M. Caramihai, © 2020

PROGRAMAREA ORIENTATA OBIECT

CURS 2

Conceptele de baza ale POO

Introducere

Descompunerea sistemelor de mari dimensiuni:

1. algoritmic: pune in evidenta ordinea evenimentelor
2. Prin intermediul obiectelor: pune in evidenta elementele ce actioneaza / sunt supuse unor actiuni

Diferente fata de abordarea clasica:

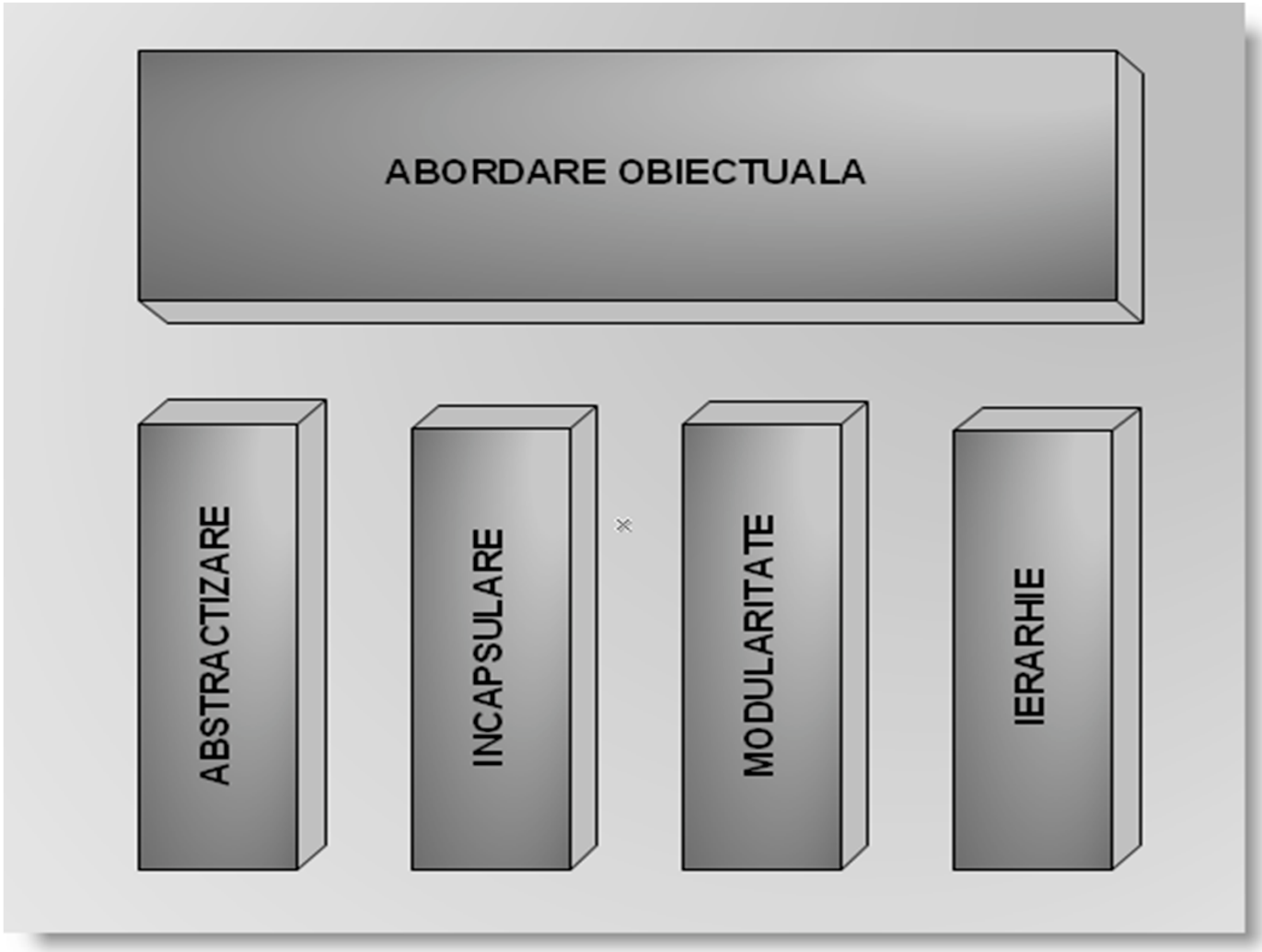
- POO se bazeaza pe obiecte si nu pe algoritmi
- Fiecare obiect este o instanta a unei clase
- Clasele pot fi legate intre ele prin relatii de mostenire

