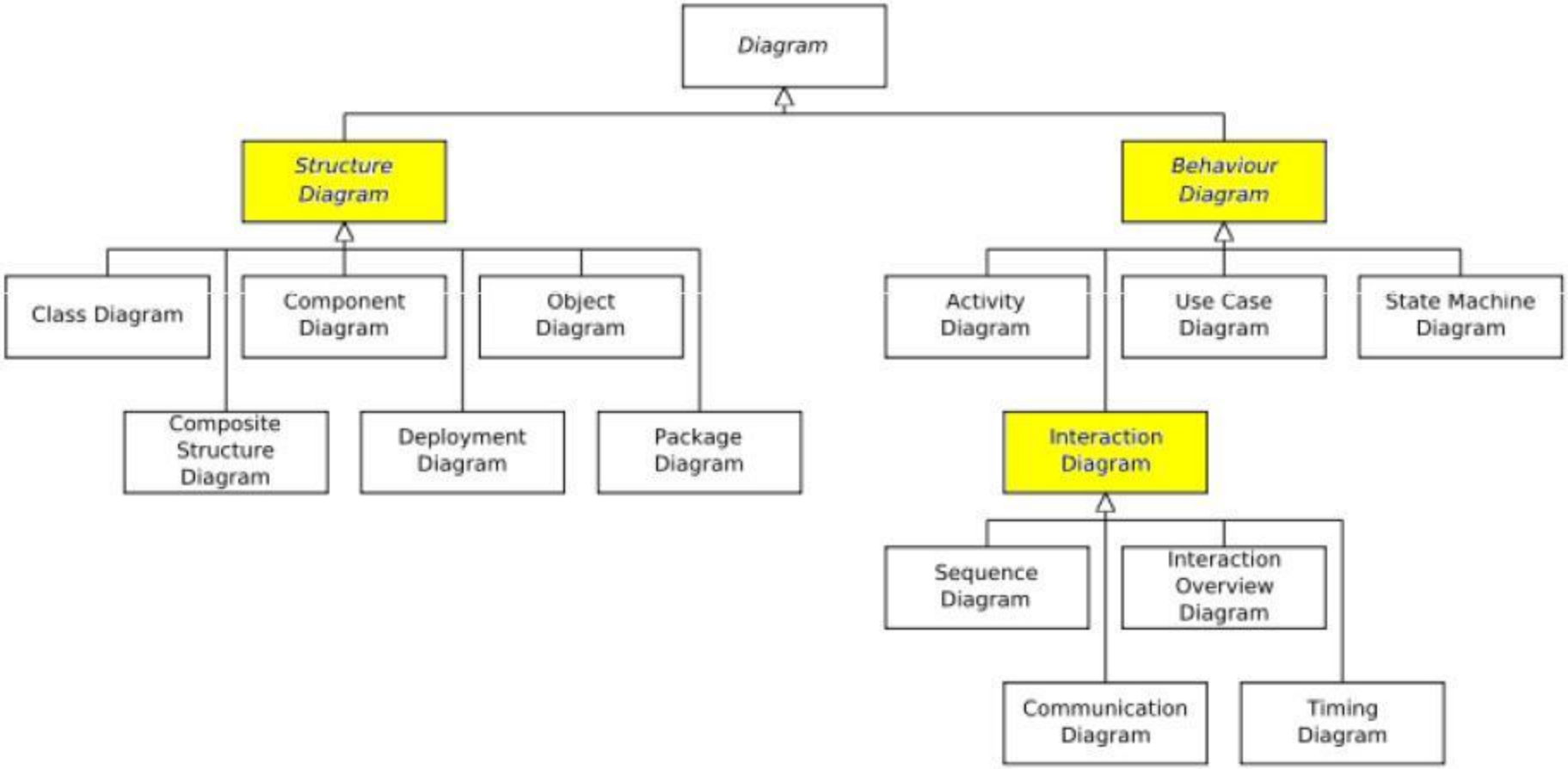


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

[wikipedia]

Diagramele UML (cel mai) des folosite:

- Diagrama cazurilor de utilizare (*use cases diagram*)
- Diagrama de secventa (*sequence diagram*)
- Diagrama de activitati (*activity diagram*)
- Diagrama de clase (*class diagram*)
- Diagrama de stari (*state diagram*)
- Diagrama de componente (*components diagram*)
- Diagrama de distributie (*deployment diagram*)

Diagramele de secventa

- Prezinta **interactiunile** dintre obiecte sau actori si obiecte **din punct de vedere temporal**

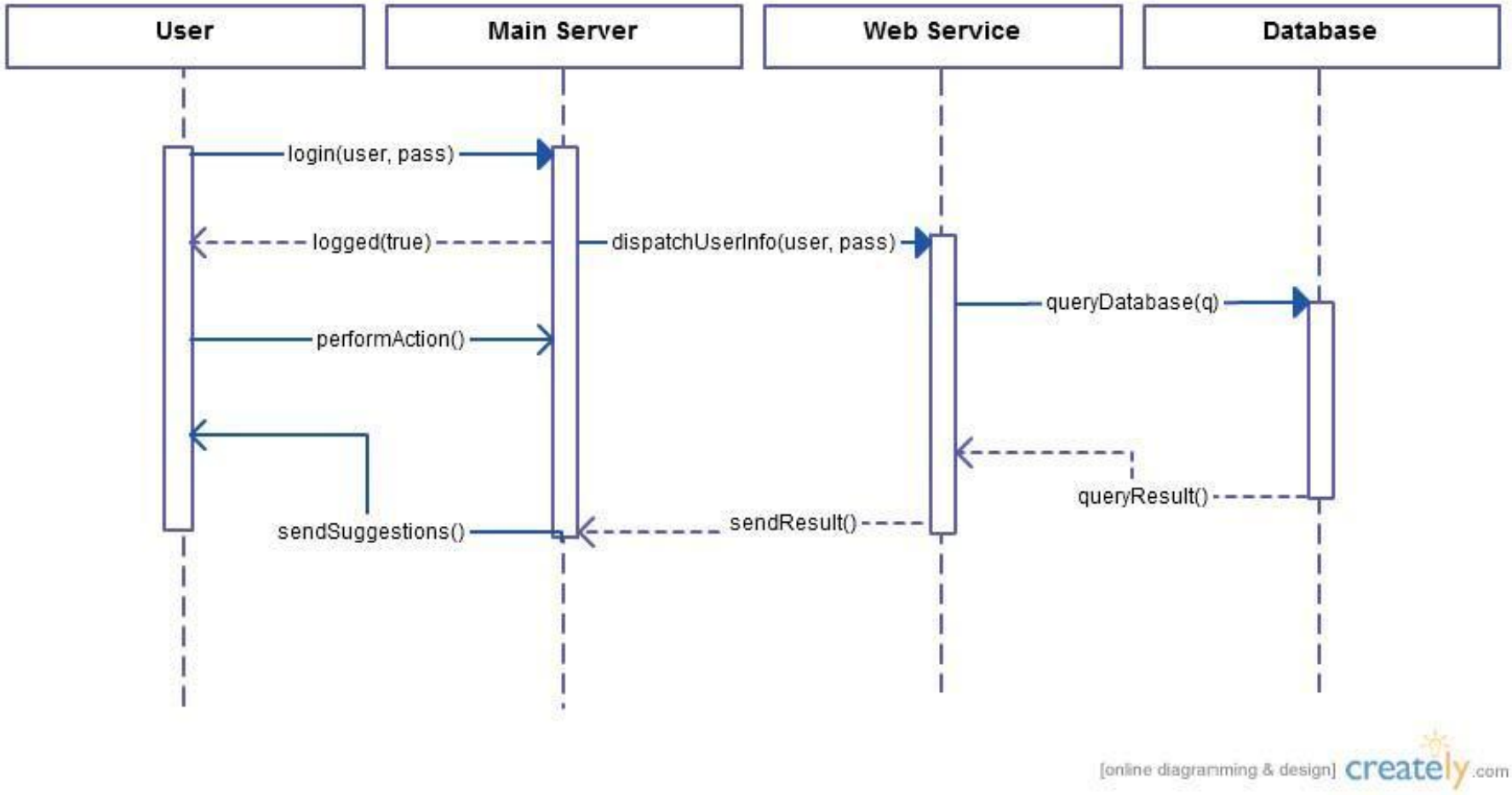
- Se construiesc plecand de la cazurile de utilizare

  a) Ca mijloc de documentare a cazurilor de utilizare: interactiunea este descrisa in termeni apropiati utilizatorului si fara a intra in detalii de sincronizare

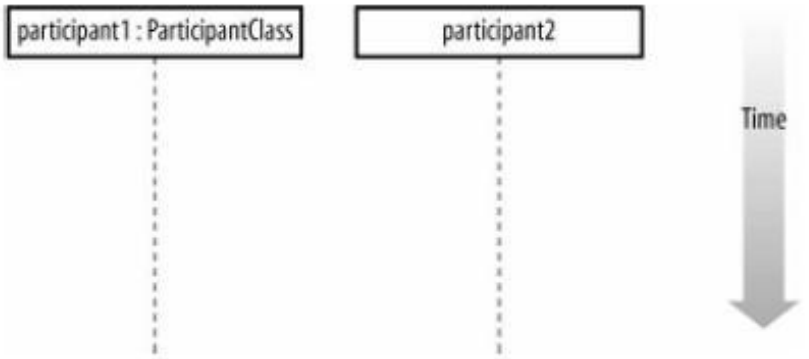
  b) Ca mijloc de reprezentare exacta a mesajelor schimbate intre obiecte. Perioada de activitate a unui obiect este reprezentata cu ajutorul unei benzi rectangulare suprapuse pe linia de viata a obiectului

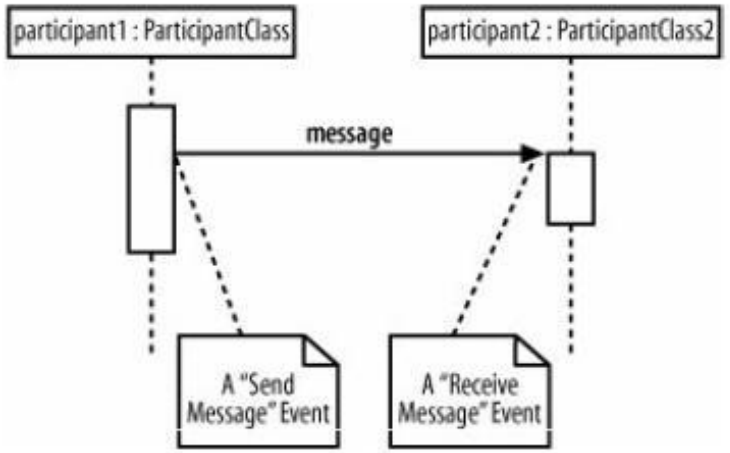
- Se folosesc in mai multe etape ale dezvoltarii
  - Definirea cerintelor, proiectare arhitecturala, proiectare de detaliu
- Ilustreaza:
    - Interactiunile dintre actori si sistem (scenarii)
    - Interactiunile dintre obiectele (entitatile) definite la nivelul sistemului

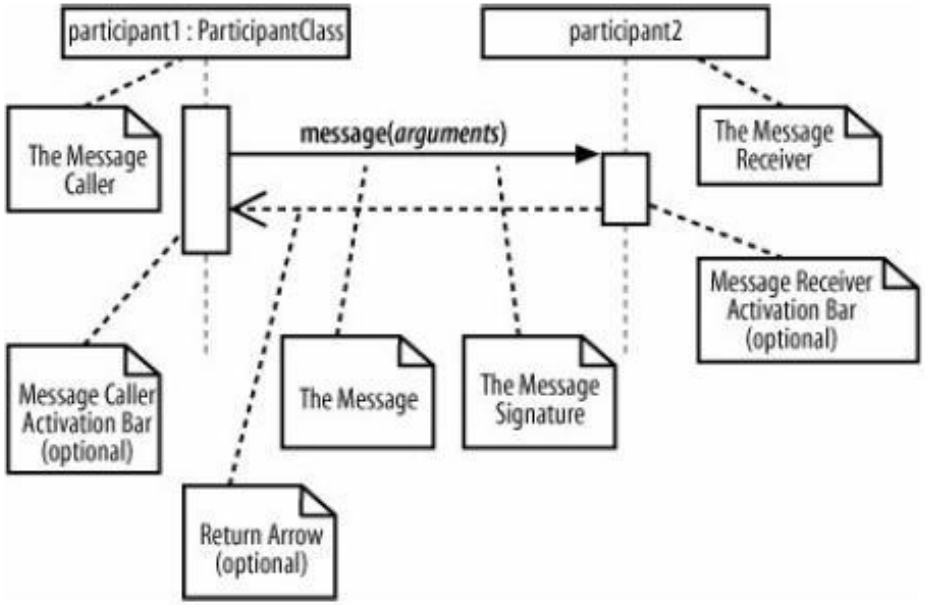
Diagramă de secventa (comportamental): prezinta interactiunea sistemului şi legăturile între ele



