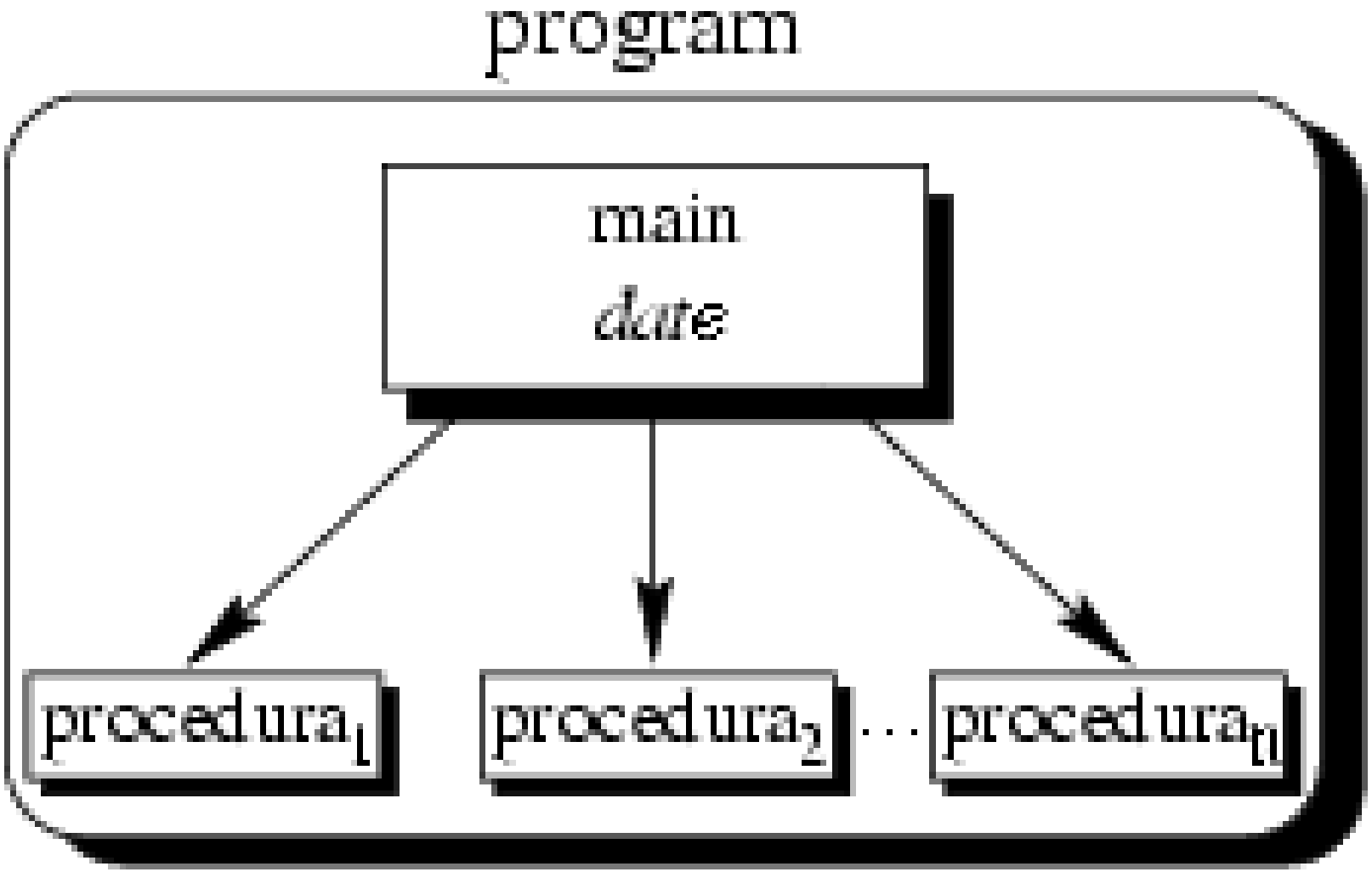
Programare orientată pe obiecte

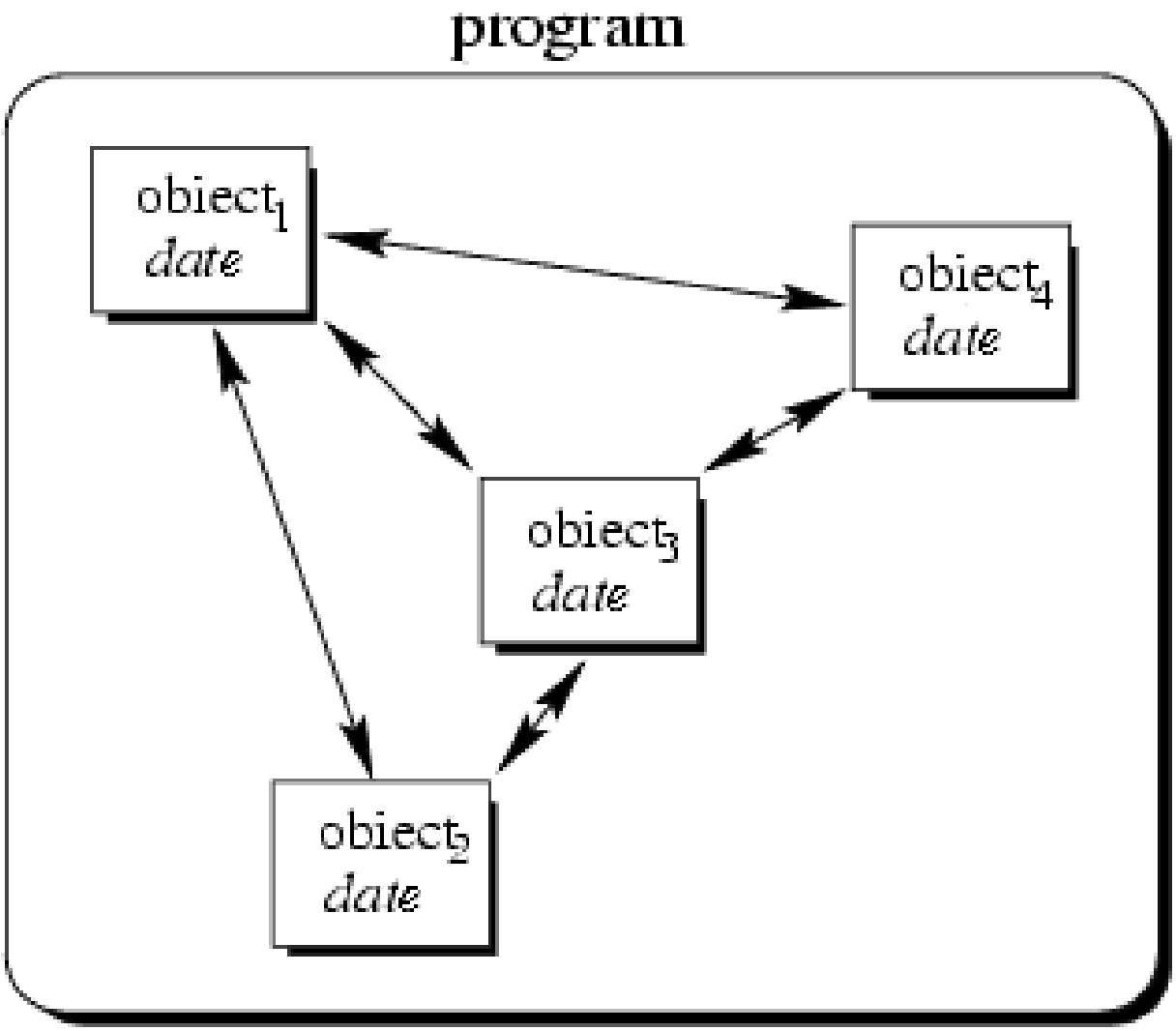
(suport de curs)

§1. Principiile programării orientate pe obiecte.

Strategia procedurală şi structurată de scriere a programelor presupune existența unui program principal şi a unui şir de proceduri (subprograme, funcții), iar rezultatul final este obținut prin apelarea procedurilor respective transmițându-le lor datele necesare:


program
main
date
procedura1
procedura2
...
procedurat1

Programarea orientată pe obiecte (POO) este tehnologia de programare în care programul este reprezentat ca o colecție de obiecte discrete, care conțin nişte seturi de structuri de date şi proceduri ce interacționează cu alte obiecte:


program
obiect1
date
obiect4
date
obiect3
date
obiect2
date

Există mai multe sensuri ale cuvântului „obiect”:

- *corp solid*, cunoscut direct cu ajutorul simțului;
- *lucru* sau *complex de lucruri* apărute ca rezultat al procesului de muncă (de exemplu, obiect de consum, obiect de uz casnic);
- *materie asupra căreia este orientată activitatea spirituală sau artistică* (de exemplu, obiect de cercetare, obiect de descriere);
- *ființă* sau *lucru* pentru care cineva manifestă un sentiment (de exemplu, obiect de admirație);
- *disciplină de studiu* într-o instituție de învățământ;
- *filos. Corp* sau *fenomen* existent în realitate, în afara subiectului şi independent de conştiința acestuia;
- *lingv. Parte de propoziție* care indică asupra cui este orientată acțiunea verbului; complement.

