Diagrama de Clase

Class Diagram

Obiecte

- **Obiect –** reprezentarea informatică a unei entități reale: loc, persoană, eveniment, lucru.
- **Exemple:** câine, bicicletă, munte.

- Obiectele au:
  - **Proprietăți / atribute (stări):** câinele are nume, rasă, culoare; bicicleta are culoare, viteză.
  - **Metode / comportament:** câinele latră, dă din coadă; bicicleta poate schimba viteza.

Clase

- **Clasa –** un grup de obiecte cu aceeasi structură (atribute) si comportament (metode).
- Obiectele sunt instanțe ale unei clase.

- **Exemplu:** Bicicleta unei persoane este o instanță a clasei *biciclete* care grupează toate bicicletele fabricate după aceeași schemă.

Moștenirea

- Definirea unei clase poate fi generală sau mai specifică.
- De exemplu, pe lângă clasa bicicletă putem avea clasele *mountain-bike* și *city-bike*. Acestea sunt clase derivate ale clasei bicicletă – practic sunt tot biciclete (au toate atributele si metodele clasei bicicletă) dar au și ceva în plus.

- Preluarea atributelor unei clase de către o altă clasă (derivată) poartă numele de *moștenire*.

Utilizarea claselor în UML

- Modelarea conceptuală (numită și „modelarea domeniului”) este activitatea de identificare a conceptelor importante pentru sistem.
- În tehnica de programare orientată obiect, modelarea conceptuală se realizează prin diagrama claselor.
- O diagramă de clasă arată cum diferitele entități sunt relaționate între ele, scopul fiind acela de a descrie structura statică a sistemului care este modelat.
- Diagrama claselor furnizează structura codului care va fi scris.

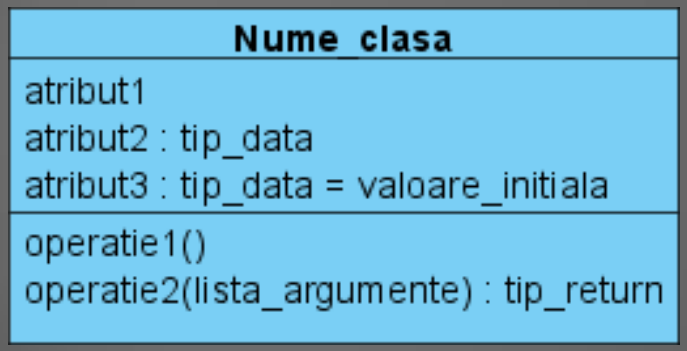
Identificare

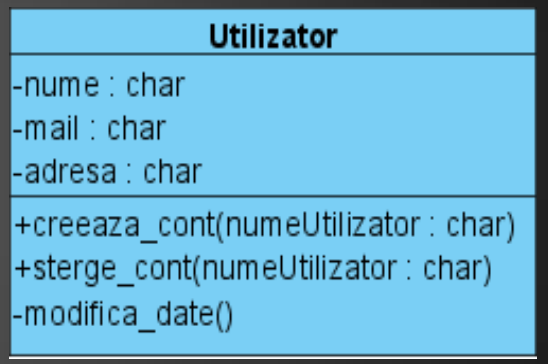
- Pentru a crea o diagramă de clase, acestea trebuie să fie identificate și descrise;
- Identificarea claselor se face răspunzând la următoarele întrebări:
  - Avem un tip de informație care trebuie depozitată sau analizată?
  - Avem sisteme externe înglobate în activitatea sistemului modelat?
  - Există dispozitive pe care sistemul trebuie să le mânuiască?
  - Există biblioteci sau componente din alte proiecte care vor fi folosite în sistem?
  - Actorii folosiți (din use-cases) au un rol în sistem astfel încât vor trebui implementați ca fiind clase?

Reprezentare

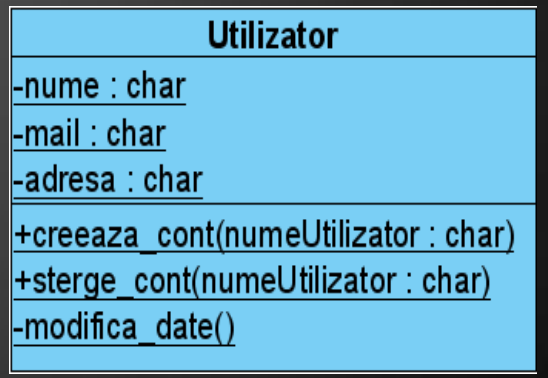
- Notația UML pentru o clasă este un dreptunghi împărțit în trei părți: numele clasei, atributele și operațiile.