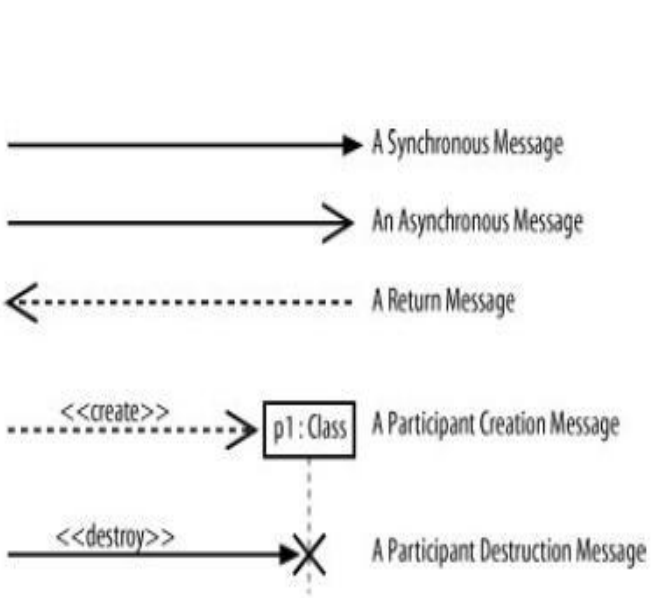
Diagrame de secventa

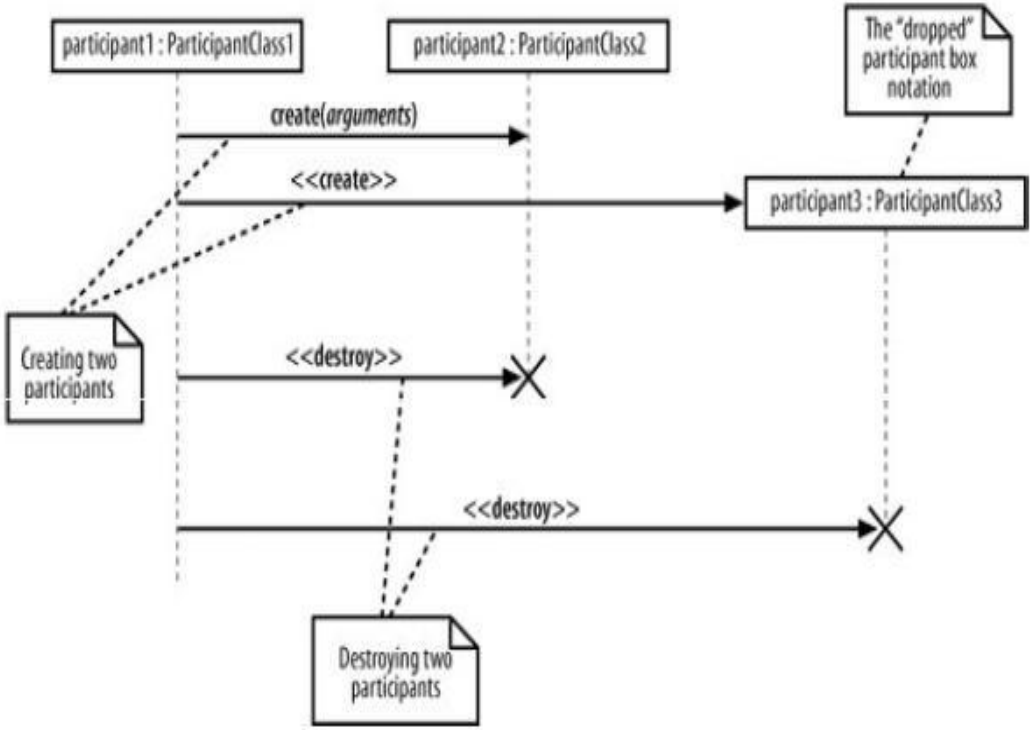




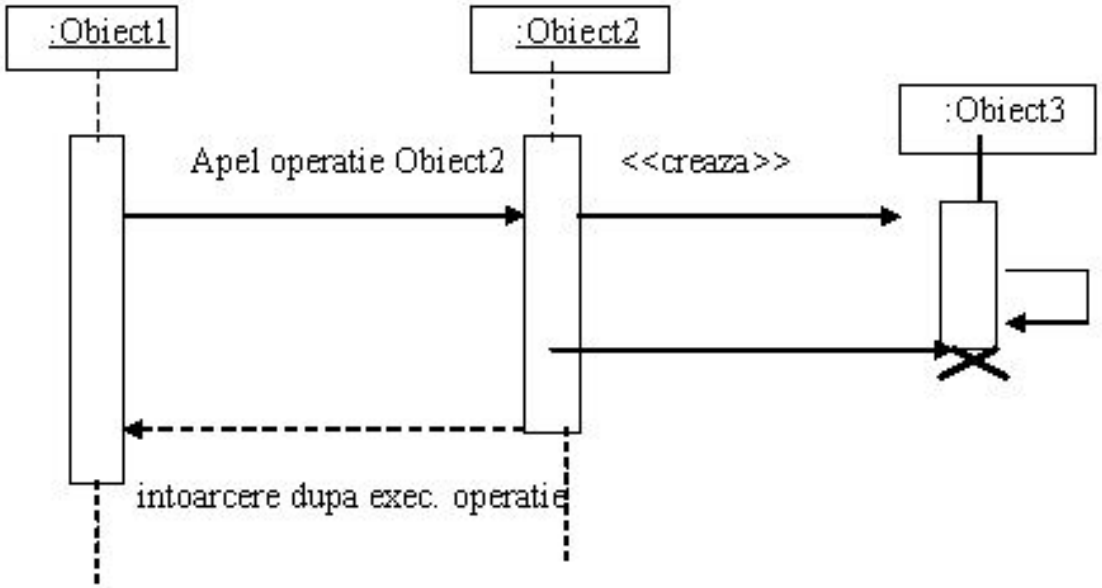


Diagrame de secventa





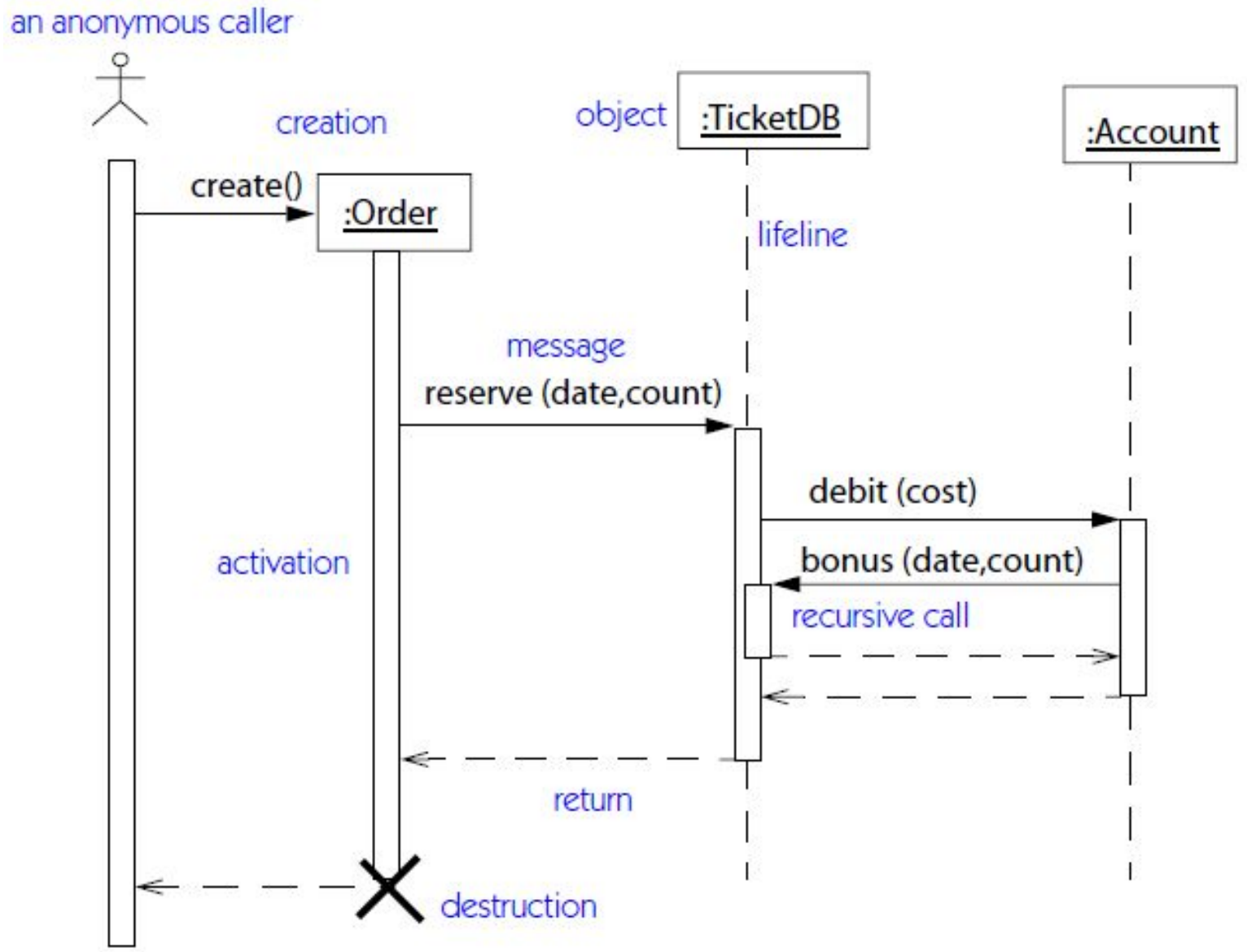
Diagrame de secventa



Apeluri de operatii, crearea si distrugerea obiectelor.

- Se poate reda perioada de activitate a unui obiect printr-o banda rectangulara suprapusa pe linia sa de viata

Diagramă de secventa (comportamental): prezinta interactiunea sistemului şi legăturile între ele



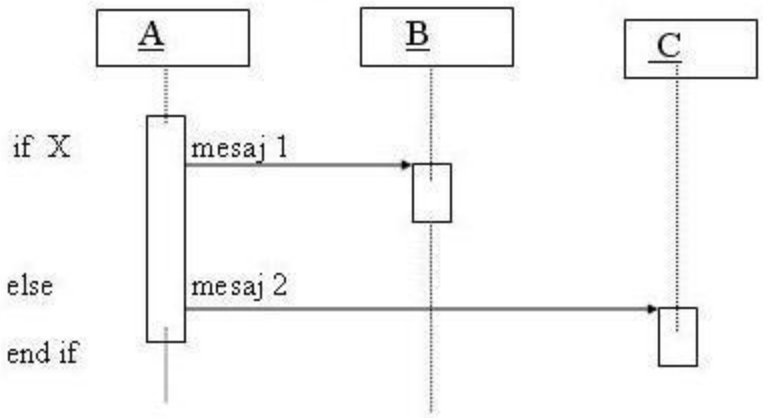
Diagrame de secventa

- **Comunicare sincrona**. Controlul executiei trece de la A la B si revine la A dupa ce B isi termina executia. Exemplu: apel de functie.
- Cand un obiect trimite un mesaj sincron, ramane in asteptare pana cand destinatarul trateaza mesajul
- Revenirea dupa tratarea unui mesaj sincron nu este necesar sa fie reprezentata

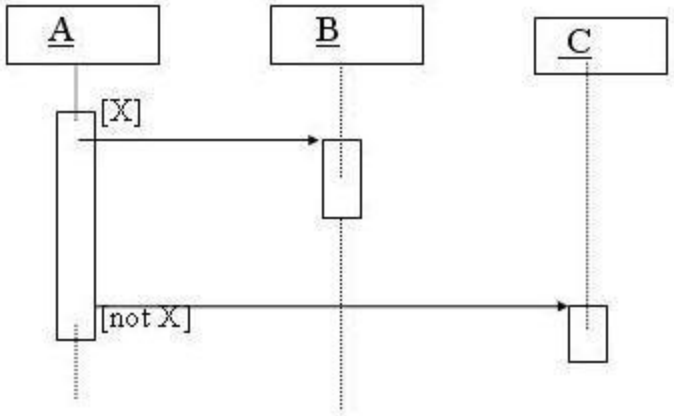
- **Comunicare asincrona**. A trimite un semnal dupa care isi continua executia mai departe. Exemple: startea unui fir de executie nou, aruncarea unei exceptii.