Obs: daca una din aceste caracteristici lipseste, nu este vorba de programare OO

Stiluri de programare

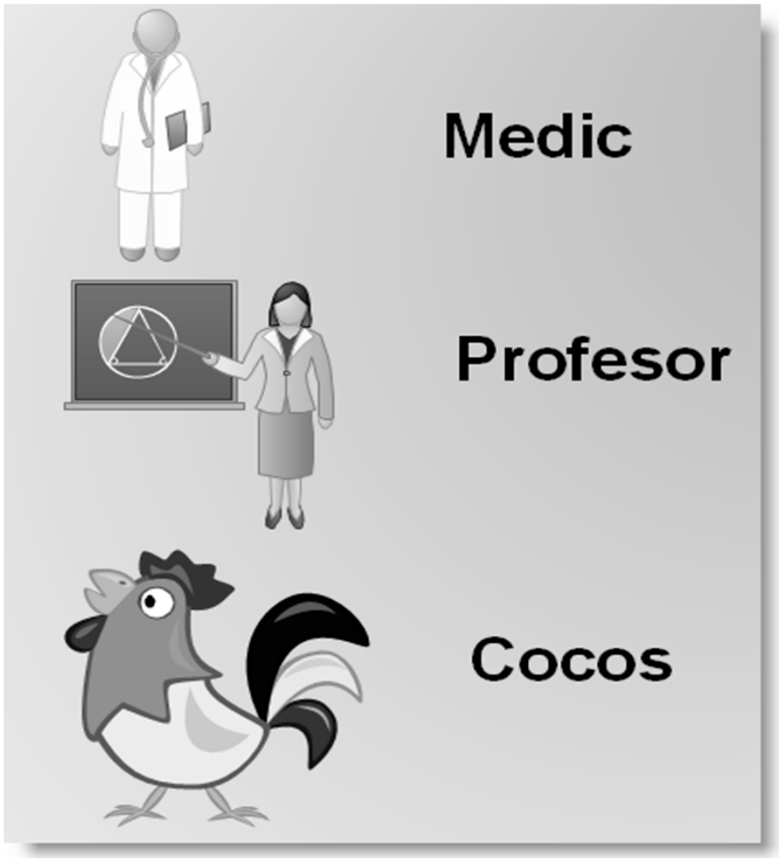
1. Orientatate catre proceduri → algoritmi
2. Orientatate catre obiecte → clase si obiecte
3. Orientatate catre logica → scop (calcul predicativ)
4. Orientatate catre reguli → reguli de tip “if – then”
5. Orientatate catre constrangeri → relatii de invarianta

Principii de baza ale abordarii obiectuale



Ce este abstractizarea (1) ?

- Exemplul atlasului: → scara de reprezentare a marimilor
- Definitie: procedura de mascare a detaliilor unui obiect in scopul accentuarii / punerii in evidenta a altora specte / detalii / structuri
- Nivele de abstractizare:
  1. Cel mai de sus: comunitatea obiectelor ce trebuie sa interactioneze unele cu altele in vederea indeplinirii unui scop comun



Ce este abstractizarea (2) ?

2. Valabil in anumite limbaje OO: grup de obiecte ce lucreaza impreuna si care sunt grupate in *unit*-uri, d.e. *package* (Java), *name spaces* (C++) si *units* (Delphi)

3. Nivelul de interactiune intre 2 obiecte (interactiune *client / server, C/S*). In acest caz exista 2 sub-nivele de abstractizare:
   - Abstractizare *client* → *server*
   - Abstractizare *server* → *client*

Ce este abstractizarea (3) ?

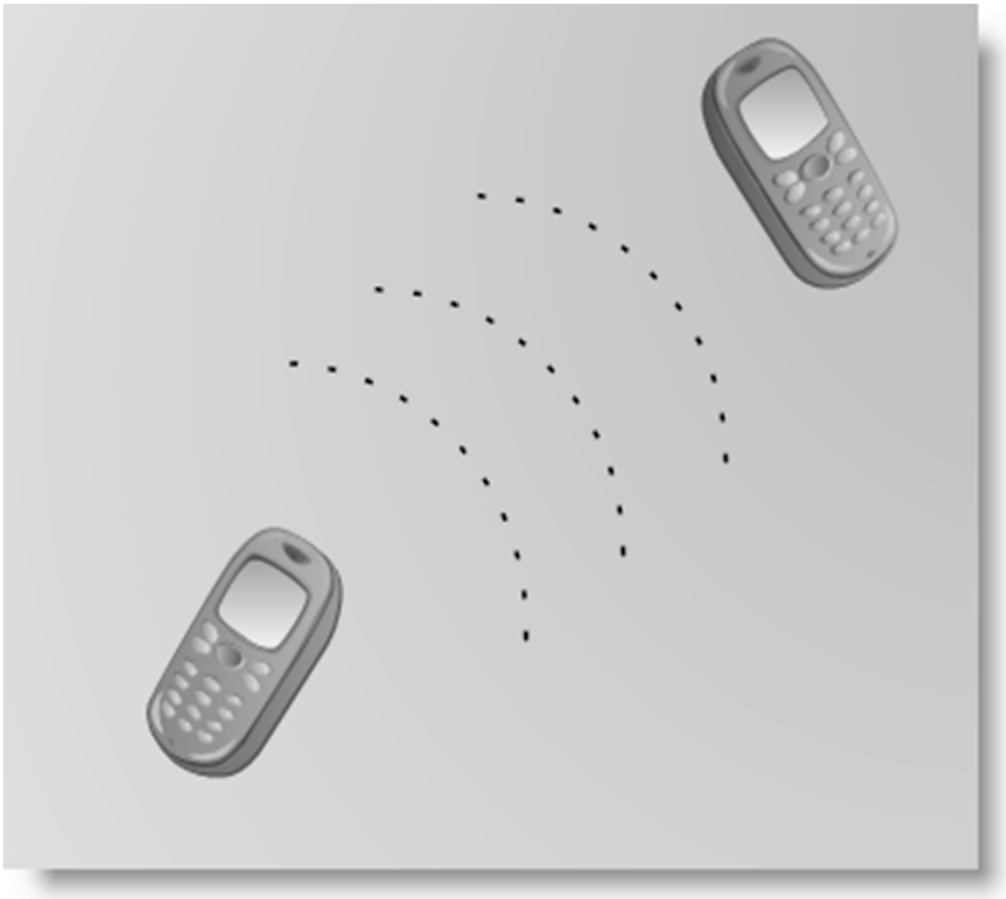
- Tipuri de abstractizari: entitate / actiune / ms virtuala / incident
- Abstractizarea este legata de notiunea de "invarianta"; ruperea unei invariante conduce la lipsa vizibilitatii unor obiecte cu altele
- Orice abstractizare are proprietati
  - Statice, de ex nume fisier
  - Dinamice, de ex valoarea unei prop statice → un nume de fisier poate fi schimbat