Exemplu:





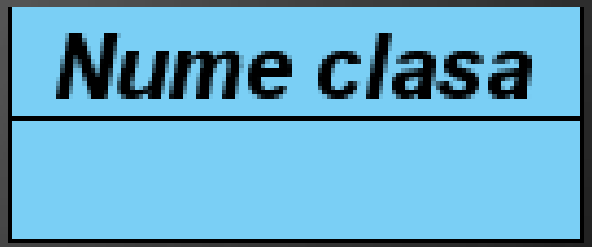
- Atât operațiile cât și atributele pot fi statice (afișate subliniat), deci nu vor aparține unei instanțe a clasei din care provin ci vor aparține chiar clasei, fiecare din instanțele ei având acces la acestea.



Numele

- Primul compartiment al dreptunghiului clasei (numele) este de obicei un substantiv care trebuie să fie derivat din domeniul problemei analizate și nu trebuie sa fie ambiguu.

- O clasă *abstractă* va avea numele scris cu caractere italice.



Atributele

- În partea mediană sunt atributele. Un atribut are un tip care poate fi:
  - primitiv (integer, real, string, byte, char etc.);
  - o instanță a unei clase construite de utilizator.

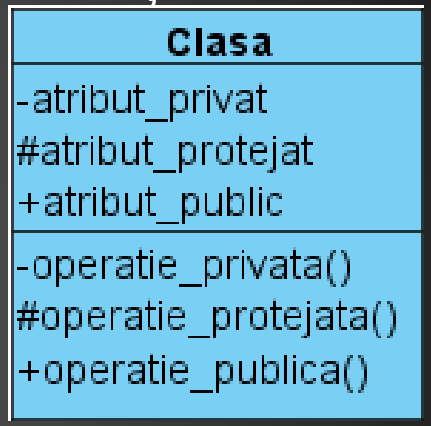
- **Observație:** Aceste tipuri primitive nu aparțin limbajului UML ci unui anumit limbaj de programare folosit în faza de implementare.

Operațiile

- Operațiile (metodele) se trec în partea de jos a clasei și sunt folosite pentru a manipula atributele sau a performa alte acțiuni.
- Descriu serviciile oferite de o clasă, de aceea pot fi văzute ca fiind interfața clasei respective.
- O operație are un ***tip returnat***, un ***nume*** și zero sau mai mulți ***parametri***.

Vizibilitatea atributelor și metodelor

- În cadrul diagramelor de clase pot fi specificați declaratorii de acces (vizibilitatea).



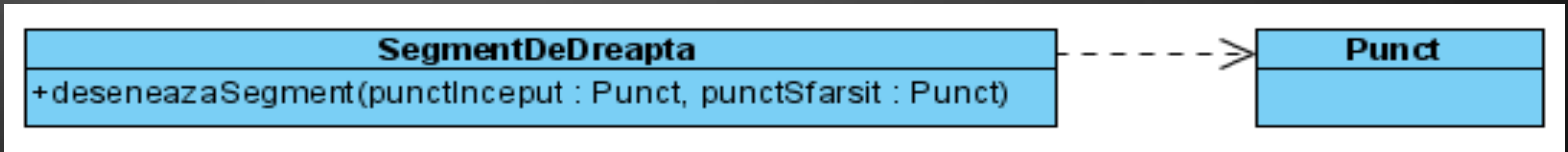
- Aceștia sunt:
  - **Public (+)**: accepta referirea din alte clase;
  - **Private (-)**: nu acceptă referirea din alte clase;
  - **Protected (#)**: acceptă referirea numai din clasele fiu (este folosit împreună cu o relație de generalizare sau specializare).

Relații într-o diagramă de clase - Dependența

- Apare când o clasă folosește pentru scurt timp o altă clasă.
- Apare între două elemente dintre care unul este unul *independent* și altul *dependent*. Orice modificare a elementului independent va fi reflectată și asupra elementului dependent.
- Putem avea o relație de dependență de exemplu în cazul în care o clasă folosește ca parametru un obiect al altei clase, sau o clasă accesează un obiect global al altei clase, sau o operație a unei clase este apelată într-o altă clasă.