- Trimiterea asincrona a unui mesaj nu întrerupe execuţia expeditorului
- Expeditorul trimite mesajul fără sa stie când, nici chiar dacă mesajul va fi tratat de către destinatar

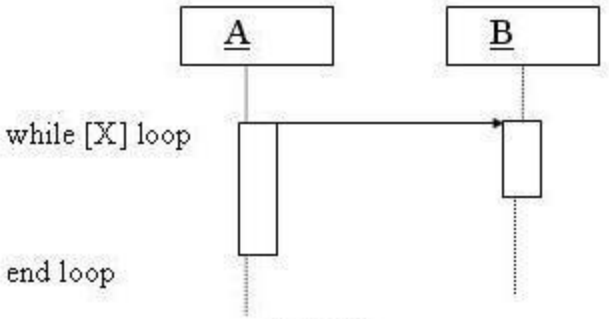
Diagrame de secventa (ramificatii si iteratii)



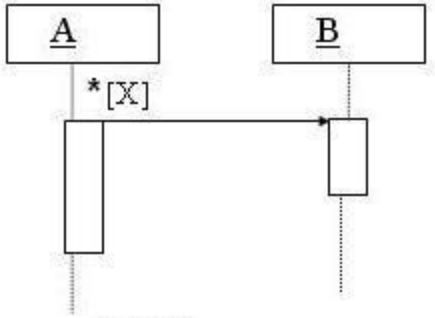
sau



Decizie.



Iterație.



Iterație.

Diagrama de secventa (modelul general)

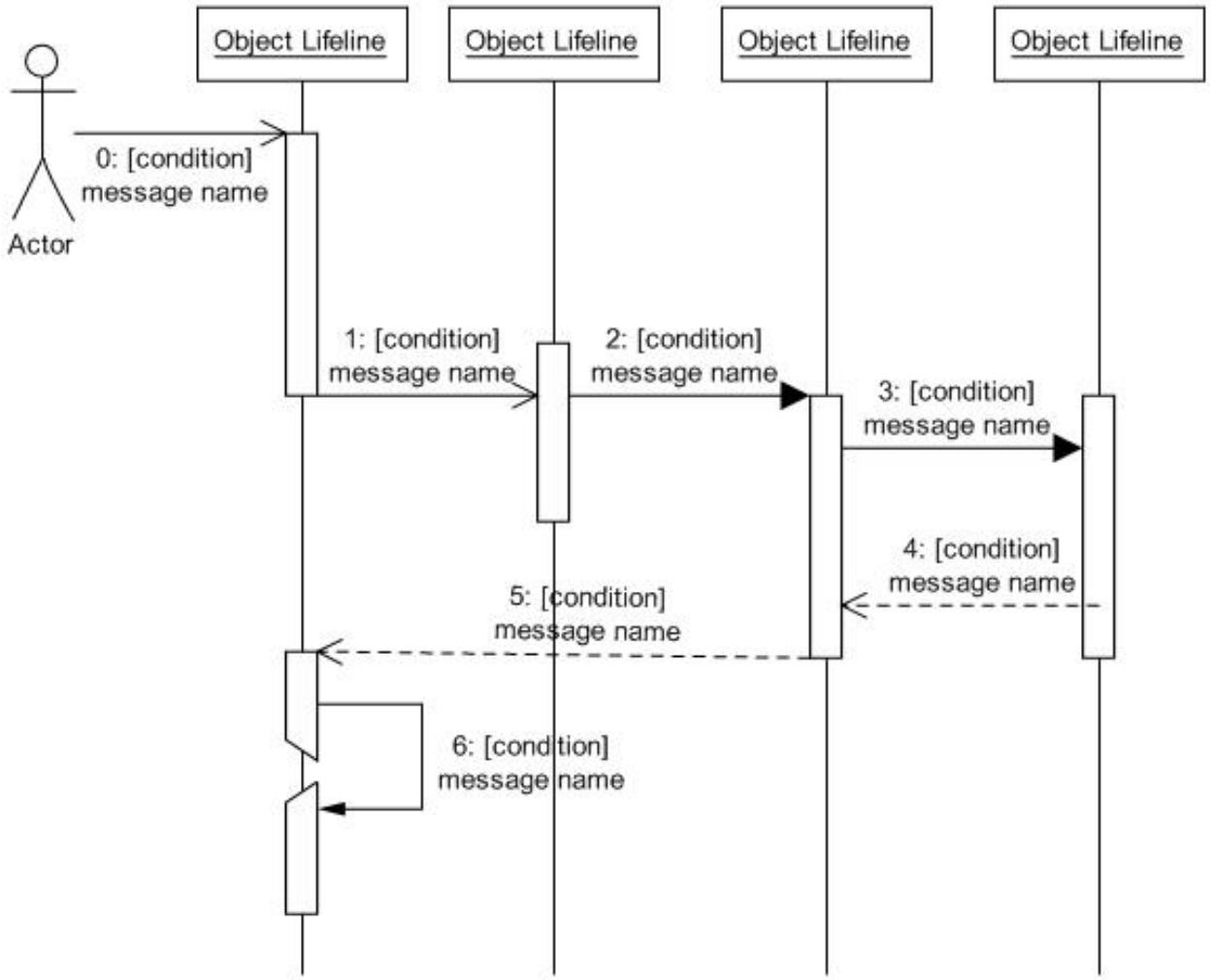


Diagrama de secventa (exemplu)

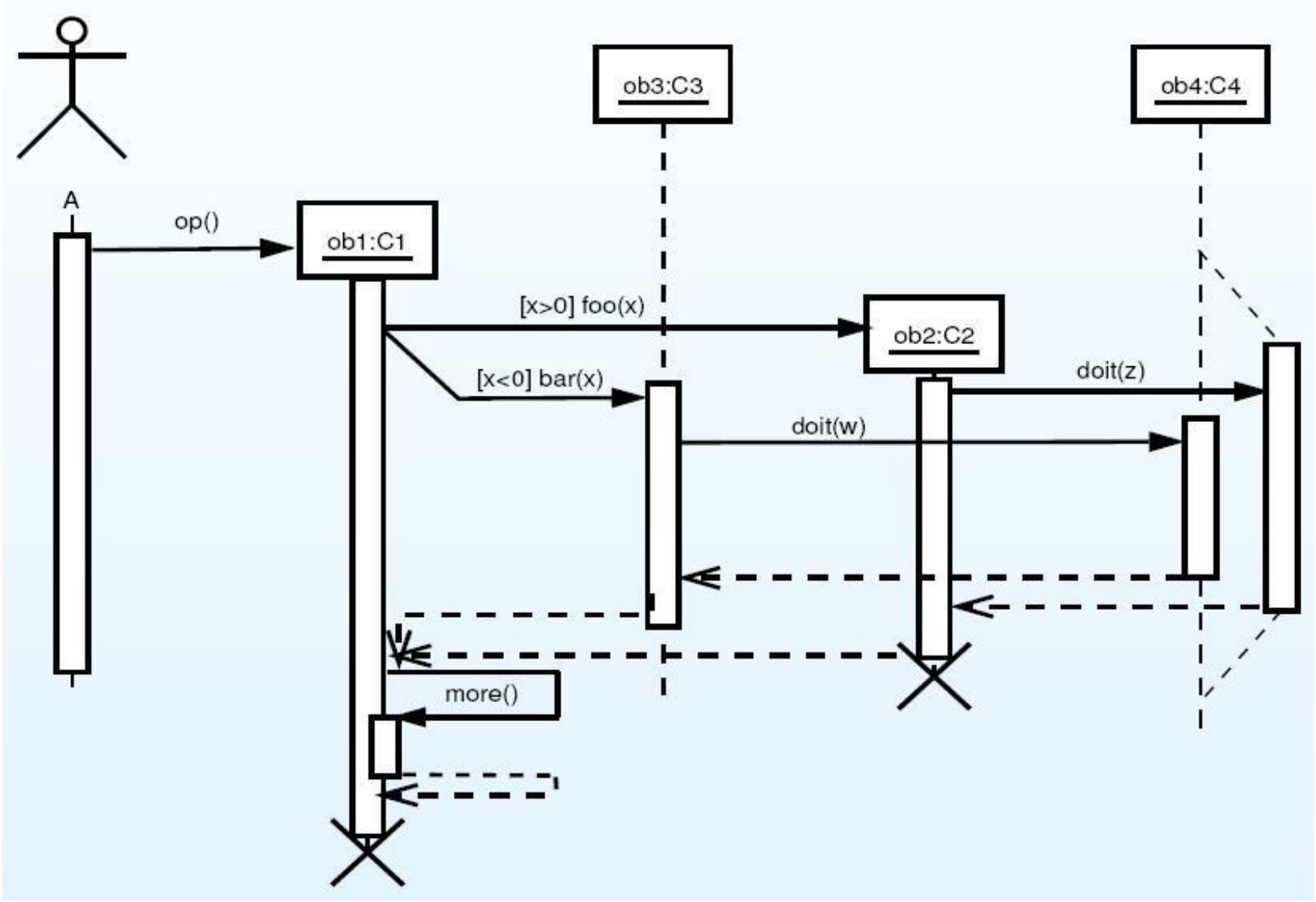
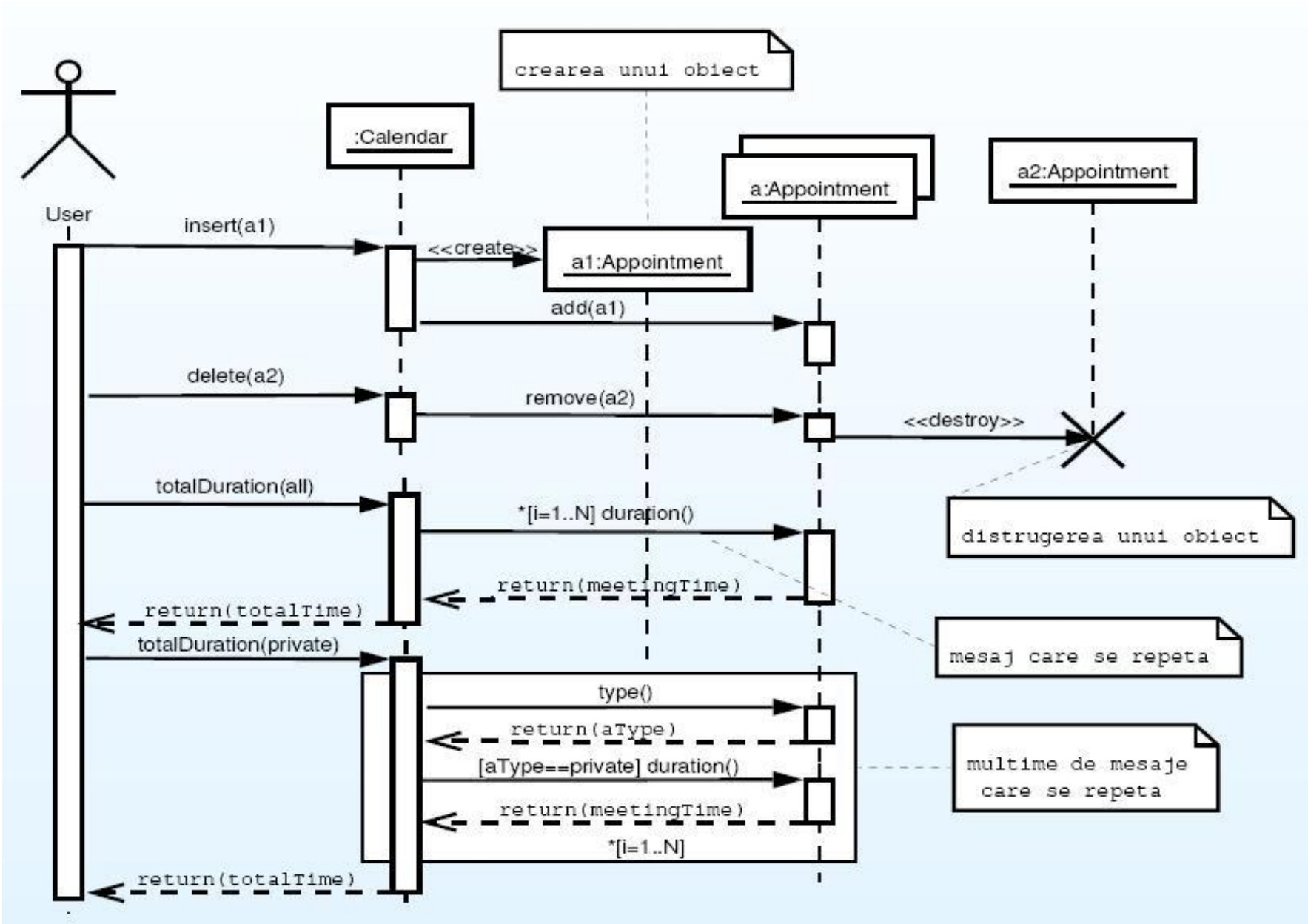


