PROGRAMAREA ORIENTATĂ OBIECT C++

Conf.univ.dr. Ana Cristina DĂSCĂLESCU

Universitatea Titu Maiorescu

Moștenire

- Mecanismul prin care o clasă preia structura (datele membre) şi comportamentul (metodele) unei alte clase create anterior, **la care adaugă elemente specifice.**

- Este un principiu strict orientat obiect

- În limbajul C++ este implementată prin conceptul de **moştenire**

- Stabileste o relatie "este un/este o" (is-a)

- Terminologie

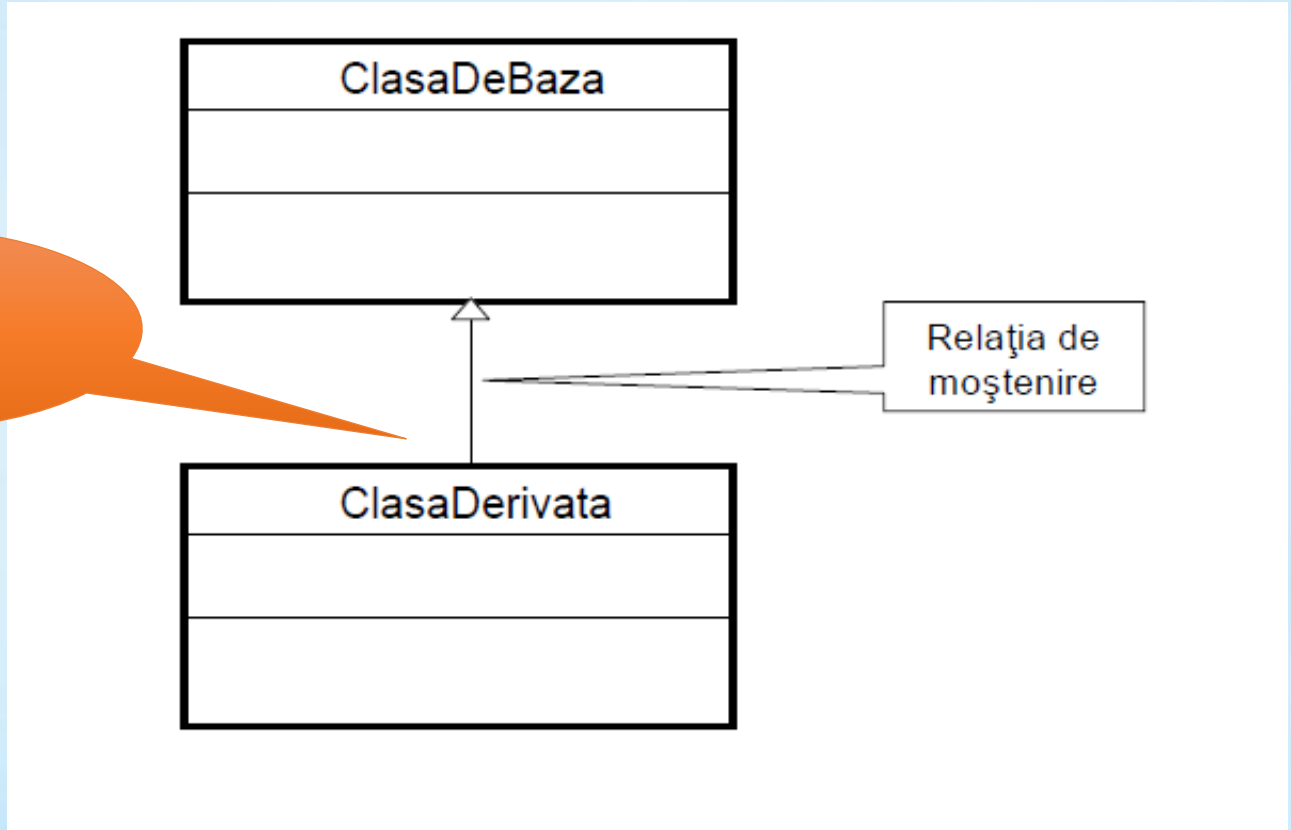
Clasă de baza = clasa de la care se preiau structura şi comportamentul

