Modelarea interfetelor în UML

Prof. univ. dr. ing. Florica Moldoveanu

Curs *Ingineria programelor* – UPB, Automatică şi Calculatoare
2020-2021

Interfeţe (1)

In Java, **definitia unei clase** poate fi separata in doua parti:

- **Interfata**: metodele publice ale clasei, exceptand constructorul si distructorul
- **Implementarea:** implementarile metodelor, constructorul, distructorul si atributele.

Avantaj: implementarea clasei este ascunsa (“information hiding”)

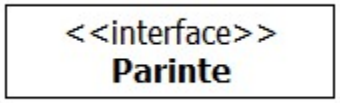
→ clientii clasei sunt fortati sa foloseasca numai interfata clasei

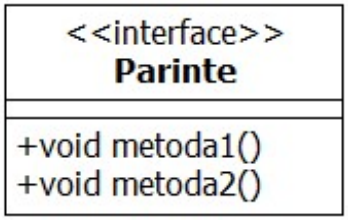
→ modificarea implementarii interfetei nu afecteaza clientii

Reprezentarea interfețelor in UML

❖ O interfață poate fi reprezentată într-o diagramă de clase sau într-o diagramă de componente.

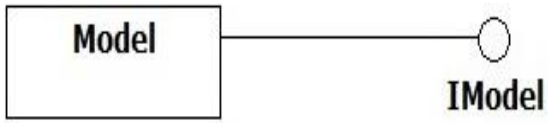
1. Reprezentare asemănătoare cu a unei clase:





Interfeţe (2)

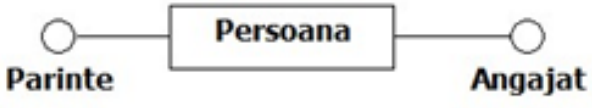
2. Reprezentare printr-un cerc adnotat cu numele interfeței, notatie numita *lollipop* (acadea):



Clasa *Model* implementează interfața *IModel*.

O clasa poate sa implementeze mai multe interfeţe:

Exemplu: clasa Persoana poate implementa interfeţele “Angajat” si “Parinte”.



Clasa **Persoana *furnizeaza*** interfeţele **Parinte** si **Angajat**.

→ **o interfaţă reprezinta un rol** pe care obiectele unei clase il joaca in raport cu obiectele altei clase

→ **obiectele unei clase pot juca mai multe roluri**

❖ In diagramele de clase conceptuale interfetele furnizate de o clasa se reprezinta prin nume de rol atasate asocierilor clasei cu alte clase.

Interfete in Java

In Java:

```
interface Angajat
{  float Salariu();
   int    oreLucrate();
   String Name();
 }

interface Parinte
{ public void metoda1();
  public void metoda2();
}

class Persoana implements Angajat, Parinte
{.....................................}
```

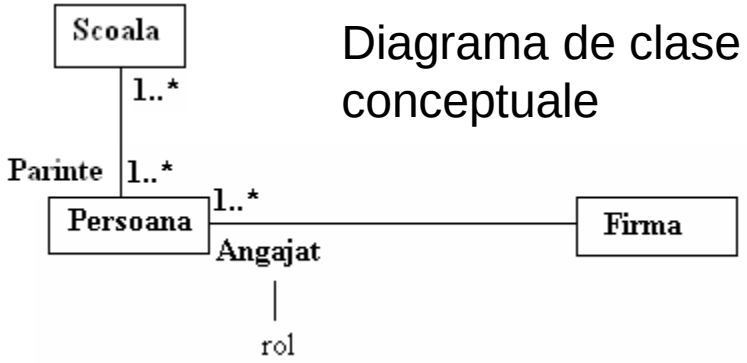
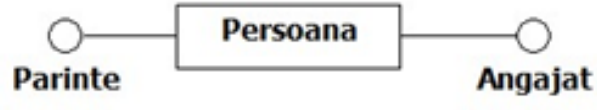
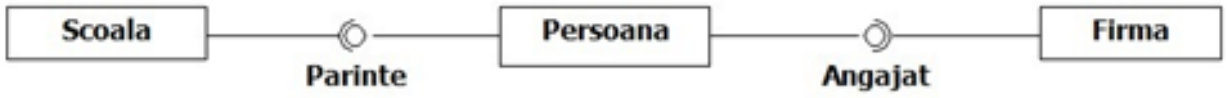


Diagrama de clase conceptuale

Interfeţele apar ca nume de rol in asocierea dintre 2 clase.