- **Consecinta: *abstractizarea* si *incapsularea* sunt concept complementare, i.e. pentru ca abstractizarea sa functioneze, trebuie sa existe incapsulare**

Ce este incapsularea ?

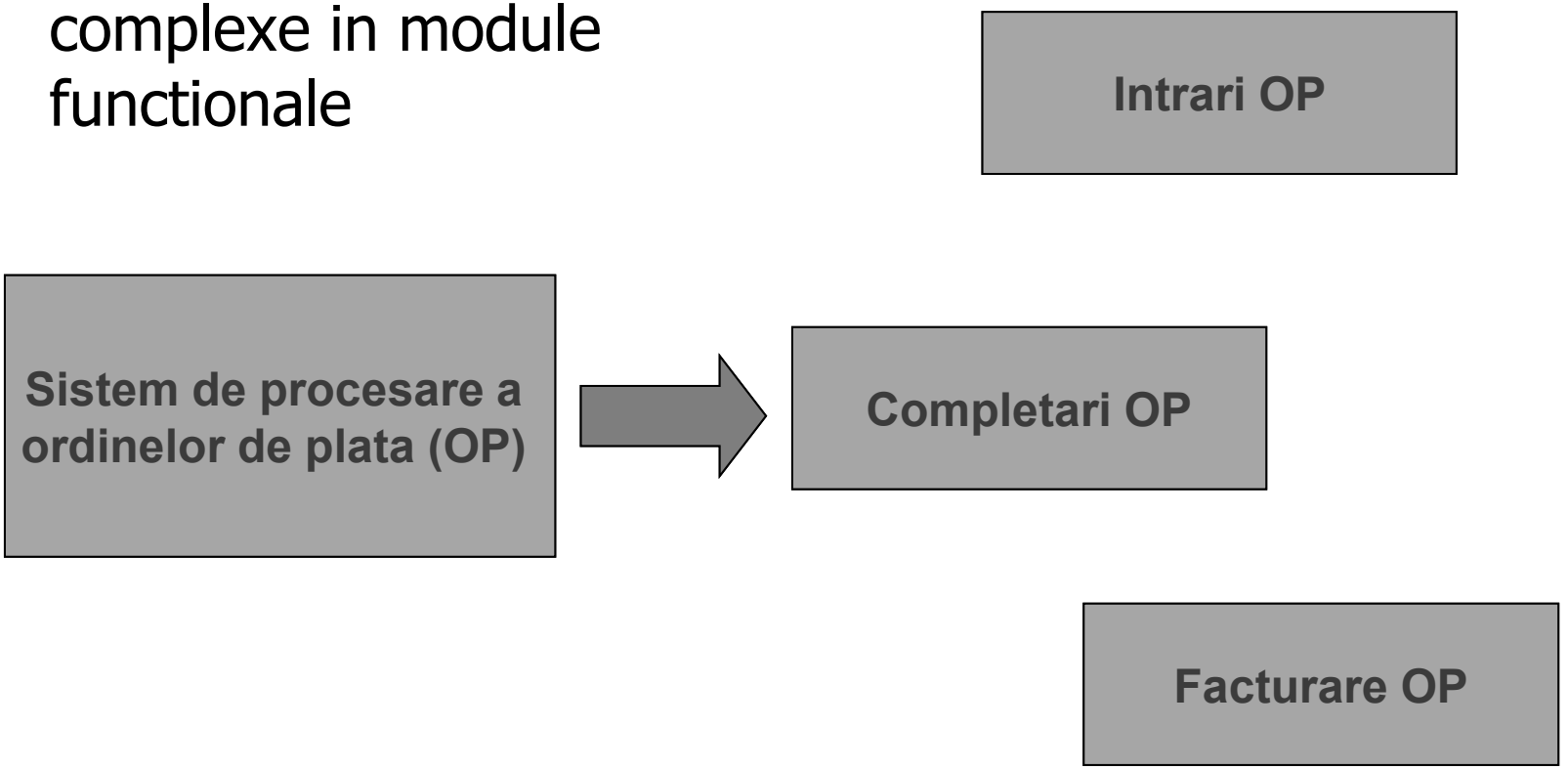
Ascunde implementarea fata de clienti

→ Clientii depind de interfata



Ce este modularitatea ?

