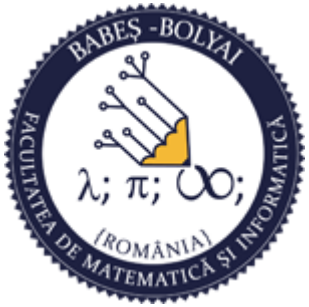
UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică



Programare orientată obiect

Curs 08

Laura Dioşan

POO

- Clase
  - Clase abstracte
  - Polimorfism
  - Legare timpurie şi târzie
  - Mecanismul virtual
  - Metode pure
  - UML
  - Avantajele polimorfismului

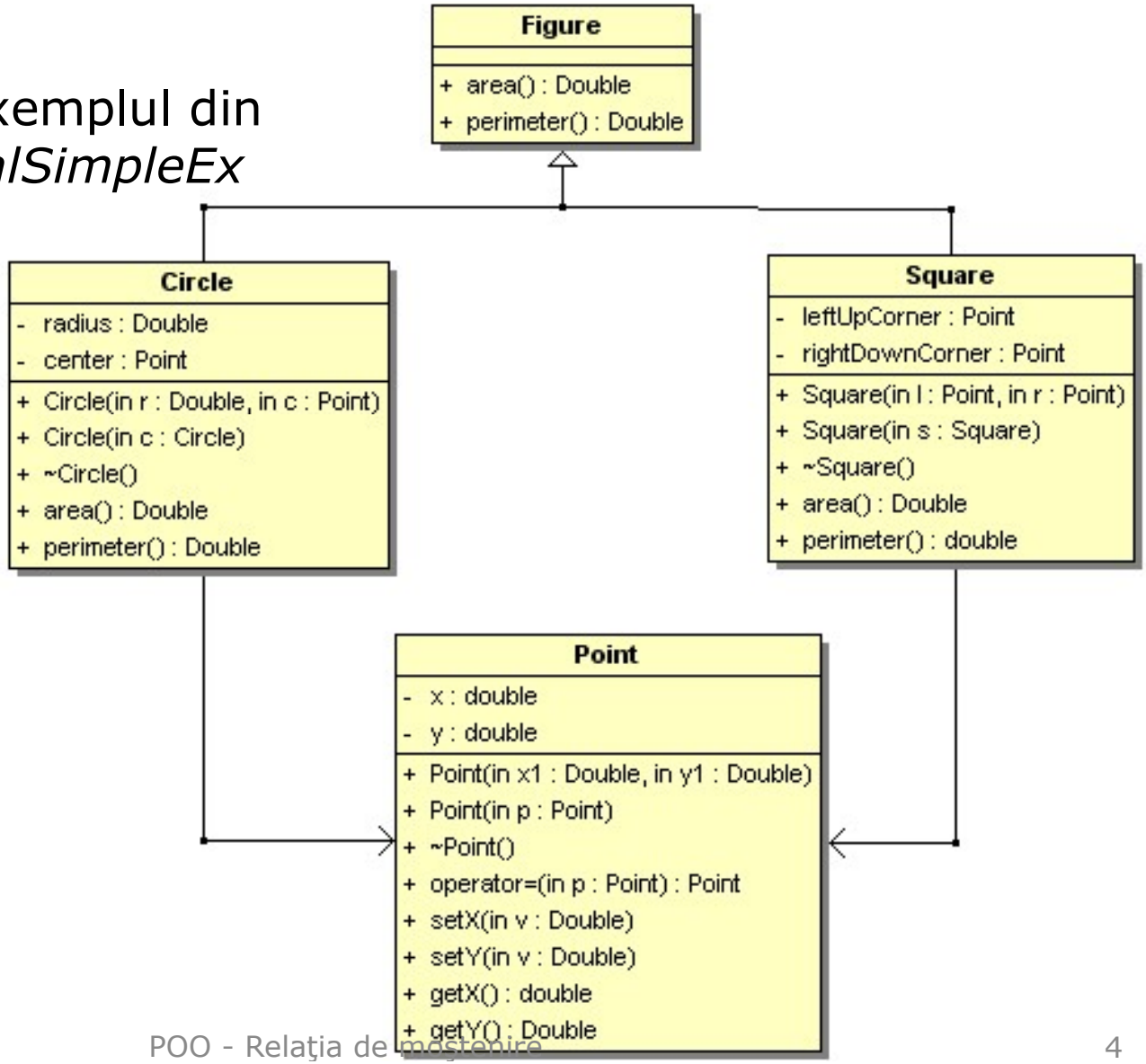
Polimorfism

- Proprietatea unor entităţi de:
  - A se comporta diferit în funcţie de starea lor
  - A reacţiona diferit la acelaşi mesaj