Diagrama de secventa (exemplu)



Diagrame de clase – Concepte *OOP*

Obiect, **Clasă**, **Instanţă**

- Obiect: Entitate care are: identitate, stare, comportament
  - Exemplu: Mingea mea galbena de tenis, cu diametrul de 10 cm, care sare
- Clasă: Descriere a unei mulţimi de obiecte cu aceleaşi caracteristici structurale si comportamentale
  - Exemplu: mingi care au culoare, diametru, întrebuinţare, sar
- Instanţă: un obiect care aparţine unei clase
  - Exemplu: Popescu Viorel este un Student

Diagrame de clase – Concepte *OOP*

Abordare bazata pe: **Incapsulare, Mostenire, Polimorfism**

Incapsularea (datelor)

- Înseamnă punerea la un loc a datelor (atributelor) şi a codului (metodelor)
- Datele pot modificate (doar) prin intermediul metodelor
- Data hiding: nu ne interesează cum se oferă serviciile, ci doar ca se oferă
- Dacă se schimba structura, sau modul de realizare,interfaţa rămâne neschimbată

Mostenirea

- Anumite clase sunt specializări (particularizări) ale altor clase
- O subclasa are (moşteneşte) caracteristicile superclasei, pe care le poate extinde într-un anume fel
- O instanţă a unei clase derivate este în mod automat şi o instanţă a clasei de bază
  - Exemplu (Student - Persoana)

Polimorfism

- Interpretarea semanticii unui apel de metoda se face de către cel care primeşte apelul
  - Exemplu: Eu spun unei forme: DESENEAZĂ-TE. Ea, daca e pătrat trage 4 linii, daca e cerc, face nişte puncte la o anumita distanta în jurul centrului
- Nu mă interesează cine, cum face

Diagrame de clase

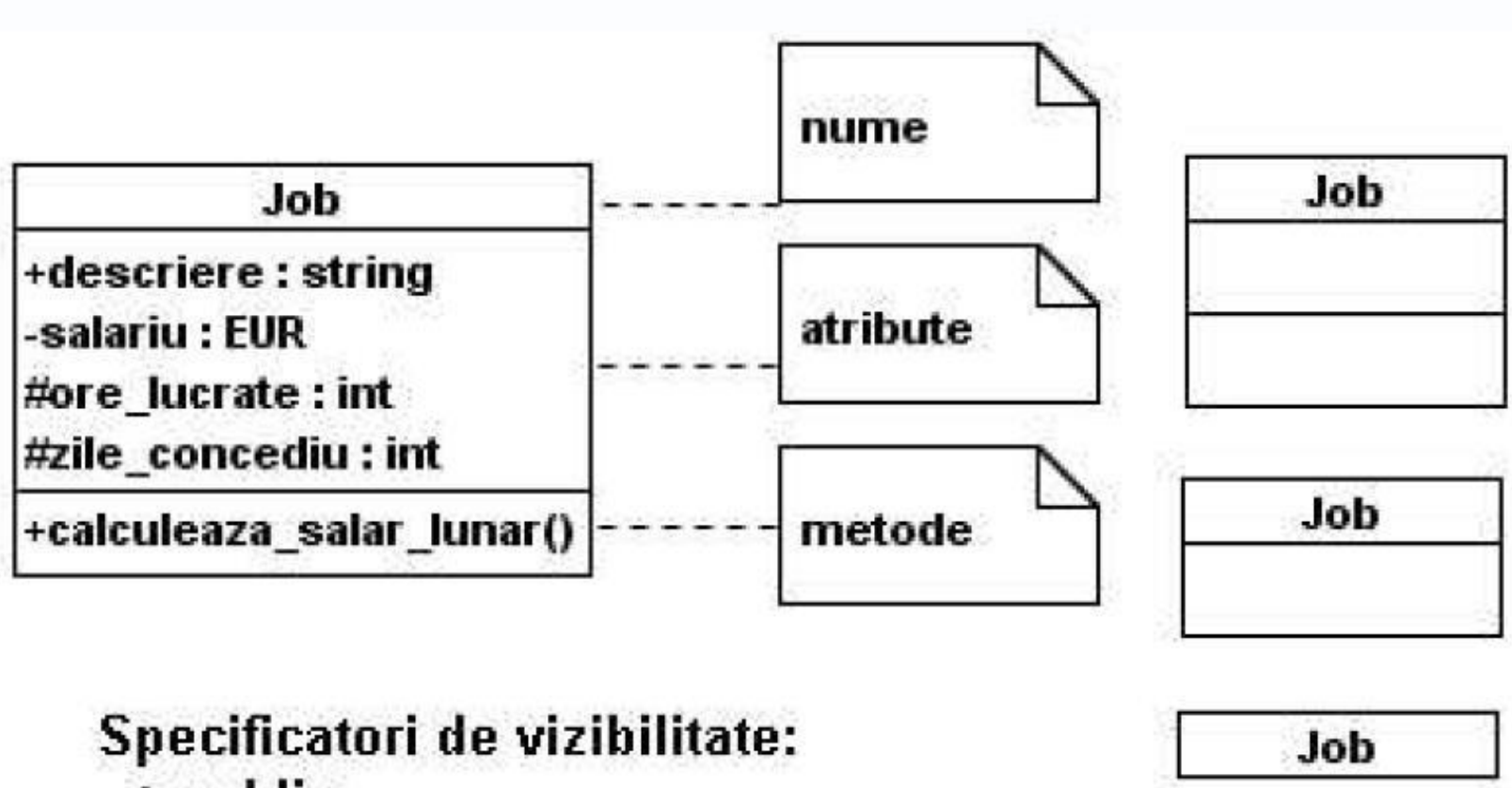
Scop

- Modelează vocabularul sistemului ce trebuie dezvoltat
- Surprinde conexiunile semantice sau interacţiunile care se stabilesc între elementele componente
- Folosită pentru a modela structura unui program

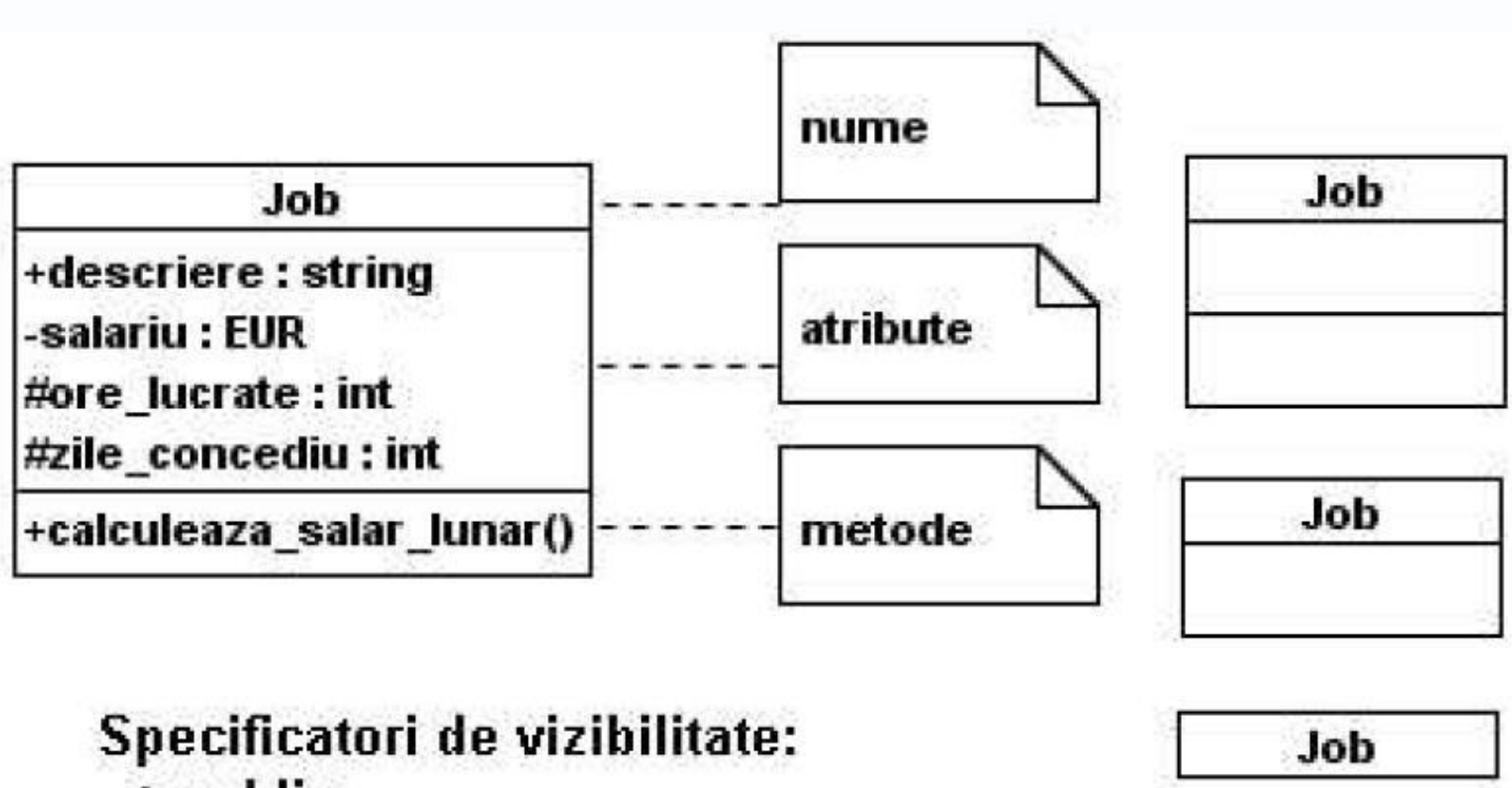
Contine

- Clase / Interfete
- [Obiecte]
- Relatii
  - Asocierea, Agregare, Generalizare, Dependenta