- Spargerea elementelor complexe in module functionale

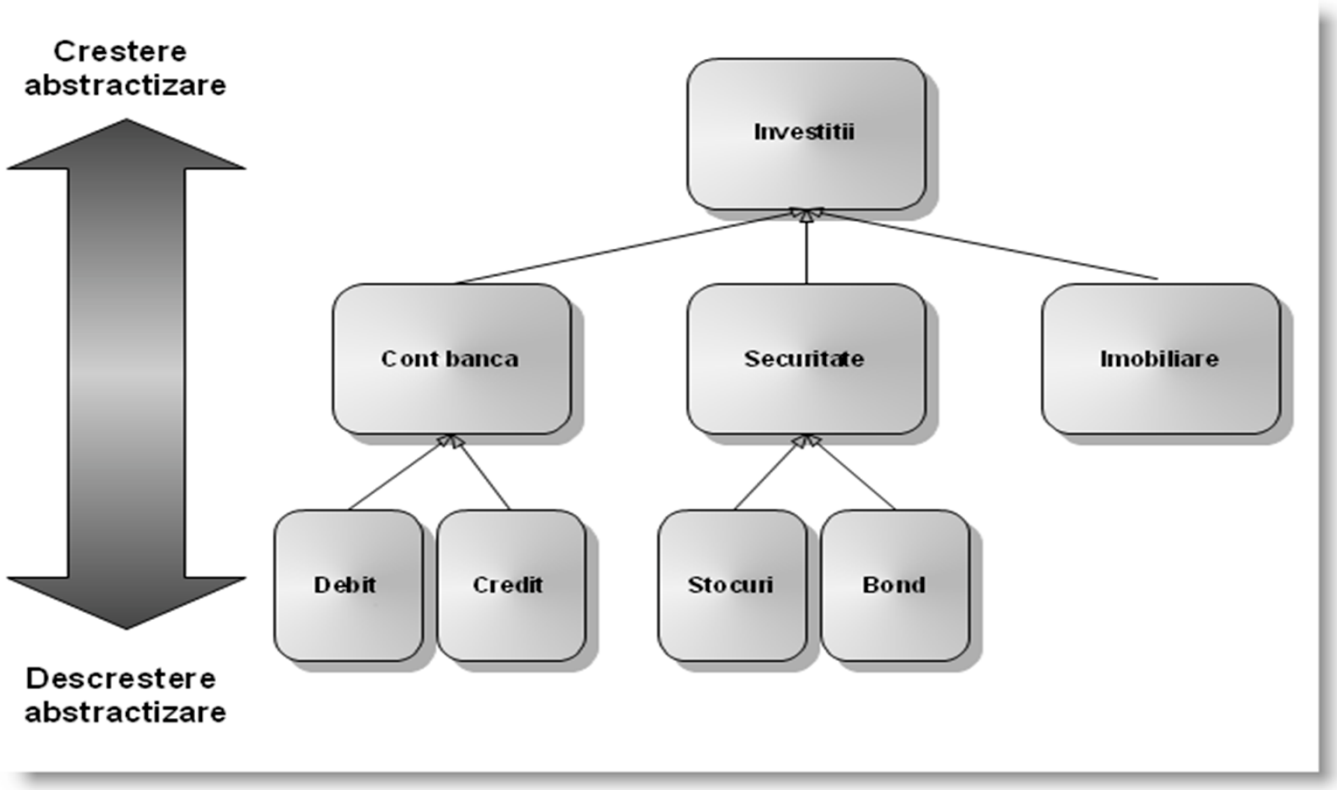


Caracteristici modularitate

- Exista programe (d.e. Smalltalk) ce nu contin module (i.e. clasa reprez. singura unitate fizica de desompunere)
- Scopul modularizarii: reducerea costurilor prin accesarea independenta a fiecarui modul

→ In general clasa + obiect se regasesc in module

Ce este ierarhia ?



Ierarhia reprezinta o ordonare a abstractizarii.

Concepte de baza ale POO

- Obiect
- Clasa
- Atribut
- Operatie
- Componenta
- Mostenire
- Incapsularea
- Polimorfism

Obiecte (1)

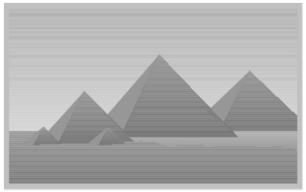
- Un obiect reprezinta o entitate (fizica, conceptuala sau soft)

1. Entitate fizica

2. Entitate conceptuala

3. Entitate software

Peste



Piramide

Componenta (subsistem)

Obiecte (2)

- Definitie formala: un obiect este un concept (real sau abstract) cu *frontiere* bine definite si posibilitate de actiune.
- Se caracterizeaza prin:
  - Evolutie – influentata de istoricitatea obiectului (d.e. automatul pentru bauturi); reprez modul in care obiectele actioneaza / reactioneaza in termini de modificare de stare si schimb de mesaje
  - Stare: orice obiect are o stare, i.e. ocupa un spatiu fizic de memorie
  - Identitate unica: “propietatea unui obiect ce il face distinct fata de un alt obiect” (Copeland)

Starea unui obiect

- Starea unui obiect reprezinta una din conditiile posibile de existenta ale acestuia
- In mod normal, starea unui obiect se schimba in timp
- Este reprezentata de:
  - Valorile atributelor
  - **+**
  - Instantele relatiilor (*relationship instances*)