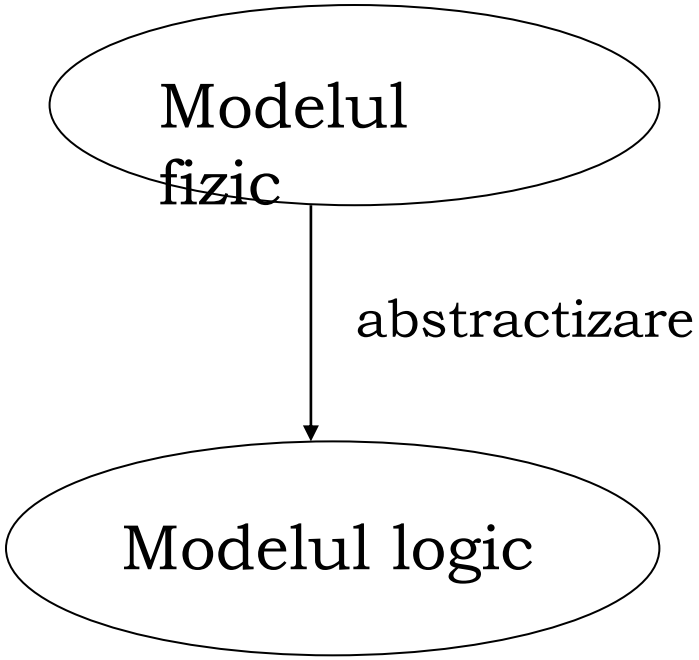
Aceste definiții cuprind diferite domenii ce țin de natură şi de activitatea omului.

Ele toate pot fi modelate în POO prin declararea **claselor** şi crearea **obiectelor** respective.

Pentru a aplica tehnologia orientată pe obiecte, inițial, este necesar de a realiza o etapă importantă - abstractizarea.

Modelul fizic al oricărui domeniu de problemă este unul complex, depinzând de mulți parametri şi relații.

Astfel, se face o simplificare a modelului fizic, obținând modelul logic al domeniului de problemă. Sunt omise din modelul fizic o serie de proprietăți şi legături care nu sunt esențiale în contextul problemei formulate. Acest proces de simplificare se numeşte abstractizare:



Deci, abstractizarea presupune „sărăcirea” datelor inițiale.

Exemplu. La elaborarea unui program de evidență a cadrelor într-o întreprindere, este necesar să descriem obiectul „Angajat al întreprinderii” care în esență este o persoană (o ființă umană). Orice persoană angajată la întreprindere poate fi descrisă (caracterizată) printr-un set de proprietăți, cum ar fi de exemplu:

- Nume
- Prenume
- Domiciliu
- Anul naşterii
- Funcția
- Salariul
- Vechimea în muncă
- Înălțime
- Culoarea ochilor
- Culoarea părului
- Greutatea
- Numărul de copii
- etc.

De aici se observă că o serie de proprietăți nu sunt adecvate problemei formulate, de exemplu, *“Înălțime”*, *“Culoarea ochilor”*, *“Culoarea părului”* şi *“Greutatea”*, ele fiind eliminate din descriere.

Pe lângă proprietăți un obiect poate fi caracterizat şi printr-o serie de operații (funcționalitate), de exemplu:

- *Schimbarea salariului;*
- *Schimbarea domiciliului;*
- *Schimbarea funcției;*
- *Creşterea vechimii în muncă;*
- *Schimbarea numelui;*
- *etc.*

Definiţie. Obiectul în programare reprezintă modulul de program care îndeplineşte următoarele cerințe principale:

- întruneşte **date** (caracterizează proprietățile obiectului) şi **operaţiile** asupra lor (se mai numesc **metode** şi caracterizează comportarea obiectului sau posibilitățile obiectului, adică ceea ce poate face obiectul);
- posedă proprietățile de moştenire, încapsulare şi polimorfism.