Diagrame de clase – reprezentarea clasei

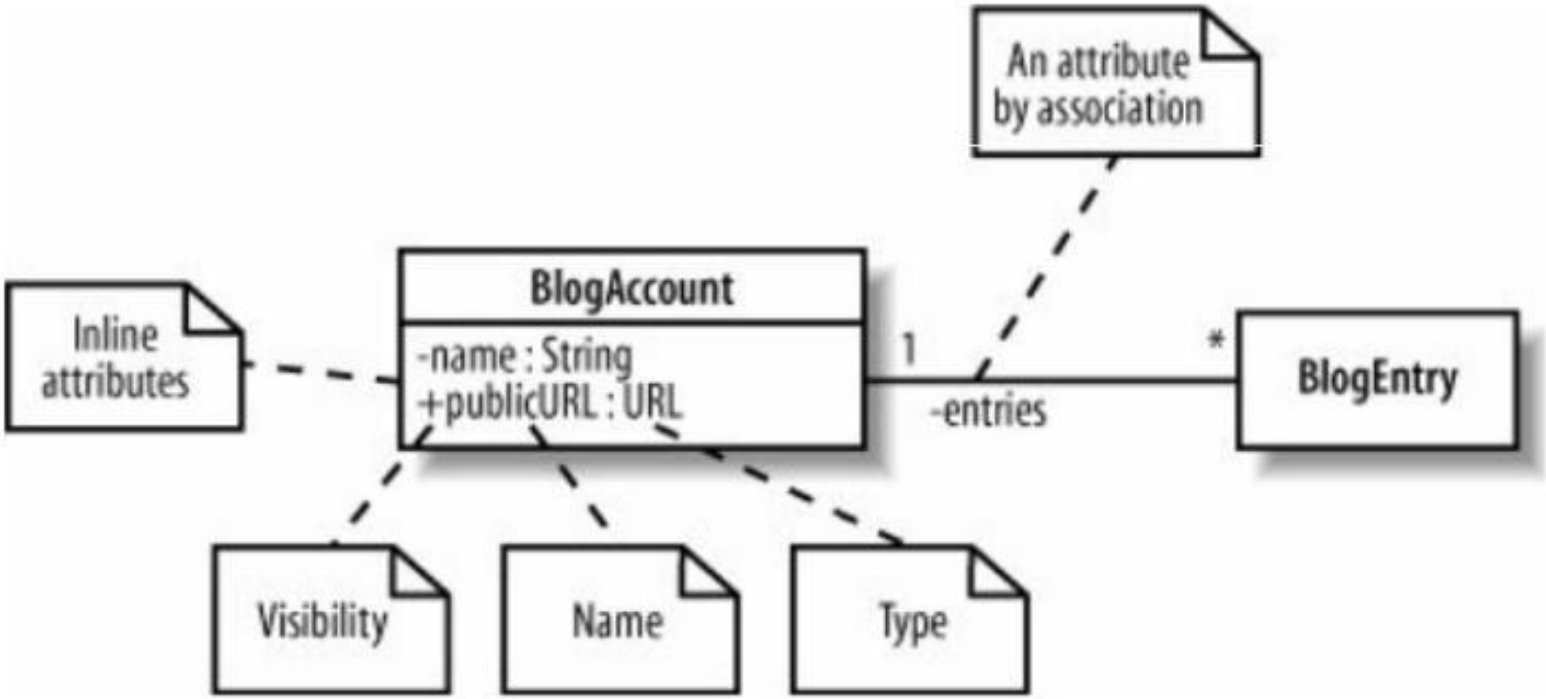


Diagrame de clase – reprezentarea clasei

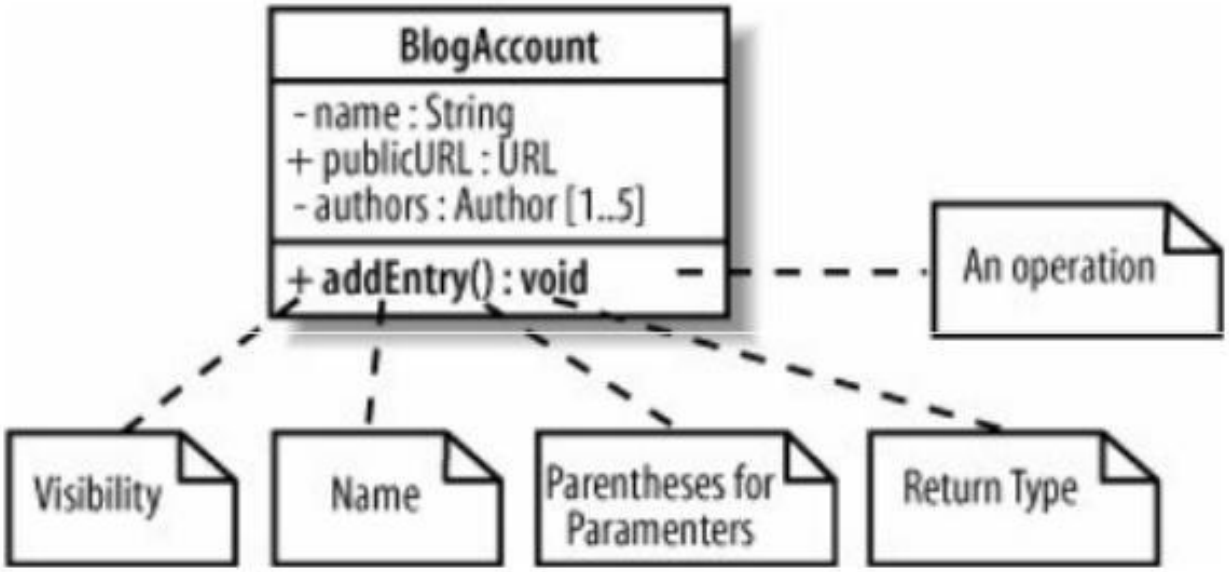


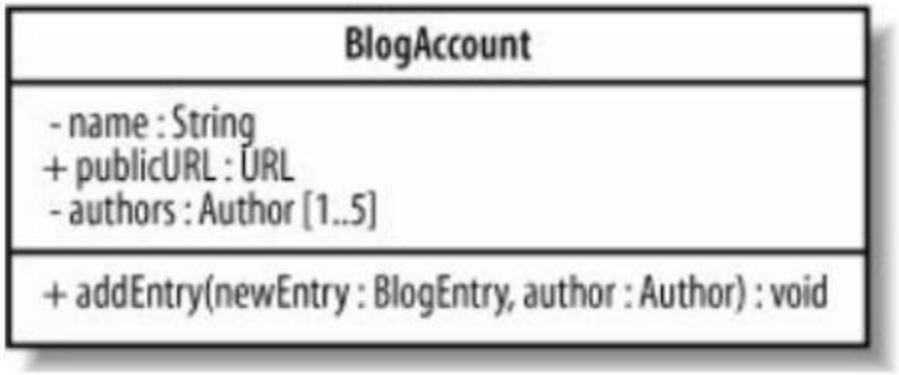
Diagrame de clase – reprezentarea atributelor

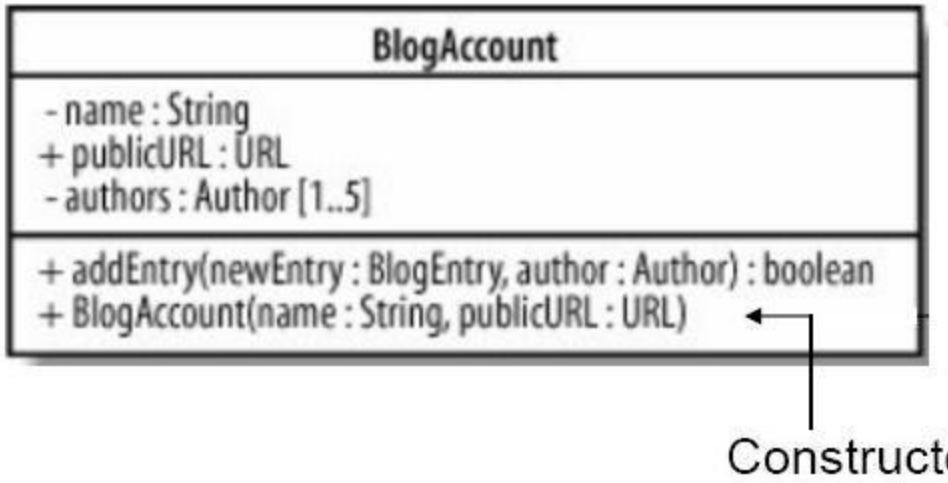
- Inline (in interiorul clasei)
- Prin asociere