Sofer

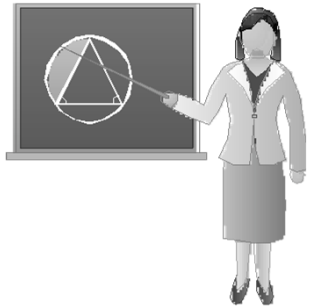
Nume:	Ion Popescu
ID:	123456789
Data angajarii:	01.01.01

Evolutia unui obiect (1)

- Evolutia determina modul in care obiectele (re) actioneaza la solicitarile altor obiecte.
- Evolutia unui obiect este reprezentata de totalitatea raspunsurilor la setul de mesaje primite (i.e. totalitatea operatiilor pe care un obiect le poate realiza)
- *Observatie*: orice obiect are o identitate unica (chiar daca starea sa este identica cu cea a unui alt obiect)

Asignare curs

Conformitate



Sistem de inregistrare

Curs 100 – Geometrie

Evolutia unui obiect (2)

- Operatii
  - De modificare: altereaza starea unui obiect
  - De selectie: operatia acceseaza starea unui obiect dar NU o altereaza
  - De iteratie: operatia prin care se permite ca toate partile unui obiect sa poata fi accesate intr'o ordine predefinita

- **Constructor**: crearea unui obiect si/sau initializarea starii acestuia
- **Destructor**: elibereaza starea unui obiect si/sau il distruge

Clase

- Ansamblul obiectelor cu aceeasi semantica, proprietati (atribute) similare sicomportament comun
- Un obiect reprezinta *instanta* a unei clase
- Gruparea obiectelor in clase reprezinta rezultatul unei operatii de abstractizare:
  - Evidentierea caracteristicilor comune
  - Eliminarea elementelor nespecifice

Clasa – exemplu

Masina

- Atribute:
  - Model
  - Serie_motor
  - An_fabricatie
- Operatii:
  - Deplasare
  - Reparare
  - Oprire

Clase & obiecte – exemplu

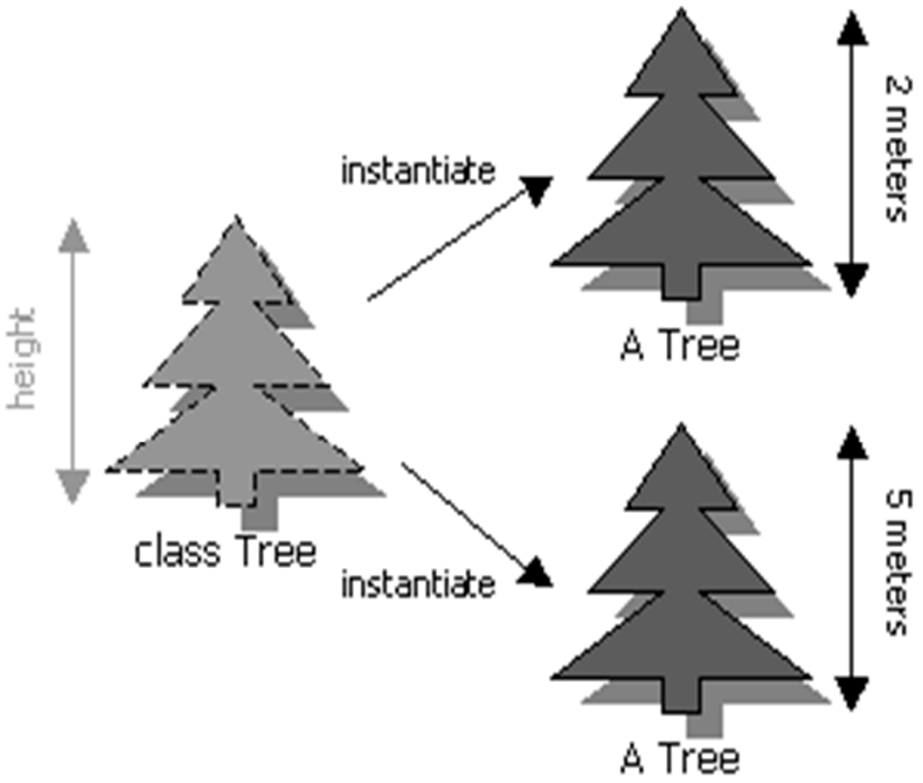


Fig. 1: Instantiating two Trees from the Tree class

Clasa “Arbore”

Obiecte (instantieri ale clasei)

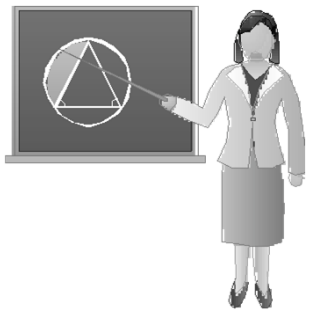
arbori, cu atribut inaltime

Tipuri de clase

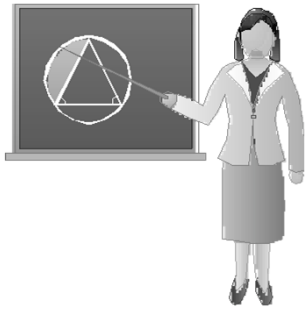
- Clasa abstracta.
  - O clasa *incompleta* care defineste elementele comune a mai multor clase
  - Fara instantiere.
- Clasa concreta
  - O clasa completa
  - Descrie complet un anumit concept
  - Poate avea instantieri