Exemplu:

- În acest exemplu există o dependență între clasa ***Punct*** și clasa ***SegmentDeDreaptă*** în sensul în care clasa ***SegmentDeDreapta*** trebuie să "știe" despre clasa ***Punct*** pentru execuția metodei ***deseneazaSegment***.

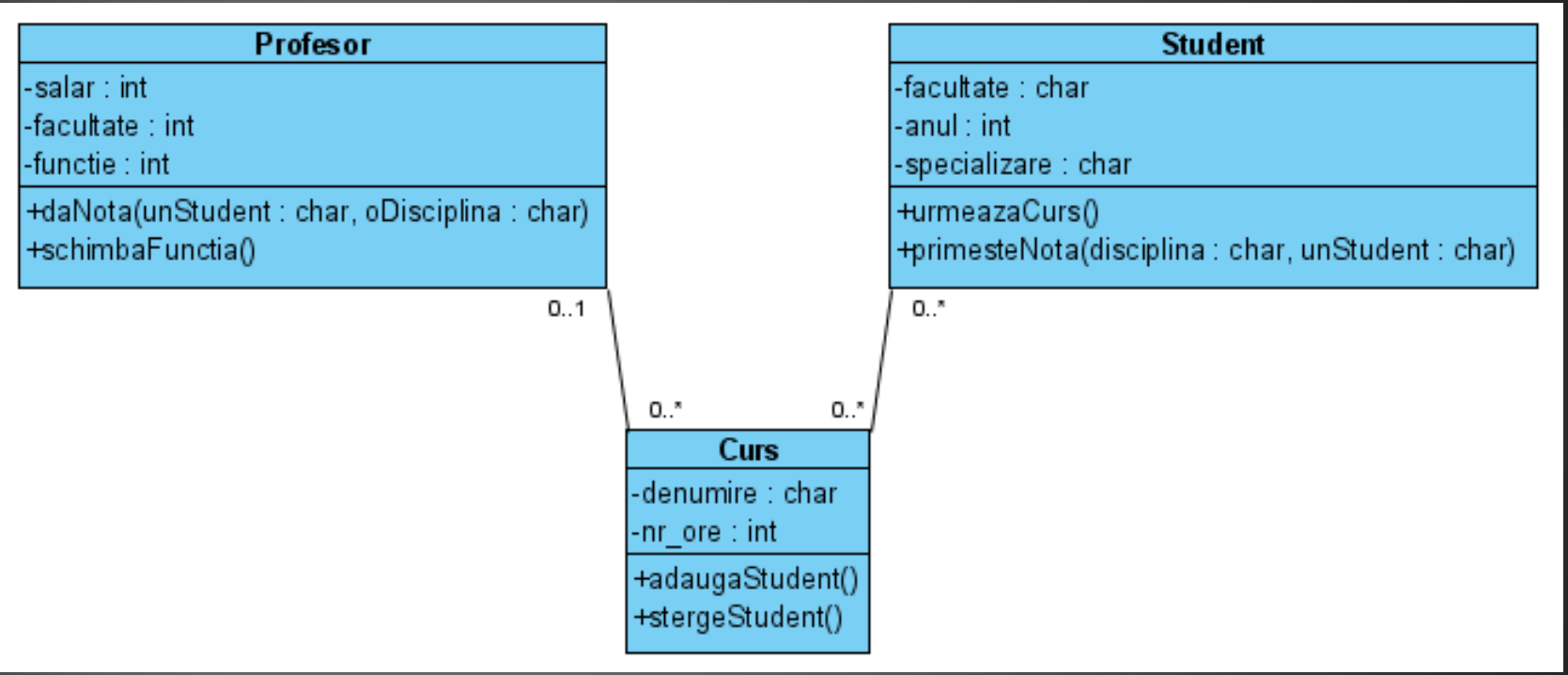


Asocierea

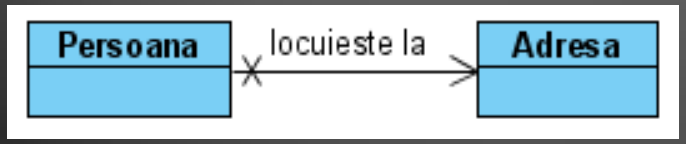
- **Asocierea**: legătură (conexiune) între două clase (relație Și între obiecte, instanțe ale celor două clase) ce permite claselor respective să comunice între ele.
- Pot exista asocieri **unidirectionale** sau **bi-directionale** (indică dacă fiecare clasă transmite mesaje celeilalte sau doar una poate transmite mesaje).
- Poate fi și recursivă.