Diagrame de clase – reprezentarea operatiilor

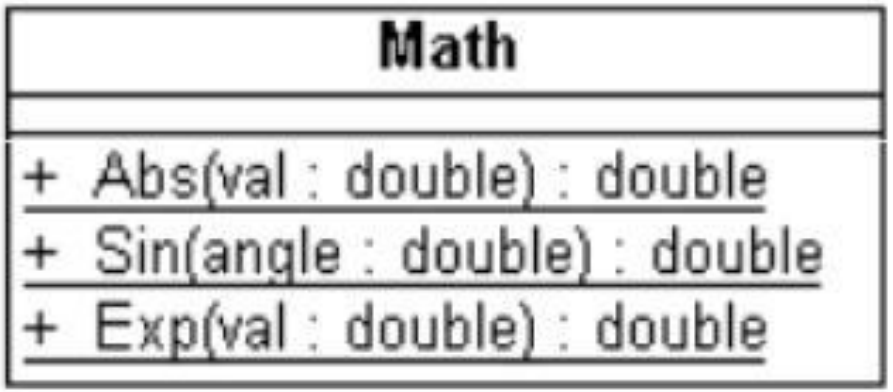




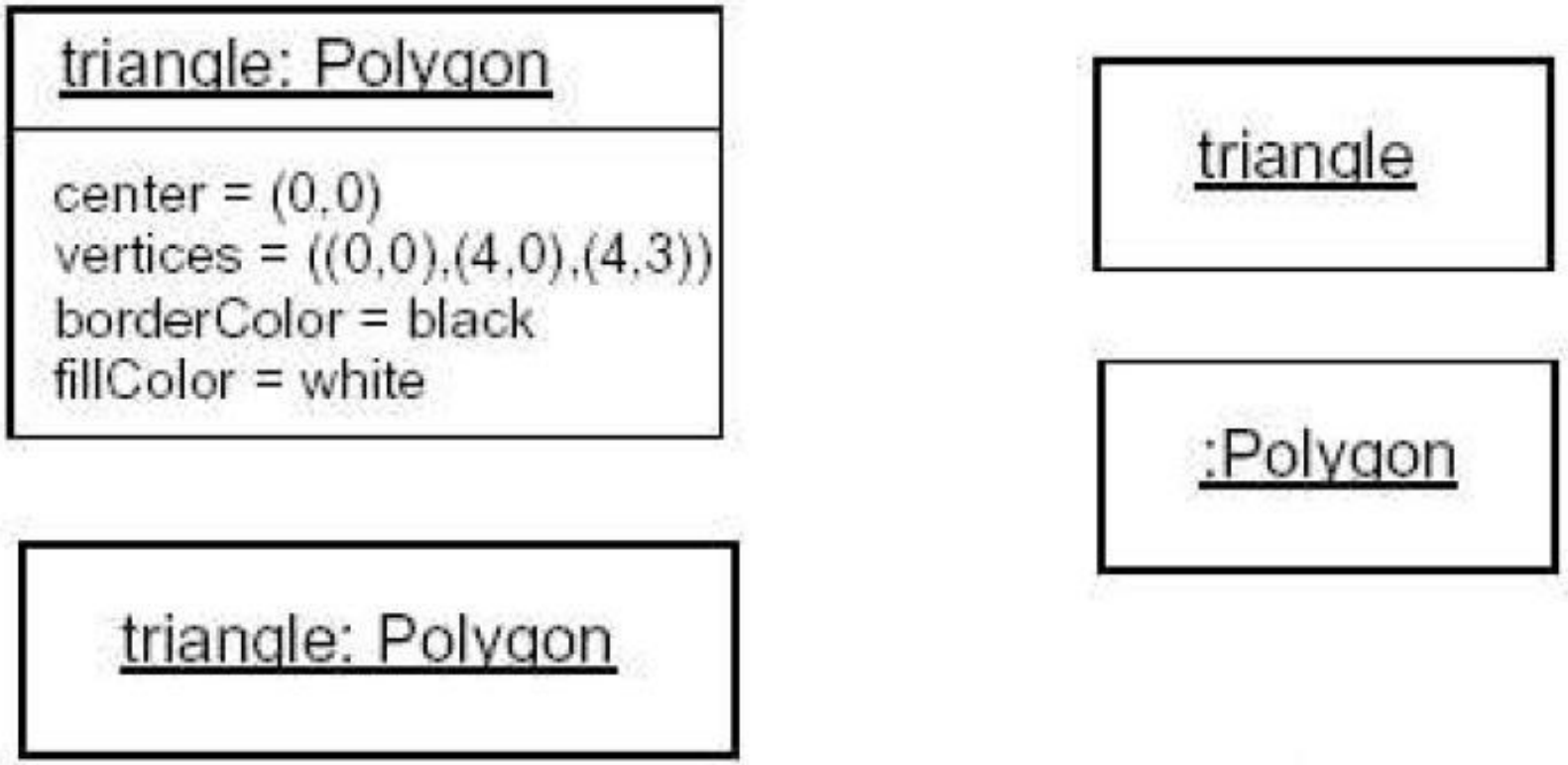


Diagrame de clase – membrii statici

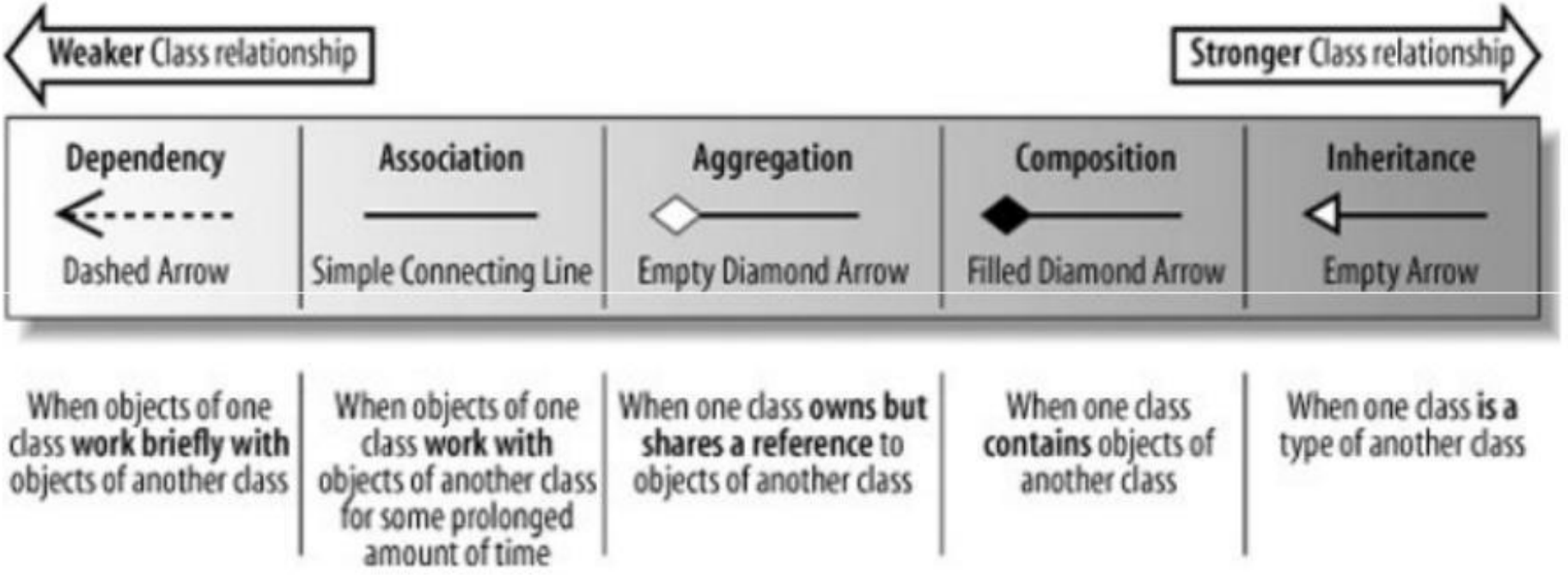
- Membrii statici: atribute si operatii
- Membrii statici: se subliniaza



Diagrame de clase – reprezentarea obiectelor

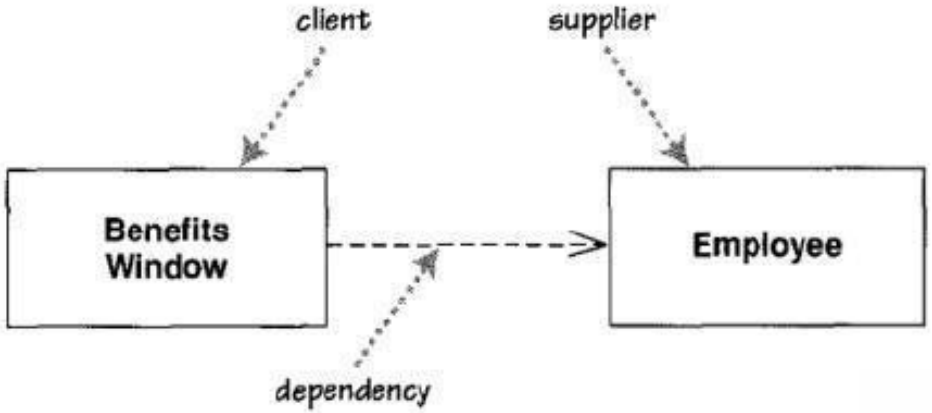


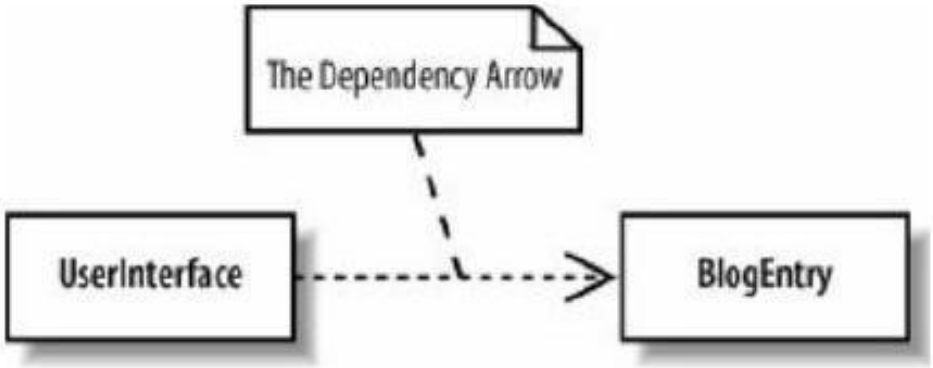
Diagrame de clase – Relatii



Diagrame de clase – Dependenta

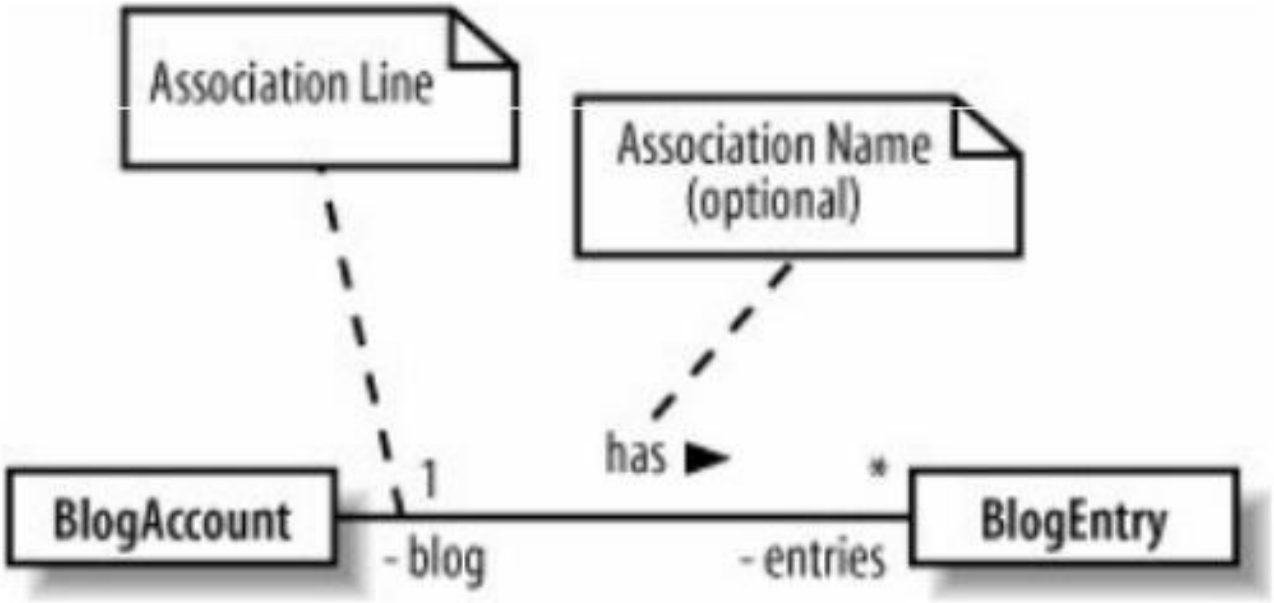
- O clasa foloseste pentru scurt timp o alta clasa
  - Exemplu: trimiterea ca parametru in cadrul unei metode