- Proprietate a unui limbaj OO de a permite manipularea unor obiecte diferite prin intermediul unei interfeţe comune
  - Există o relaţie de derivare între interfaţă şi celelalte clase

Exemplu

- A se consulta exemplul din directorul *VirtualSimpleEx*
  - Figure.h
  - Point.h
  - Circle.h
  - Square.h



Explicații

- Metodele unei clase
  - Clasice – adresa metodei:
    - este fixată la compilare/link-editare
    - nu poate fi modificată în timpul rulării
  - Polimorfice – adresa metodei:
    - poate fi modificată în timpul rulării

- Legare (binding)
  - Conectarea unui apel de funcţie cu corpul funcţiei

- Legare timpurie (legare statică) - e*arly binding*
  - când legarea se realizează înainte de rularea programului (de către compilator şi link-editor)
  - mecanism implicit

- Legare târzie (legare dinamică sau la rulare) - *late binding*
  - când legarea are loc la rulare, bazându-se pe tipul obiectelor
  - cauzată de folosirea funcţiilor virtuale

Funcţii virtuale

- Legarea târzie apare doar:
  - prin folosirea metodelor virtuale în clase de bază
  - când o adresă a clasei de bază (unde se află acele funcţii virtuale) este folosită
- Definire

```
virtual type fc_name(fpl);
```

- Cuvântul rezervat *virtual* se foloseşte:
  - doar pentru declararea funcţiei (nu şi pentru definirea ei)
  - doar în clasa de bază (deoarece acea funcţie virtuală va fi virtuală în toate clasele derivate)
  - doar funcţiile membre pot fi virtuale
  - funcţiile globale prietene nu pot fi virtuale

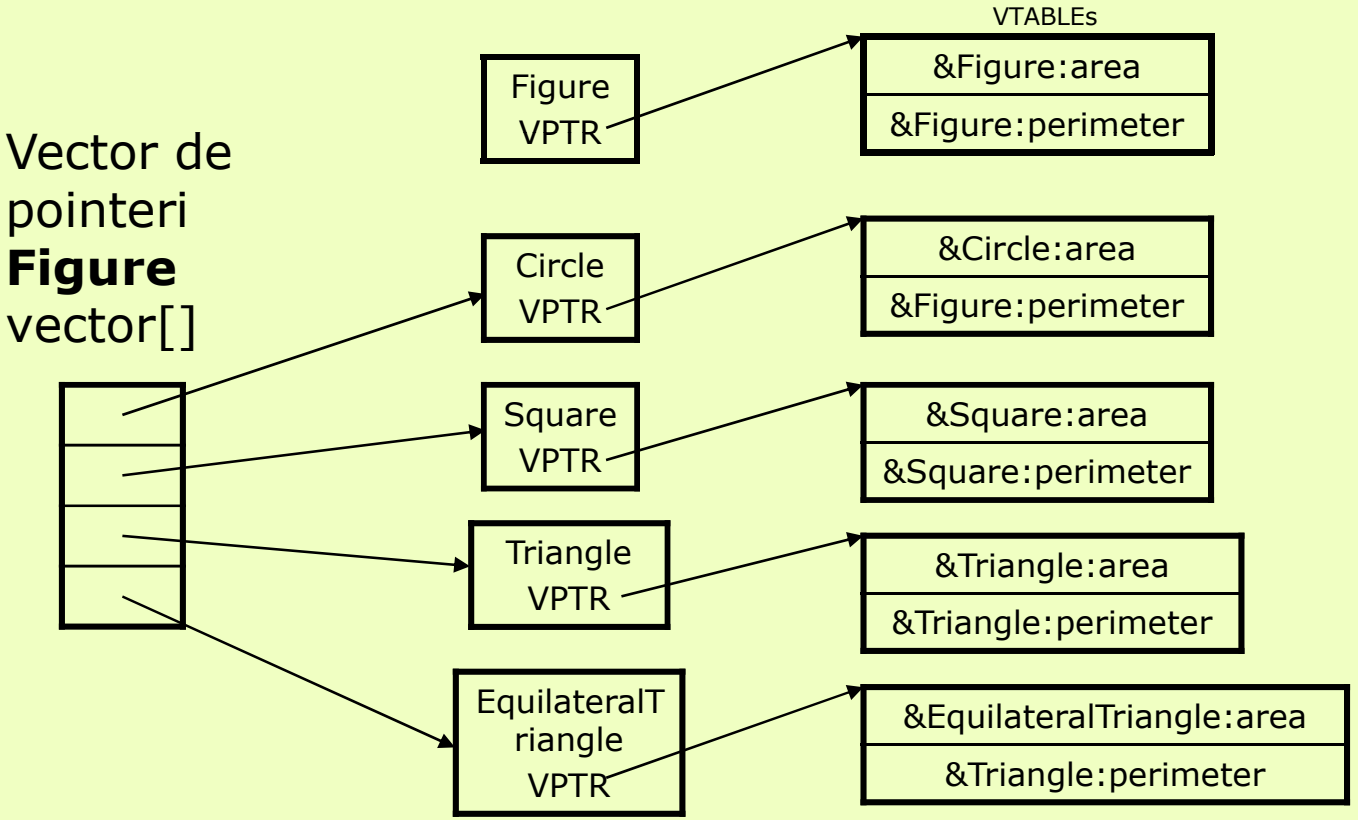
- O metodă virtuală din clasa de bază poate fi:
  - Moştenită în clasa(ele) derivată(e)
  - Redefinită în clasa(ele) derivată(e) → suprascriere (*overriding*)
    - redefinirea unei funcţii virtuale într-o clasă derivată