Exemple

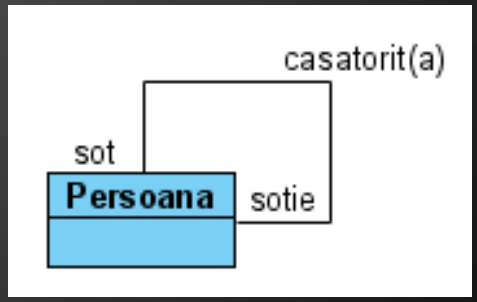
1. Asociere bidirecţională:



2. Asociere unidirecţională:

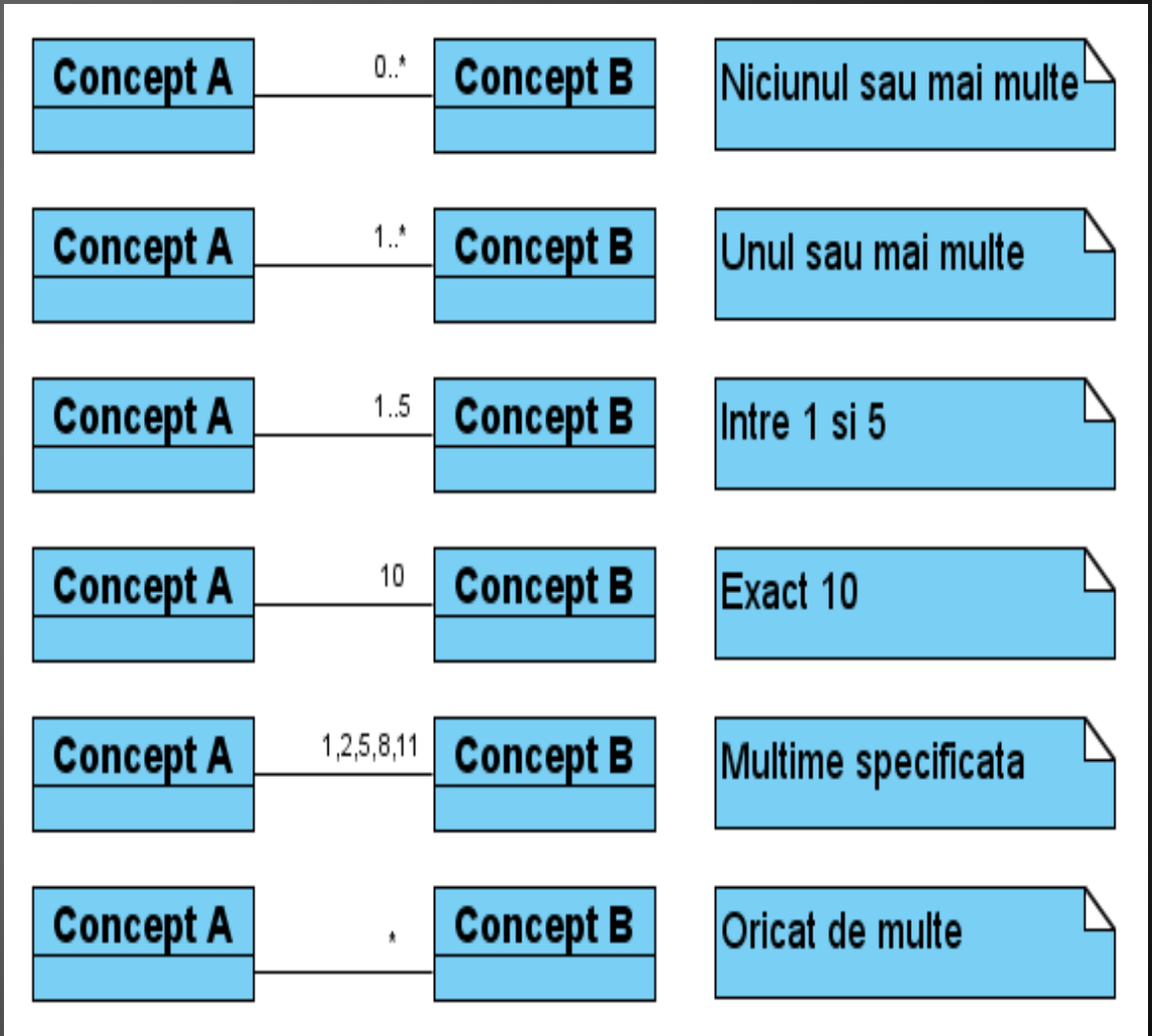


3. Asociere recursivă



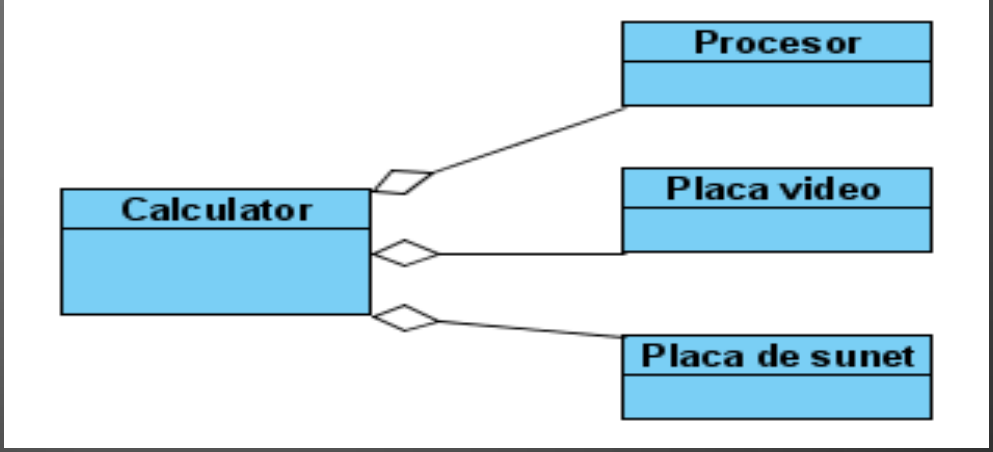
Cardinalitatea

- Asocierea se reprezintă printr-o linie simplă.
- Fiecare asociere are la capete un indicator de multiplicitate, care stabilește modalitatea