Relatii intre obiecte si clase

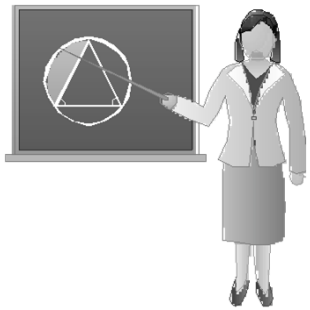
- O clasa reprezinta o definitie abstracta a unuiobiect
  - Defineste structura si evolutia pentru fiece obiect ce apartine clasei respective
  - Poate fi un *template* pentru crearea de obiecte
- Obiectele pot fi grupate in clase



Profesor Popescu



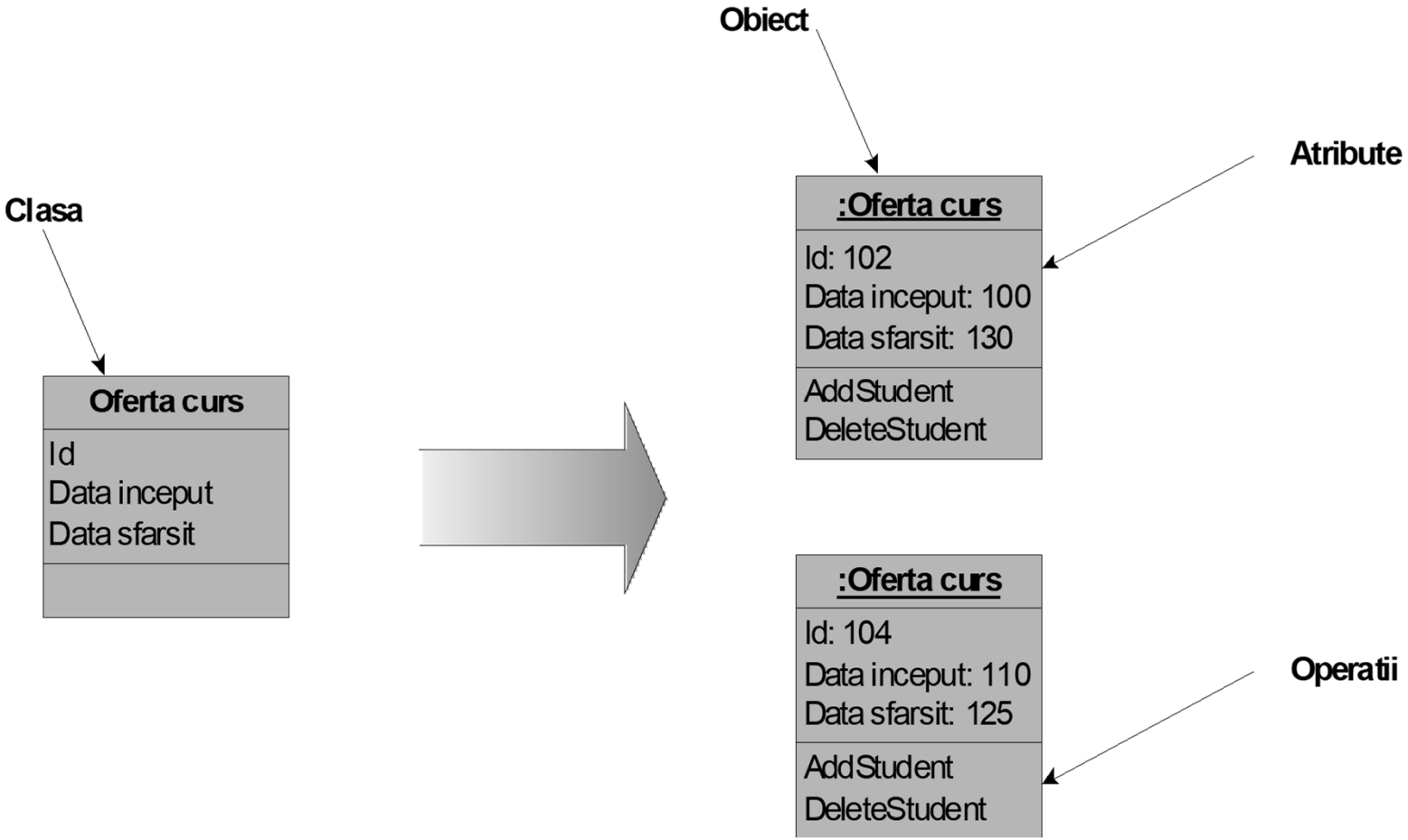
Profesor Ionescu



Profesor Vasilescu

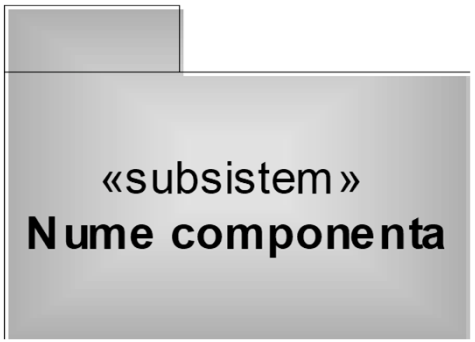
Clasa Profesor

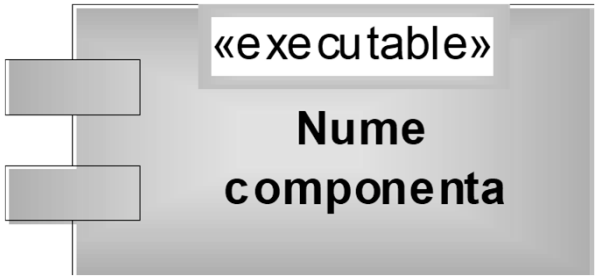
Atribute & operatii



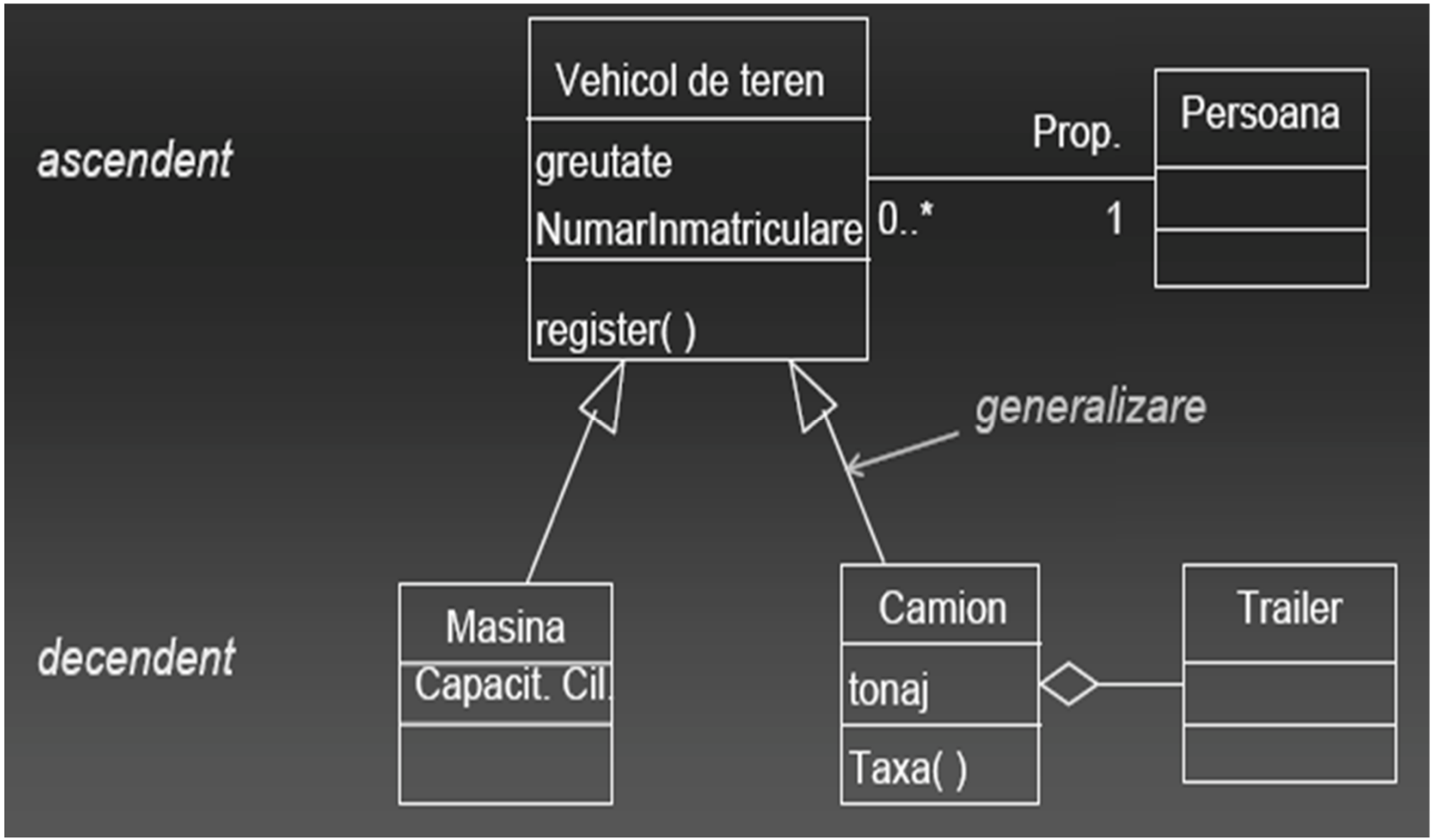
Componenta

- Partea unui sistem ne-triviala si inlocuibila ce indeplineste o functie bine precizata in contextul unei arhitecturi bine definite.
- O componenta poate fi:
  - Cod sursa
  - O structura executabila





Generalizare



Mostenire

- Proprietatea de partajare a atributelor si operatiilor unei / unor clase de catre alta / alte clase in cadrul relatiilor ierarhice, diferentele ramanand neschimbate
- Prin mostenire se pot exprima relatii de:
  - Generalizare
  - Specializare
  - Clasificare
  - Aproximatie
  - Evolutie
- Tipuri de mosteniri:
  - Simple
  - Multiple

Incapsularea

- Gruparea detaliilor de implementare intr'o zona ascunsa, inaccesibila din exterior
- Consecinta: reducerea gradului de interdependenta dintre elemente
- Vizibilitatea claselor este structurata pe trei nivele:
  - Public → toti!
  - Privat → proprie unei clase
  - Protejat → vizibilitate la nivelul subclasei