Mecanismul legării întârziate

- necesită
  - Funcţii virtuale
  - Referinţe sau pointeri la clasa de bază
- Compilatorul:
  - Crează o singură tabelă (numită VTABLE) pentru fiecare clasă care conţine funcţii virtuale
  - Plasează adresele funcţiilor virtuale ale clasei respective în VTABLE
  - Plasează (în mod secret) un pointer în fiecare clasă cu funcţii virtuale, numit *vpointer* (abreviat VPTR), care referă VTABLE-ul acelui obiect
- Când o funcţie virtuală este apelată printr-un pointer la clasa de bază (**apel polimorfic**), compilatorul:
  - inserează cod pentru plasarea VPTR şi
  - se uită la adresa funcţiei în VTABLE, apelând astfel funcţia corectă – are loc legarea întârziată

Mecanismul legării întârziate

- A se consulta exemplul din directorul *VirtualSimpleEx*
  - Figure.h, Point.h
  - Circle.h, Square.h, Triangle.h, EquilateralTriangle.h



De ce legare întârziată?

- Funcţiile virtuale – o opţiune
  - Care nu este chiar eficientă
    - 2 apeluri sofisticate (în locul unui simplu apel la o adresă absolută)
  - C++ provine din moştenirea C-ului, unde eficienţa este critică
  - Cuvântul rezervat **virtual** este folosit pentru eficienţa prezentării (adaptării)

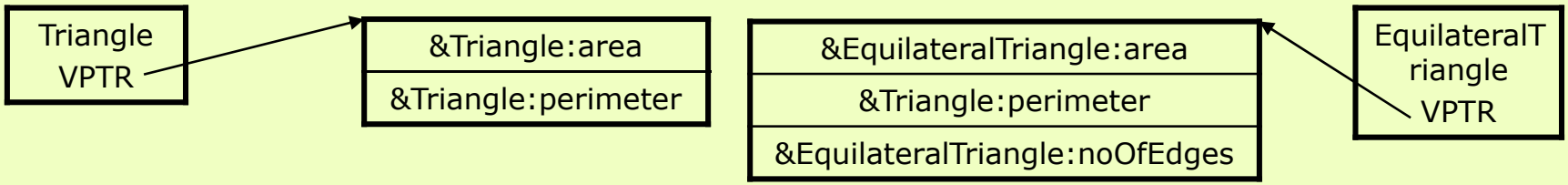
Funcţii virtuale – mai multe detalii

- Nu pot să fie virtuali
  - constructorii
    - VPTR este instanţiat în cadrul constructorului
  - metodele statice
  - funcţiile globale prietene
- Pot să fie virtuale
  - metodele ordinare
  - metodele prietene
  - destructorii
    - uneori aceştia TREBUIE să fie virtuali

Funcţii virtuale – mai multe detalii

- A se consulta exemplul din directorul *VirtualConstrDestructor*

- Moştenirea şi VTABLE
  - Funcţii virtuale noi în clasele derivate
  - Downcast
  - A se consulta exemplul din directorul *VirtualSimpleEx*
    - *Figure.h, Triangle.h, EquilateralTraingle*
    - *Test.cpp* and *addNewVirtualFunctionInDerivedClass()*



Metode nule

- Metode cu corpul vid
  - Declarate in clasa de **bază**
  - Definite în clasa de **bază**
- Metode folosite pentru declararea unui comportament general (abstract) la nivelul clasei de bază
- Definire

```
virtual type fc_name(fpl) {
};
```