de relaționare a obiectelor de acel tip (în genul relațiilor între mulțimi).



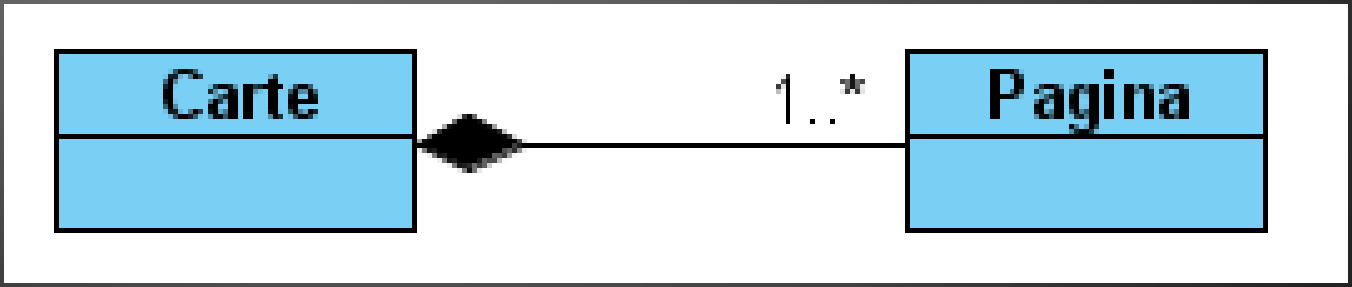
Agregarea

- Este o formă specială de asociere a claselor, utilizată în cazul în care relația dintre cele două clase este de tipul *parte din întreg*.
- De exemplu, un calculator este o agregare între procesor, placă video, placă de sunet etc.