Clasa derivată = clasa care preia structura şicomportamentul

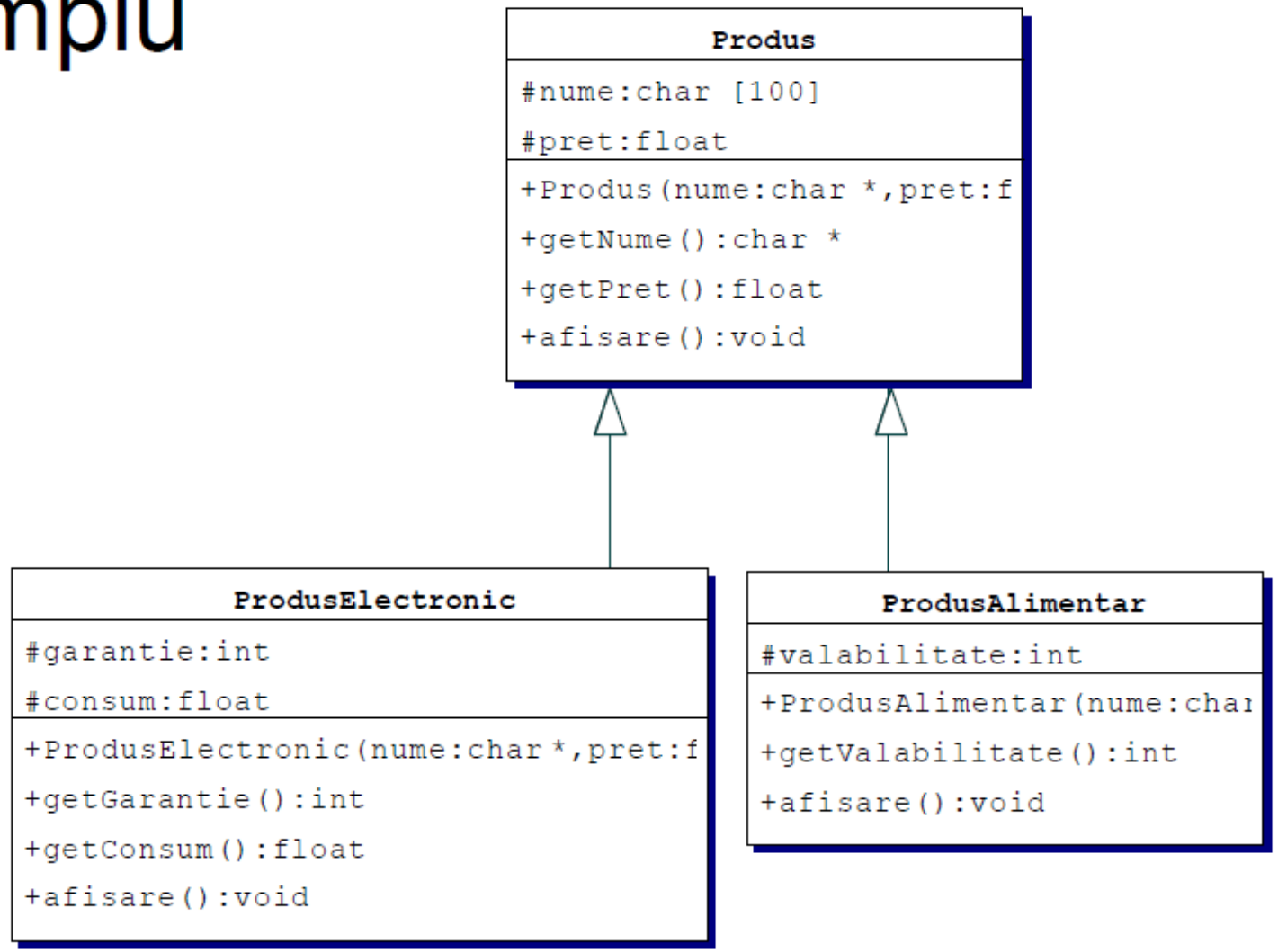
Avantaje

- Reutilizarea codului creat anterior
- Dezvoltarea unei aplicații într-o manieră incrementală, respectiv se compilează și se tetează doar clasa nou obținută
- Obținrea unei ierahii de clase ce se poate modela cu ușurință prin diagame UML
- Implementarea polimorfismului în timpul executării programului, folosind funcții virtuale

Diagrama UML



Exemplu


Produs
#nume:char [100]
#pret:float
+Produs(nume:char *,pret:f
+getNume():char *
+getPret():float
+afisare():void
ProdusElectronic
#garantie:int
#consum:float
+ProdusElectronic(nume:char *,pret:f
+getGarantie():int
+getConsum():float
+afisare():void
ProdusAlimentar
#valabilitate:int
+ProdusAlimentar(nume:char
+getValabilitate():int
+afisare():void

Ce se moștenește?

- Clasa derivată moștenește toți membrii publici și protejați din clasa baza.
- Membrii privați nu sunt moșteniți, dar pot fi accesați prin metode publice sau protejate din superclasă.
- Nu se moștenesc datele membre și metodele statice, precum și metoda operator=
- Metodele constructor, nefiind considerate metode membre ale unei clase, nu se moștenesc, dar un constructor din clasa derivată poate apela constructorii din clasa de bază.