Clasa **Scoala *utilizeaza*** interfata **Parinte** iar clasa **Firma** *utilizeaza* interfata **Angajat.**



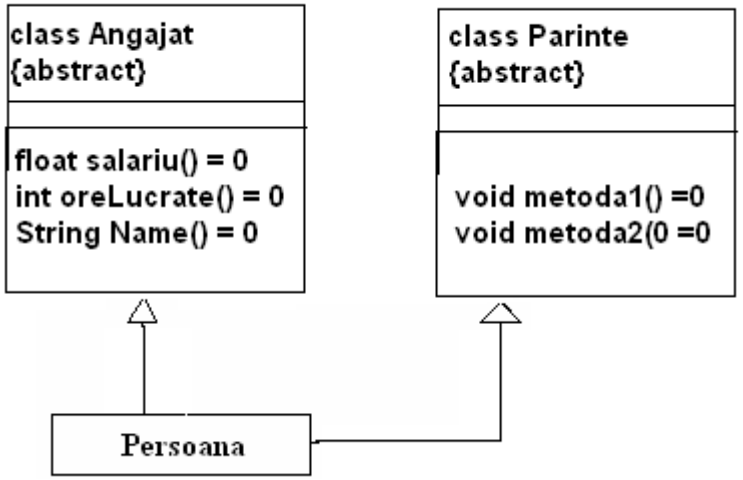
Interfete in C++

In C++ o interfata poate fi reprezentata printr-o clasa abstracta pura:

```
class Angajat // clasa abstracta pura
{ // contine numai functii virtuale pure
  // nu contine atribute
   public: virtual float Salariu()=0;
           virtual int oreLucrate()=0;
           virtual String Name()=0;
};

class Parinte // clasa abstracta pura
{ public: virtual void metoda1()=0;
           virtual void metoda2()=0;
};

class Persoana: public Angajat, Parinte //mostenire
{ public: Persoana(...); // constructorul
           virtual  void ~ Persoana(); // distructorul
           virtual float Salariu();
           virtual int oreLucrate();
           virtual String Name();
           virtual void metoda1();
           virtual void metoda2();
}
```



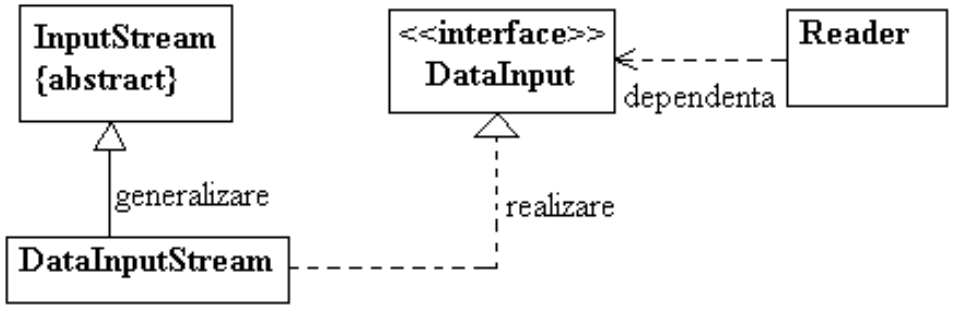
Interfeţe şi clase

In concluzie:

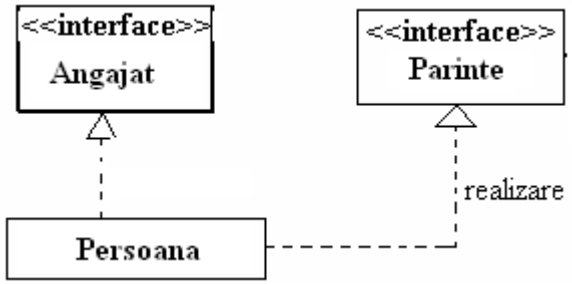
- O interfaţă este un set de metode corelate care definesc o anumita comportare.
- Toate metodele sunt publice si nu se specifica nici un fel de implementare pentru ele.
- O interfata nu are stare (nu contine variabile).