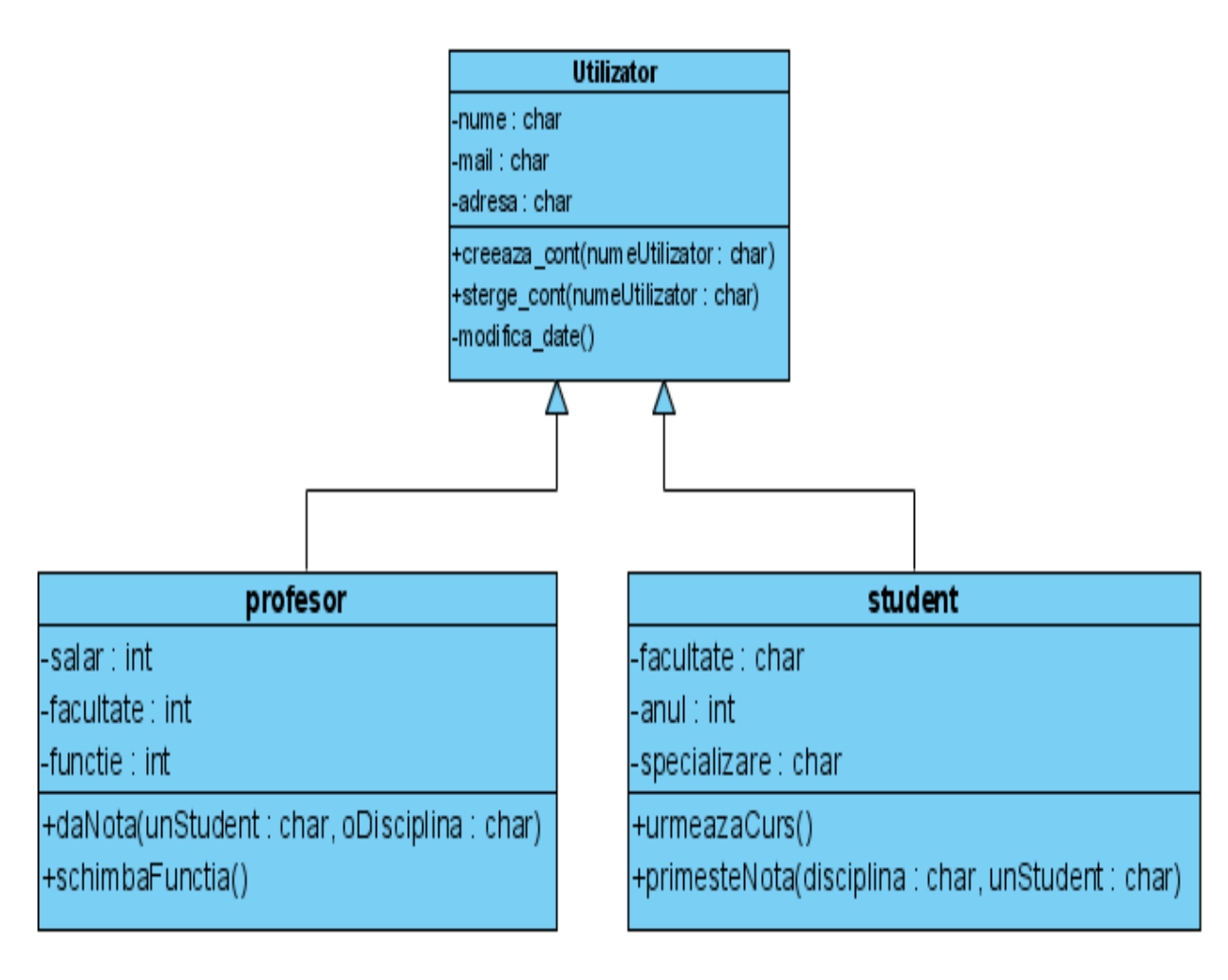
Compoziția

- Compunerea este un concept similar cu agregarea, însă mai puternic deoarece implică faptul că întregul nu poate exista fără părți.
- În exemplul de agregare anterior, dacă se înlătură placa de sunet, calculatorul rămâne tot calculator. Însă o carte nu poate exista fără pagini; o carte este *compusă* din pagini.