Sintaxa unei clase derivate

Sintaxa definirii unei clase derivate este următoarea:

```
class IdClasaDerivata: modif_acces1 IdClasaB1, …, modif_accesN IdClasaBn{
        //date si metode specifice clasei derivate
};
```

unde:

- IdClasaDerivata este numele clasei derivate

- IdClasaB1, …, dClasaBn sunt clasele de bază de la care se moştenesc datele şi metodele modif_acces1, …, modif_accesN sunt modificatori de acces: *public, protected, private*

Accesul asupra membrilor moșteniți

Protectia în clasa de bază	Modificator de acces utilizat în lista claselor de bază	Drept de acces în clasa derivată
public private protected	public public public	public inaccesibil protected
public private protected	private private private	private inaccesibil private

Exemplu

```
class Produs{
  protected:
     char nume[100];
     float pret;
  public:
     Produs(char *nume, float pret);
     char* getNume();
     float getPret();
     void afisare();
};
```

```
class ProdusAlimentar: public Produs{
     protected:
     int valabilitate;
     public:
     ProdusAlimentar(char *nume,
float pret, int valabilitate);
     int getValabilitate();
     void afisare();
};
```

Clasa derivată

Clasa de bază

modificator acces

Exemplu

Clasa derivata

Clasa de baza

modificator acces

```
class ProdusElectronic: private Produs{
      protected:
      int garantie;
      float consum;
      public:
      ProdusElectronic(char *nume, float
pret, int garantie, float consum);
      int getGarantie();
      float getConsum();
      void afisare();
};
```

```
int main(){
  ProdusAlimentar pa("Iaurt",15,30);
  pa.afisare();
  cout<<"Pretul p.a"<<pa.getPret();
  ProdusElectronic pe("LCD",1000,24,12);
  pe.afisare();
  cout<<"Pretul p. e." <<pe.getPret();

}
```

inaccesibila

Redifinirea membrilor moșteniți

- Mecanismul de redefinire reprezintă un concept puternic în limbajele orientate pe obiecte care permite clasei derivate să redefinească o metoda moștenită și să-i modifice comportamentul.

- La executare, în raport cu tipul obiectului se va invoca metoda corespunzătoare.

- **Observații**

- O metodă din clasa derivată care redefinește o metodă din clasa de bază trebuie să păstreze lista inițială a parametrilor formali

- Metoda din clasa de bază este „ascunsă” de cea redefinită în clasa derivată!!!

- Metoda redefinită în clasa derivată poate să apeleze metoda din clasa de bază folosind operatorul **::**

Redefinirea membrilor moșteniți

```
class Angajat{
  char *nume;
  double salariu;
public:
 Angajat(){nume=NULL; salariu=1900;}
 Angajat(char *nume, double salariu)
 {
this->nume = new  char[strlen(nume)+1];
strcpy(this->nume, nume);
this->salariu = salariu;
 }
void afisare()
  {
     cout<<nume<<" "<<salariu<<" ";
  }};
```

```
class Administrator:public Angajat
{
    int sectie;
public:

    void afisare()
    {
        Angajat::afisare();
        cout<< sectie;
    }
}
```

Medoda redefinită

Constructorii și destructorul claselor derivate

- Dacă o clasă D este derivata din clase (B1, …, Bn), atunci constructorul clasei D va avea suficienţi parametri pentru a iniţializa datele membru ale claselor B1,…, Bn.

- La crearea unui obiect al clasei D se vor apela mai întâi constructorii claselor B1,…,Bn, în ordinea specificată în lista de moştenire, pentru a se iniţializa datele membre ale claselor de bază şi apoi se vor executa inițializările constructorului clasei D.

- Dacă o clasă D este derivata din clase (B1, …, Bn) atunci constructorul clasei B va avea suficienţi parametri pentru a iniţializa datele membru ale claselor B1,…, Bn.

Constructorii și destructorul claselor derivate

- Explicit, se definește lista parametrilor de inițializare

```
class D: public B1, ..., public Bn{
    D(lista de parametri);
};

D::D(lista de parametri):B1(sub_lista_param1), . . ., Bn(sub_lista_param1){
  instructiuni
}
```

Apelul explicit

- Implicit, dacă o clasă de bază nu conține niciun constructor, atunci constructorul clasei derivate apelează constructorul implicit, definit de către compilator!!!

Constructorii și destructorul claselor derivate

```
Administrator(): Angajat()
  {
     sectie = 0;
  }
  Administrator(char* nume, double salariu, int sectie):Angajat(nume, salariu)
  {
     this->sectie = sectie;
  }
```

Apel constructor din clasa de bază
nume = NULL
Salariu=199

Argumente necesare inițializării datelor membre din clasa ade bază

Apel constructor din clasa de bază

.