La baza programării orientate pe obiecte stă noțiunea de **clasă**.

O clasă în programare întruneşte toate obiectele de una şi aceeaşi natură (la fel ca şi în lumea reală). Obiectele care aparțin uneia şi aceleiaşi clase posedă una şi aceeaşi structură, comportare şi relații cu obiecte din alte clase.

Definiţie. Clasa este tipul abstract de date creat de utilizator (programator).

În clasă se descriu împreună câmpurile de date şi metodele sub formă de funcții membre.

Metodele efectuează operații asupra acestor date şi alte acțiuni ce caracterizează comportamentul obiectelor clasei respective.

În baza claselor deja create există posibilitatea de a crea clase derivate care moştenesc proprietățile claselor de bază.

- Exemplarul de obiect în POO se mai numeşte **instanţă** (*engl.* instance), este un obiect concret din setul de obiecte ale uneia şi aceleiaşi clase. Crearea exemplarului de obiect al unei clase se numeşte **instanţierea** obiectului.

- Când creăm o clasă, definim implicit un nou tip de date. Deci, un obiect este o variabilă de un tip definit de utilizator (tipul clasă).

Programarea orientată pe obiecte se bazează pe un şir de principii.

Principiul 1. **Încapsulare.**

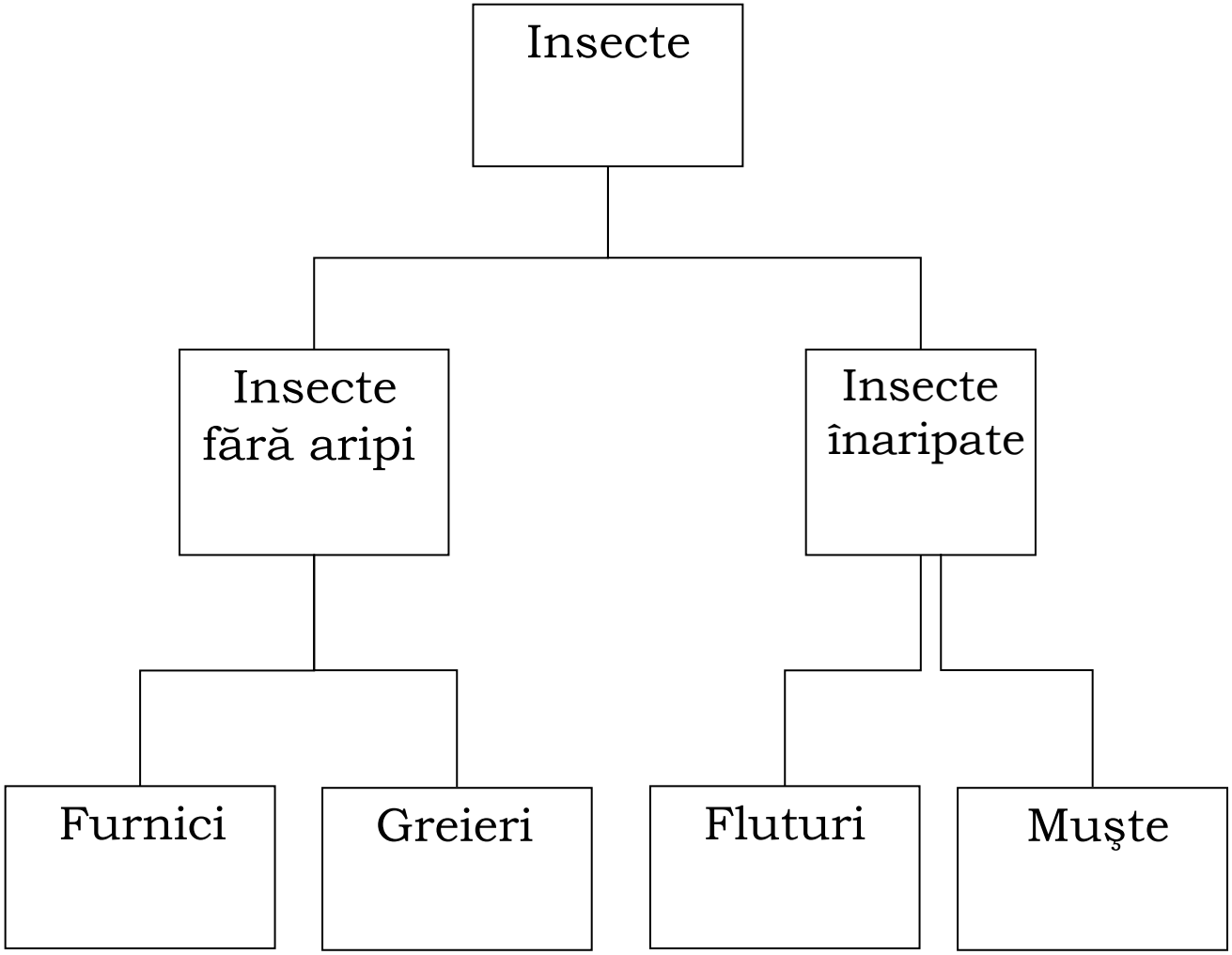
Încapsularea în POO (*engl.* encapsulation) este unirea într-un singur tot a datelor şi metodelor, ceea ce permite ascunderea structurii interne de date şi a metodelor obiectului de restul programului. Celelalte obiecte din program au acces numai la interfața obiectului. Interfața este reprezentată prin membrii publici prin care se efectuează toate interacțiunile cu acest obiect.