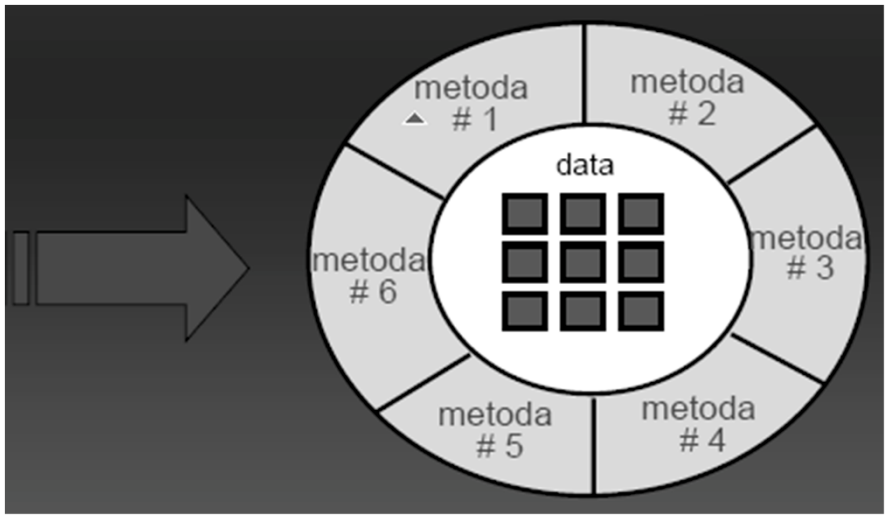
Principiile lui David Parnas

1. Dezvoltarea de componente software trebuie sa furnizeze utilizatorilor toata informatia necesara spre o buna utilizare a componentelor, dar **NICI** o alta informatie in plus.

2. Dezvoltatorul de componente software trebuie sa primeasca toata informatia necesara in vederea finalizarii cu succes a responsabilitatilor sale referitoare la componente, dar **NICI** o alta informatie in plus.

Incapsulare – consecinte

- Obiectele incapsuleaz a atat date cat si procedurile aferente manipularii datelor



Polimorfism (1)

- Capacitatea unui operator de a se aplica obiectelor din clase diferite

Deplasare

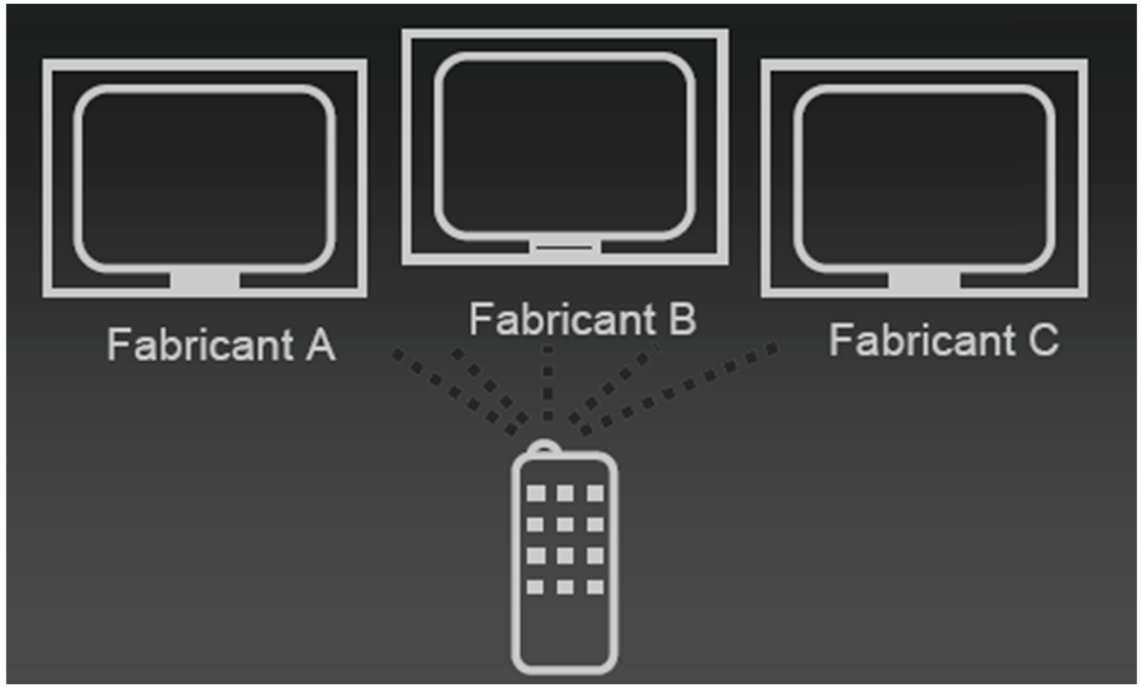
Deplasare

Deplasare

Polimorfism (2)

sau

- Abilitatea unui operator de a masca implementari diferite in spatele unei interfete unice



Puncte *tari* ale POO

- O singura paradigma
- Un singur limbaj utilizat de utilizatori, analisti, proiectanti, etc.
- Faciliteaza re-utilizarea codului si structurile arhitecturale
- Modeleaza mai corect lumea reala:
  - Descriere mai precisa a fluxurilor de date si a proceselor aferente
  - Descompuneri naturale ale structurilor mari
- Usor de inteles si de intretinut
- Stabilitate
- O mica schimbare la nivelul cerintelor nu conduce la modificari profunde in sistemul aflat in faza de dezvoltare