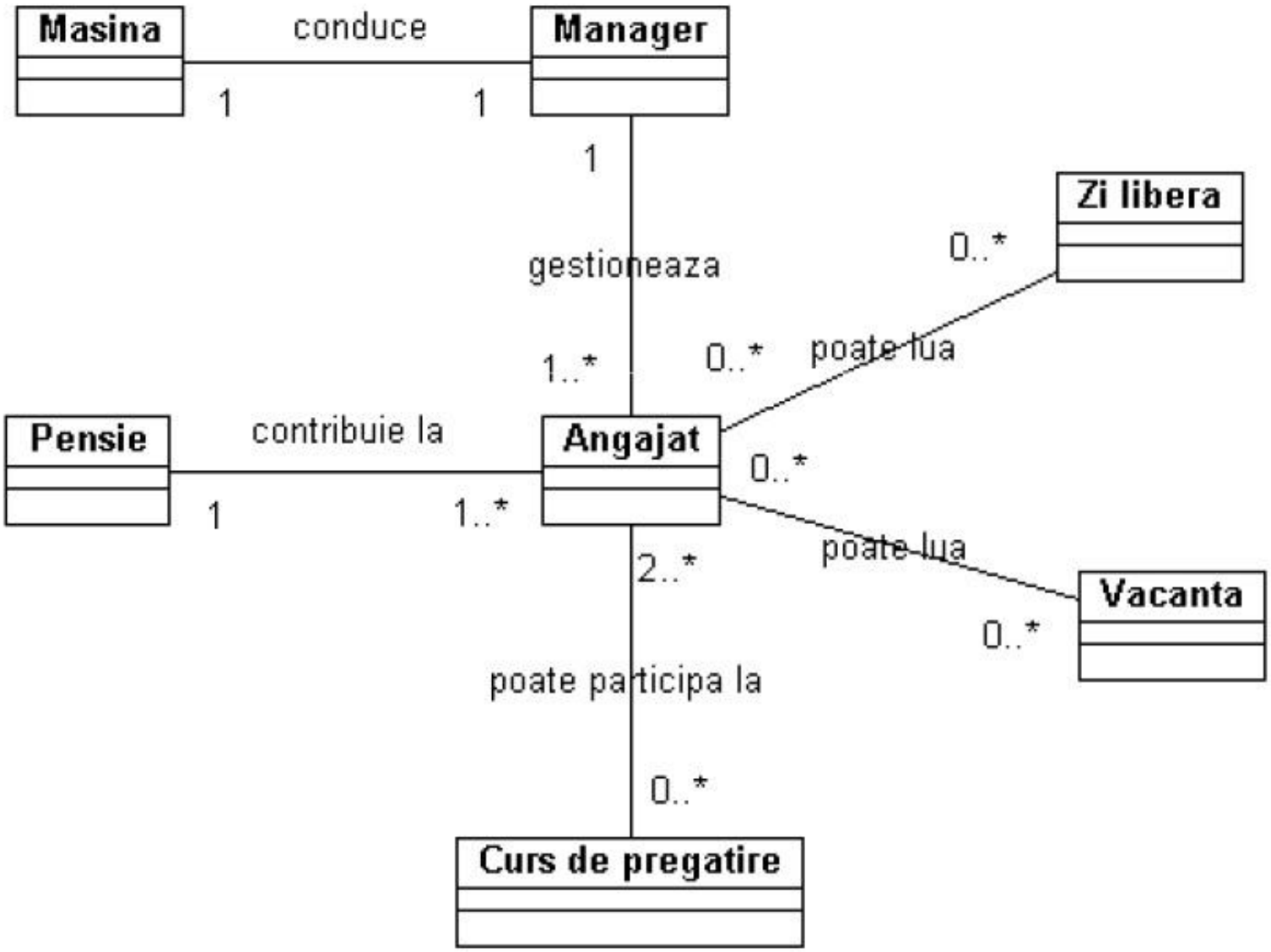


Diagrame de clase – Asocierea

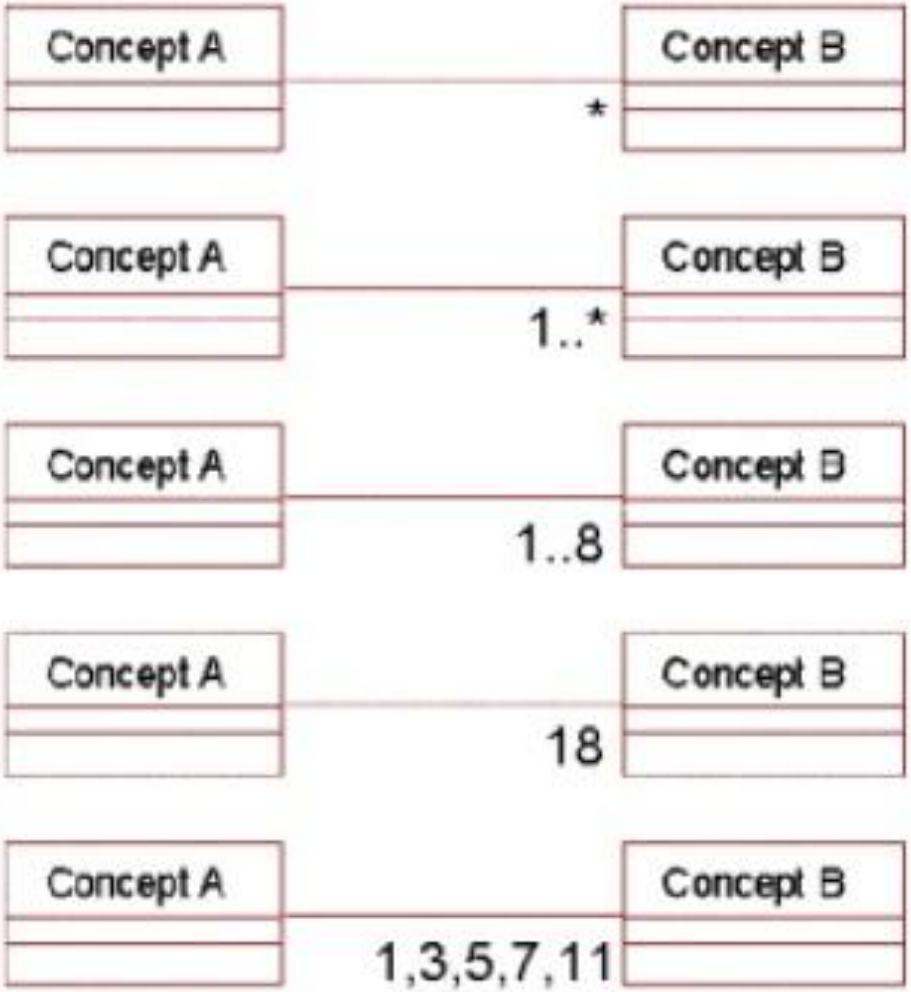
- O clasa poate avea unul sau mai multe campuri instantiat(e) din celalta clasa
  - Exemplu: clasa *BlogAccount* poate contine instante ale clasei *BlogEntry*



Diagrame de clase – Asocierea multipla

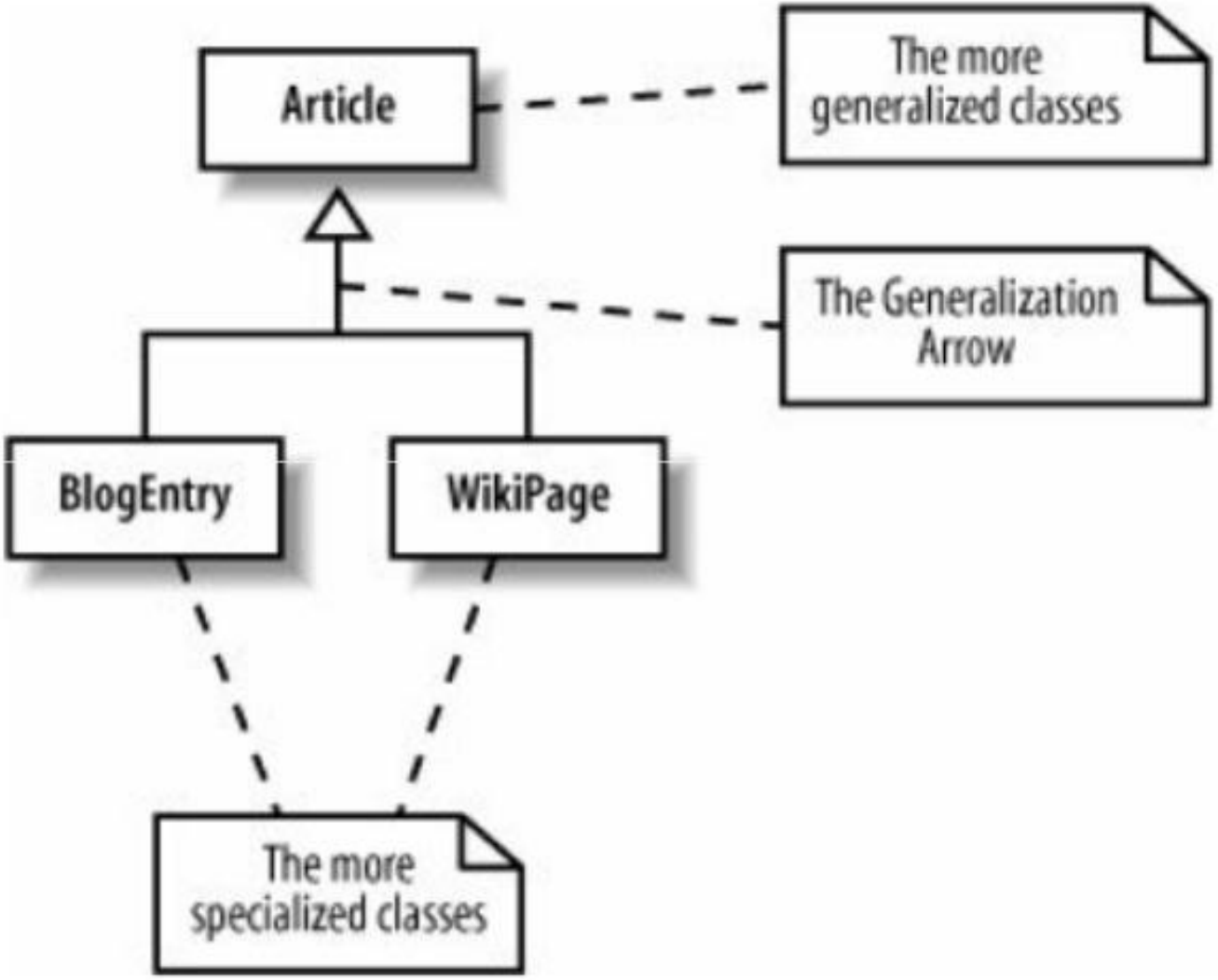


Diagrame de clase – Asocierea multipla (cardinalitatea)



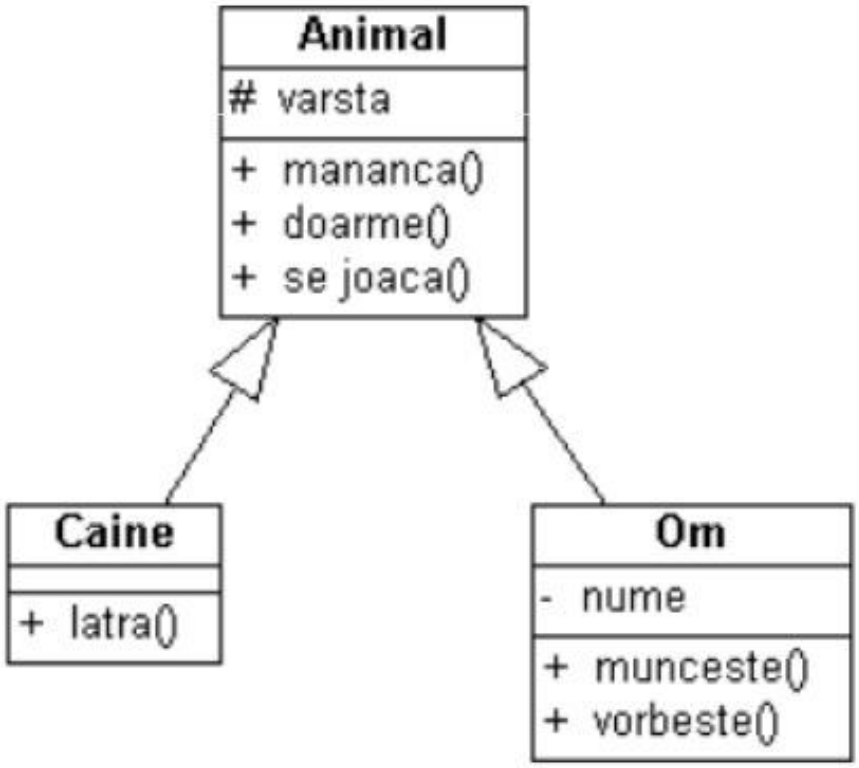
Diagrame de clase – Mostenirea (Generalizarea)

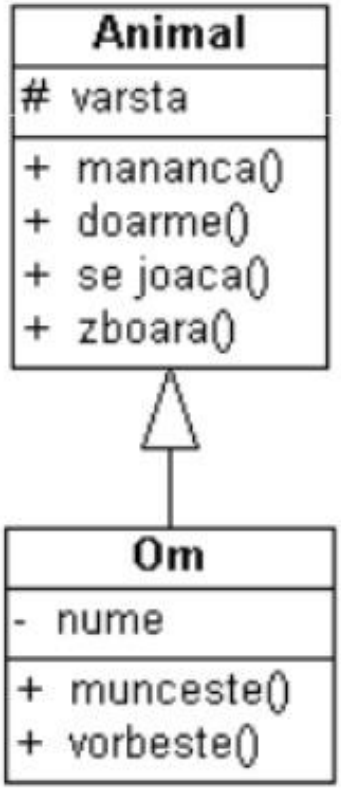
- Este o relatie de tip **ESTE-UN / ESTE-O**



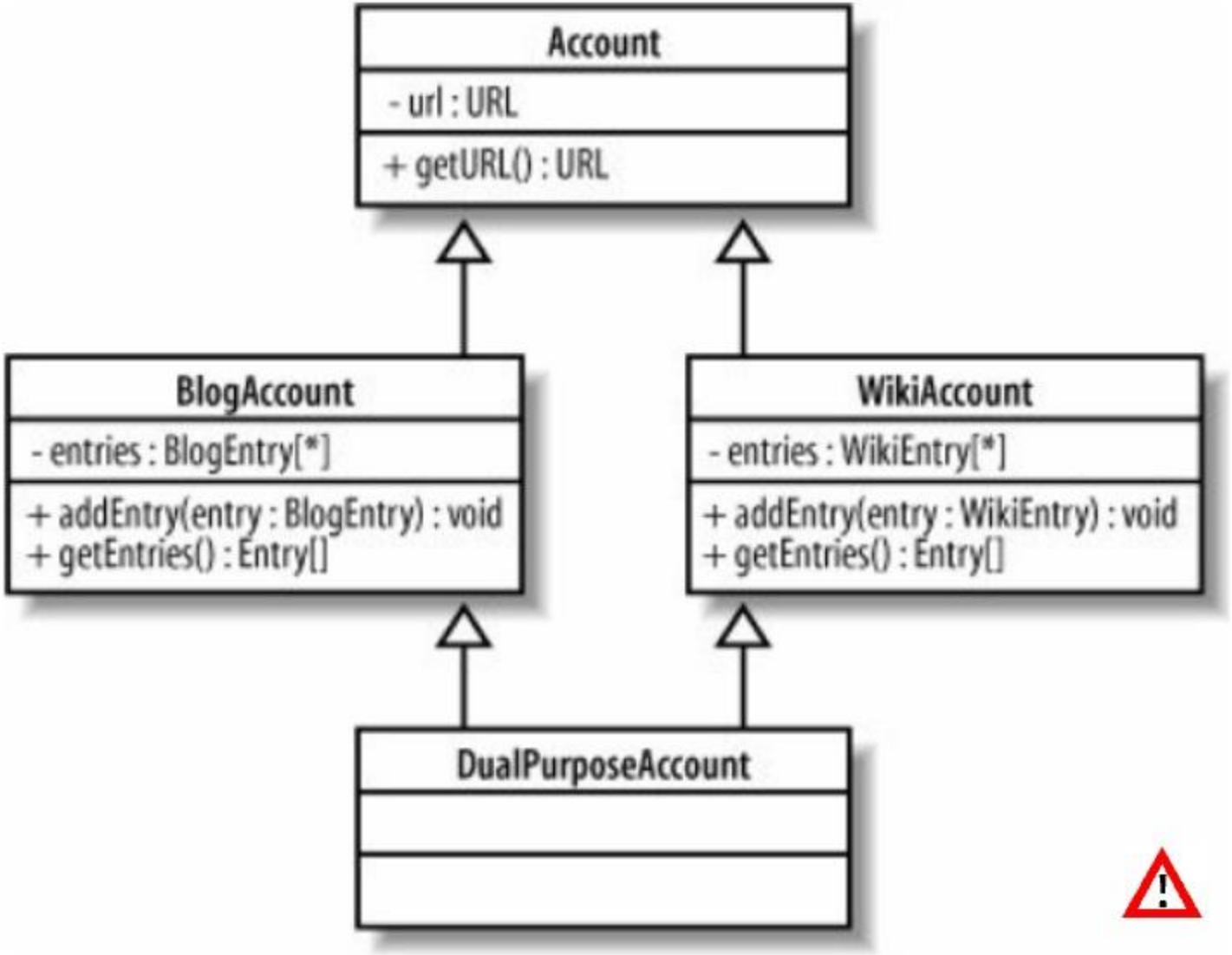
Diagrame de clase – Mostenirea (Generalizarea)