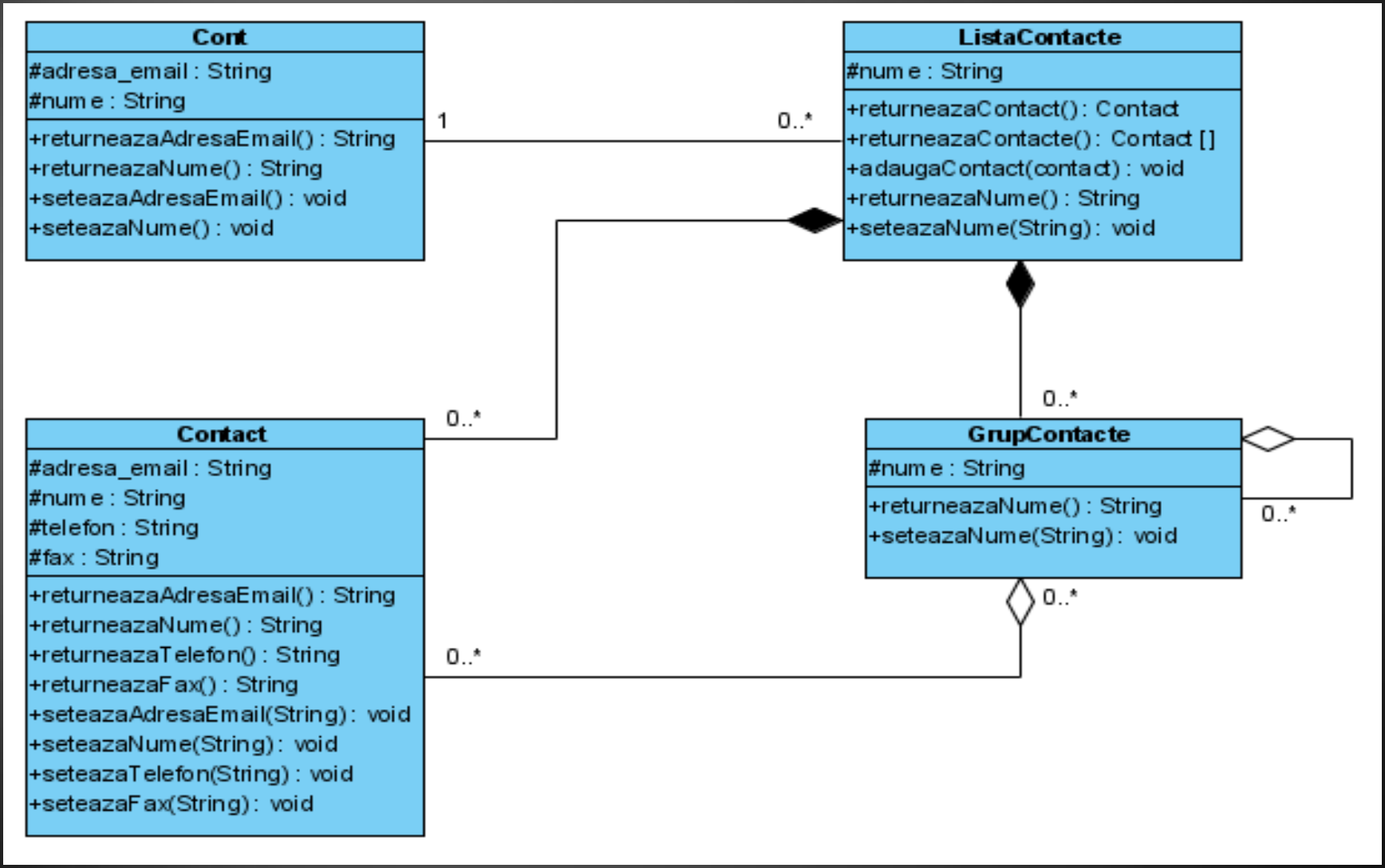


Generalizarea

- Relație între o clasă și subclasele sale, este prezentă ori de câte ori se semnalează de-a lungul unei ierarhii proprietăți comune sau operații ce evidențiază comportament comun.



Exemplu diagramă de clase



Temă de laborator

- O societate comercială are diferiți angajați care lucrează. Aceștia se află grupați în echipe iar fiecare dintre acestea sunt coordonate de către un șef de echipă. Aplicația care gestionează personalul acestei societăți comerciale este intreresată de gestiunea echipelor dar și a persoanelor angajate. Pentru o persoană este relevantă ocupația, numele, prenumele (și alte date personale) cât și echipa din care face parte. Echipele sunt importante în managementul de personal iar datele relevante pentru o echipă sunt specializarea acestei echipe, numele ei și data creării.
- Să se modeleze utilizând o diagramă de clasă aplicația de mai sus privind gestiunea personalului din întreprindere.