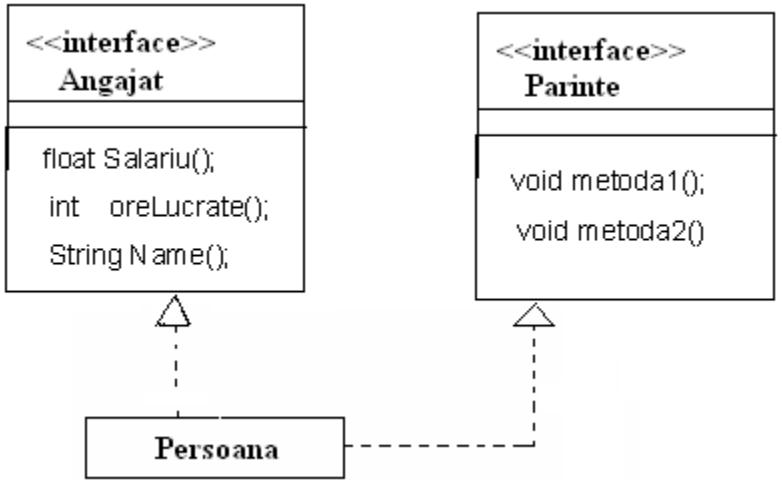
Intre interfete si clase pot fi stabilite relatii de realizare şi dependenţă:

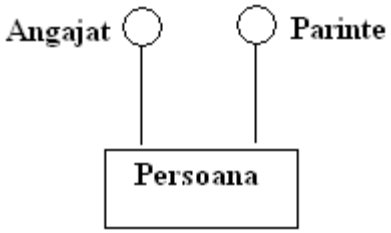
- clasa **InputStream** este abstracta.
- clasa **DataInputStream *implementeaza*** atat clasa abstracta **InputStream** cat si interfata **DataInput.**
- clasa **Reader *utilizeaza*** functiile oferite de interfata **DataInput**.

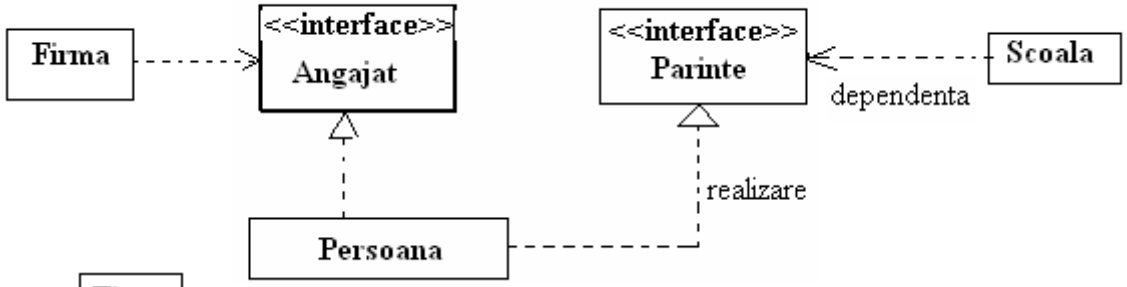


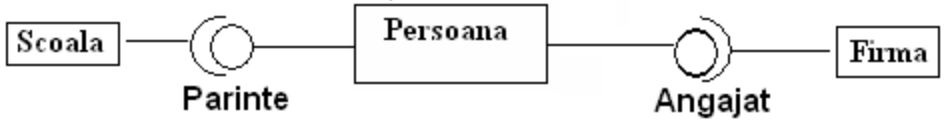
Reprezentarea relatiilor dintre clase si interfate











Intelegerea unei interfeţe

Pentru a se uşura întelegerea unei interfete, se pot ataşa:

- pre si post conditii pentru fiecare operatie
- specificarea formala a semanticii, folosind OCL (Object Constraint Language, inclus in UML)
- se poate atasa un automat (diagrama de stari) pentru a specifica ordonarea in timp a operatiilor interfetei
- se pot atasa diagrame de colaborare pentru a specifica comportarea prevazuta pentru interfata.