- Toate definitiile clasei de bază trebuie să se aplice tuturor claselor derivate

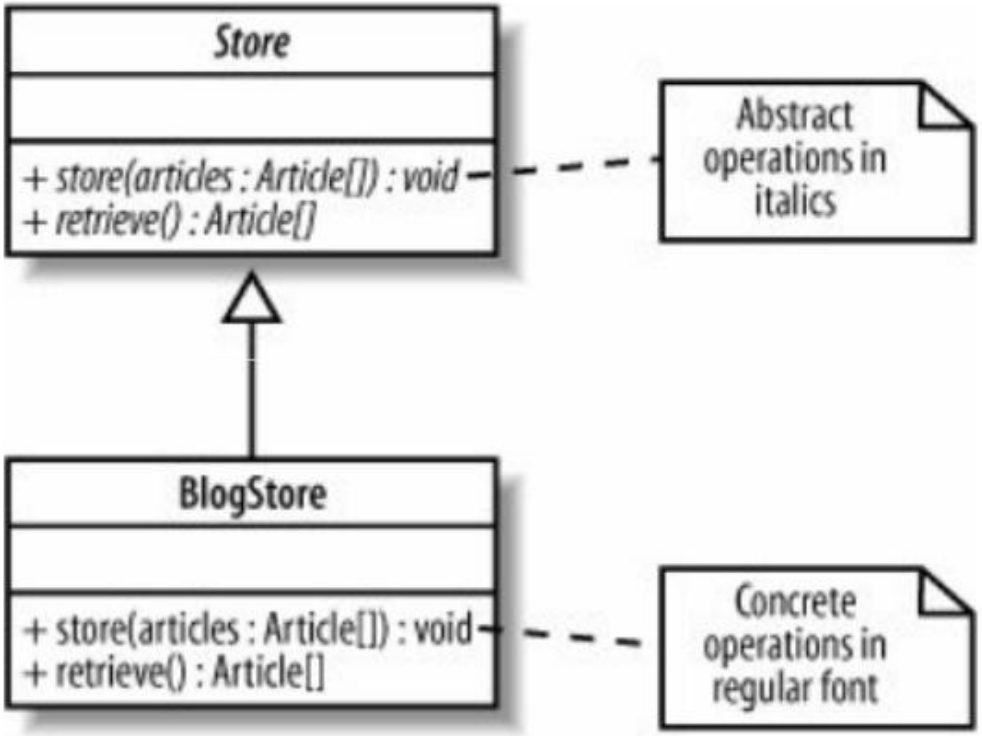


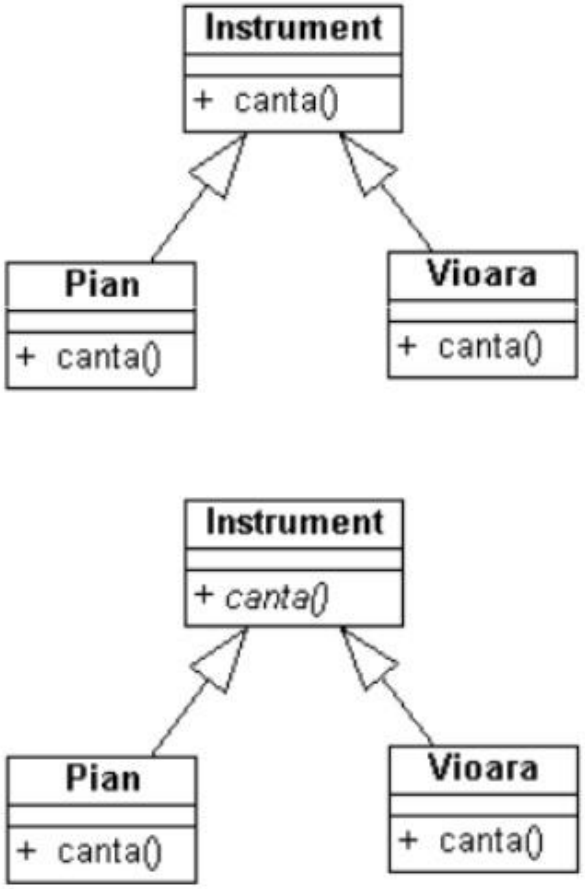


Diagrame de clase – Mostenirea multipla



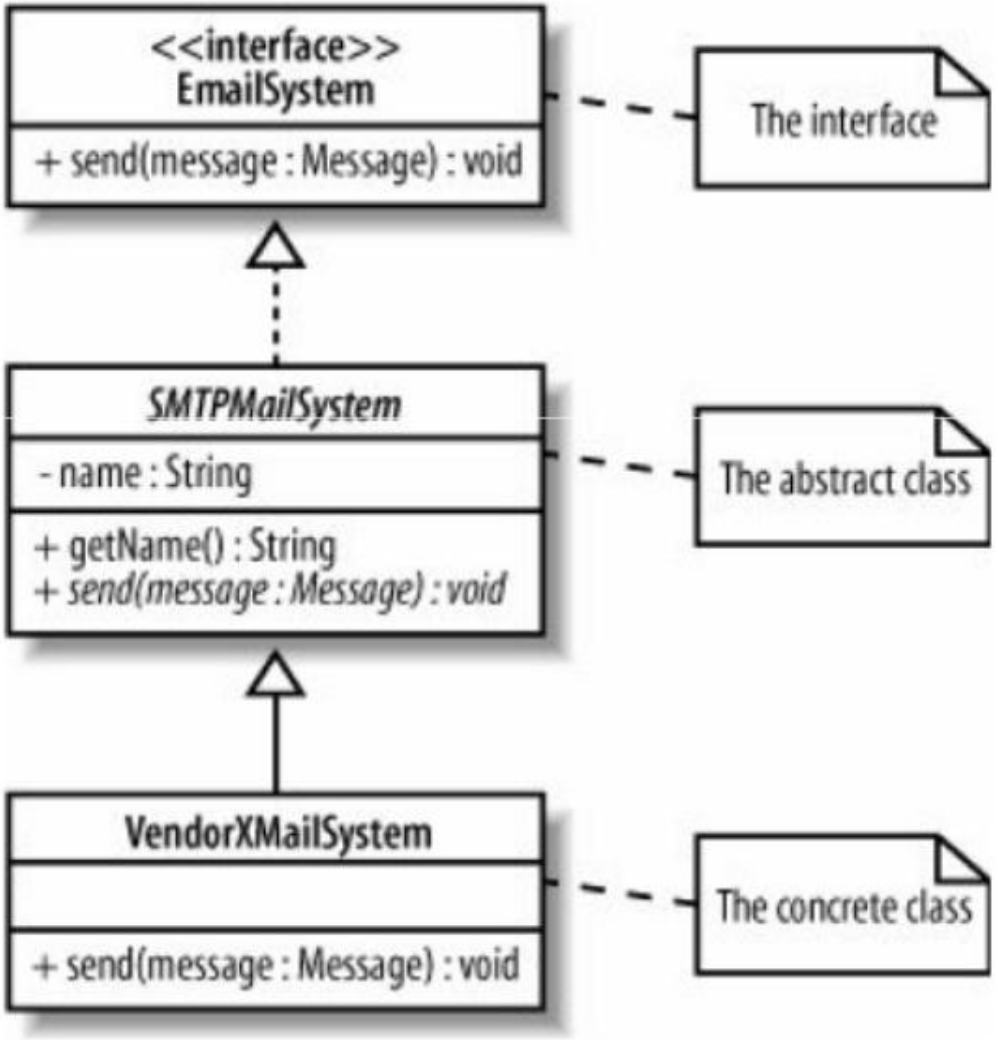
Diagrame de clase – Clase si operatii abstracte

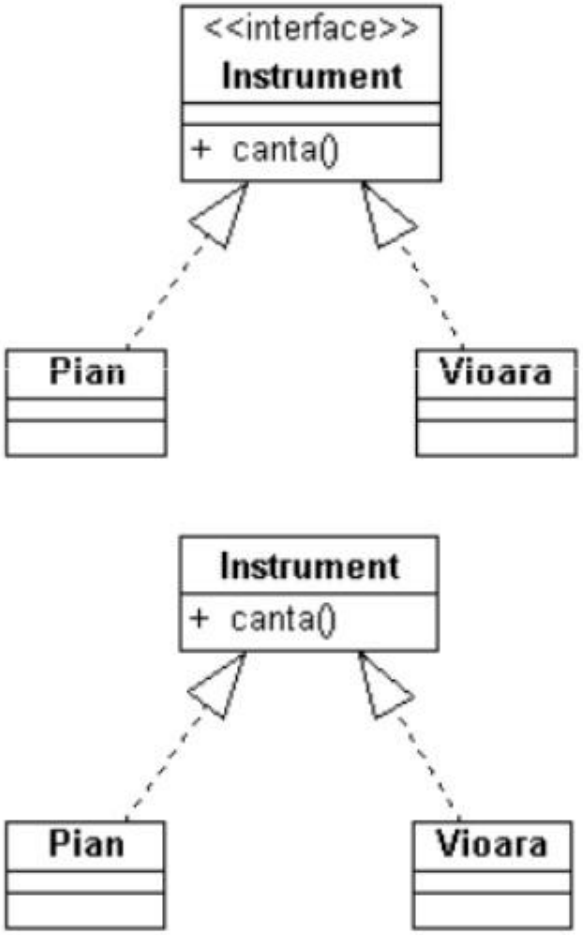




Diagrame de clase – Interfete

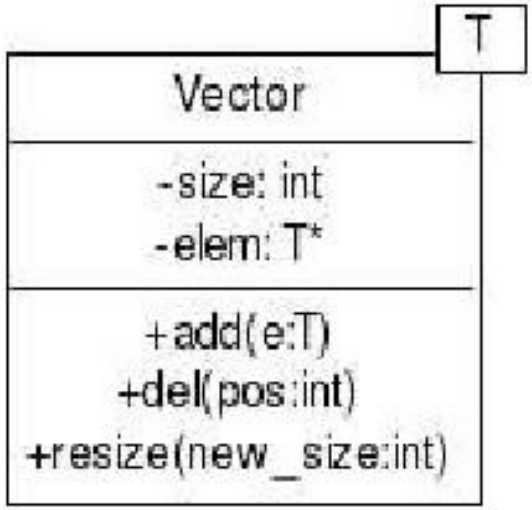
- Specifica o multime de operatii, dar nu mentioneaza structura interna sau implementarea acestora.
- O interfata poate implementata de mai multe clase





Diagrame de clase – Template-uri

- Este o clasa care are unul sau mai multi parametri formali
- Defineste o familie de clase (dand valori parametrilor formali)
- De obicei parametrii reprezinta tipuri ale atributelor



Recomandari

- Nu introduceti prea multe informatii în diagramă
- Ignorati atributele si operatiile necritice
- Aratati într-o diagramă numai clasele relevante pentru un caz de utilizare
- Nu includeti clasele sistem (string, Hashtable etc.)
- Nu includeti prea devreme informatii despre implementare (navigabilitate, vizibilitate)