Încapsularea este un mecanism care leagă într-un tot întreg codul şi datele, păstrându-le pe ambele în siguranță contra intervențiilor din afară şi de utilizări greşite. În plus, încapsularea asigură crearea obiectelor. Un obiect este o entitate logică ce încapsulează atât date, cât şi codul care manevrează aceste date. Într-un obiect, o parte din cod şi/sau date pot fi particulare (private sau protejate) acelui obiect şi inaccesibile pentru oricine din afara lui. Astfel, un obiect dispune de un nivel semnificativ de protecție care împiedică modificarea accidentală sau utilizarea incorectă a părților obiectului de către secțiuni ale programului cu care acestea nu au legătură.

Principiul 2. **Moştenire.**

Moştenirea în POO (*engl*. inheritance) reprezintă proprietatea obiectului, care constă în aceea că caracteristicile unui obiect (obiectul-părinte) pot fi transmise unui alt obiect (obiect-fiu) fără descrierea lor repetată. Moştenirea simplifică descrierea obiectelor şi admite clasificarea lor.

Exemplu. Clasa **furnici** face parte din clasa de **insecte fără aripi**, care la rândul său face parte din clasa mai generală **insecte.** O astfel de clasificare se numeşte **taxonomie**. Taxonomia este o ştiință care se ocupă cu stabilirea legilor de clasificare şi de sistematizare a domeniilor cu o structură complexă.



Exemplu de ierarhie taxonomică

Fără utilizarea claselor cu proprietatea de moştenire fiecare obiect ar trebui definit cu enumerarea tuturor caracteristicilor sale. Însă, prin folosirea clasificărilor, un obiect are nevoie doar de definirea acelor calități care îl fac unic în clasa sa.

Mecanismul moştenirii permite ca un obiect să fie descris ca un exemplar al unui caz mai general.

Principiul 3. **Polimorfism.**

În limbajele de programare orientate pe obiecte polimorfismul este caracterizat prin fraza „o singură interfață, metode multiple”. Polimorfismul în POO (*engl*. polymorphism) este capacitatea obiectului de a selecta, din mai multe metode metoda corectă, în dependență de tipul de date, primite în mesaj, sau, altfel spus, posibilitatea de a denumi la fel diferite acțiuni la diferite niveluri ale ierarhiei de clase. Polimorfismul simplifică programarea, deoarece selectarea acțiunii necesare se realizează automat.

Dăm un exemplu de polimorfism din matematică.

Una şi aceeaşi expresie **a+b** va fi interpretată în conformitate cu tipul operanzilor operației de adunare. Dacă **a** şi **b** sunt două numere, expresia dată reprezintă suma a două numere, dacă **a** şi **b** sunt două matrice – rezultatul va fi suma matricelor, dacă **a** şi **b** sunt doi vectori – expresia reprezintă vectorul rezultant etc.

În limbajele de programare orientate pe obiecte se folosesc mai multe mecanisme prin care se realizează principiul de polimorfism. Cele mai principale sunt suprascrierea şi supraîncărcarea funcţiilor şi operatorilor, posibilitatea de a specifica valorile implicite pentru parametri, regulile de atribuire între obiecte ale claselor descendente, funcţiile virtuale.

(~ §1)