Metode pure

- Metode fără corp
  - Declarate în clasa de **bază**
  - Definite în clasa(ele) **derivată(e)**
- Metode folosite pentru declararea unui comportament general (abstract) la nivelul clasei de bază

- Definire

```
virtual type fc_name(fpl) = 0;
```

  - Compilatorul rezeră un *slot* pentru funcţia pură în VTABLE, dar nu pune (încă) o adresă concretă in acel slot
  - Chiar dacă doar o singură metodă a unei clase este virtuală pură, VTABLE este incompletă
- Nu pot fi apelate
  - Scopul lor este de a anunţa doar signatura metodei

Metode pure - exemplu

```
class Figure{
public:
    virtual ~Figure(){}
    virtual float area() = 0;
    virtual float perimeter() = 0;
};
class Circle : public Figure{
private:
    float radius;
    Point center;
public:
    Circle(float r, Point &c):radius(r), center(c){
    Circle(const Circle &c){
        radius = c.radius;
        center = c.center;
    }
    ~Circle(){}
    float area(){
        cout << "[Circle] : area " << endl;
        return PI * radius * radius;
    }
    float perimeter(){
        cout << "[Circle] : perimeter " << endl;
        return 2 * PI * radius;
    }
};
```

```
void printArea(Figure* f){
    cout << f->area() << endl;
}

void printPerimeter(Figure* f){
    cout << f->perimeter() << endl;
}

void virtualSimpleDerived(){
    Figure* figPointer;
    figPointer = new Circle(5, Point(2,3));
    printArea(figPointer);
    printPerimeter(figPointer);
}

int main(){
    virtualSimpleDerived();
    return 0;
}
```

Clase abstracte

- O clasă cu cel puţin o metodă pură virtuală
- Folosite pentru a declara un comportament comun mai multor clase derivate (din clasa abstractă)
- O clasă abstractă poate să conţină şi:
  - Date comune (tuturor claselor derivate)
  - Metode comune (tuturor claselor derivate)

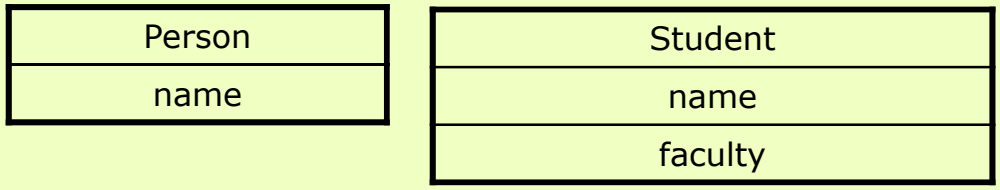
Clase abstracte

- O clasă abstractă nu poate fi instanţiată
  - Amintiţi-vă:
    - Compilatorul rezervă un *slot* pentru funcţia pură în VTABLE, dar nu pune (încă) o adresă concretă in acel slot
    - Chiar dacă doar o singură metodă a unei clase este virtuală pură, VTABLE este incompletă
  - Întotdeauna trebuie folosit un pointer sau o referinţă la clasa abstractă

- Funcţiile virtuale pure:
  - Sunt utile pentru că:
    - Ele explicitează abstractul unei clase
    - Indică utilizatorului şi compilatorului ceea ce se doreşte a fi făcut
  - Previn transmiterea prin valoare a claselor abstracte
    - O modalitate de a evita felierea obiectelor (*object slicing)*

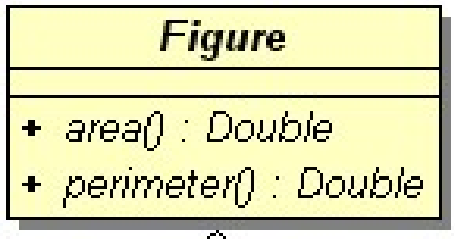
Object slicing

- A se consulta directorul *ObjectSlicing*
- Pentru că **print( )** acceptă un obiect **Person** (în loc de un pointer sau o referinţă), orice apel al funcţiei **print( )** va determina punerea pe stivă a unui obiect de dimensiunea **Person** (care va fi curăţat la sfârşitul funcţiei)
- Dacă un obiect al unei clase moştenite din clasa **Person** (e.g. **Student**) este transmis funcţiei **print( )**, compilatorul îl va accepta, dar va copia pe stivă doar partea corespunzătoare tipului **Person** din acel obiect
  - Se foloseşte constructorul de copiere al clasei **Person**, care iniţializează VPTR cu VTABLE al clasei **Person** şi copiază doar partea corespunzătoare clasei **Person** din acel obiect
- Se feliază şi se pierde partea necomună



Notație UML

- Fonturi italice

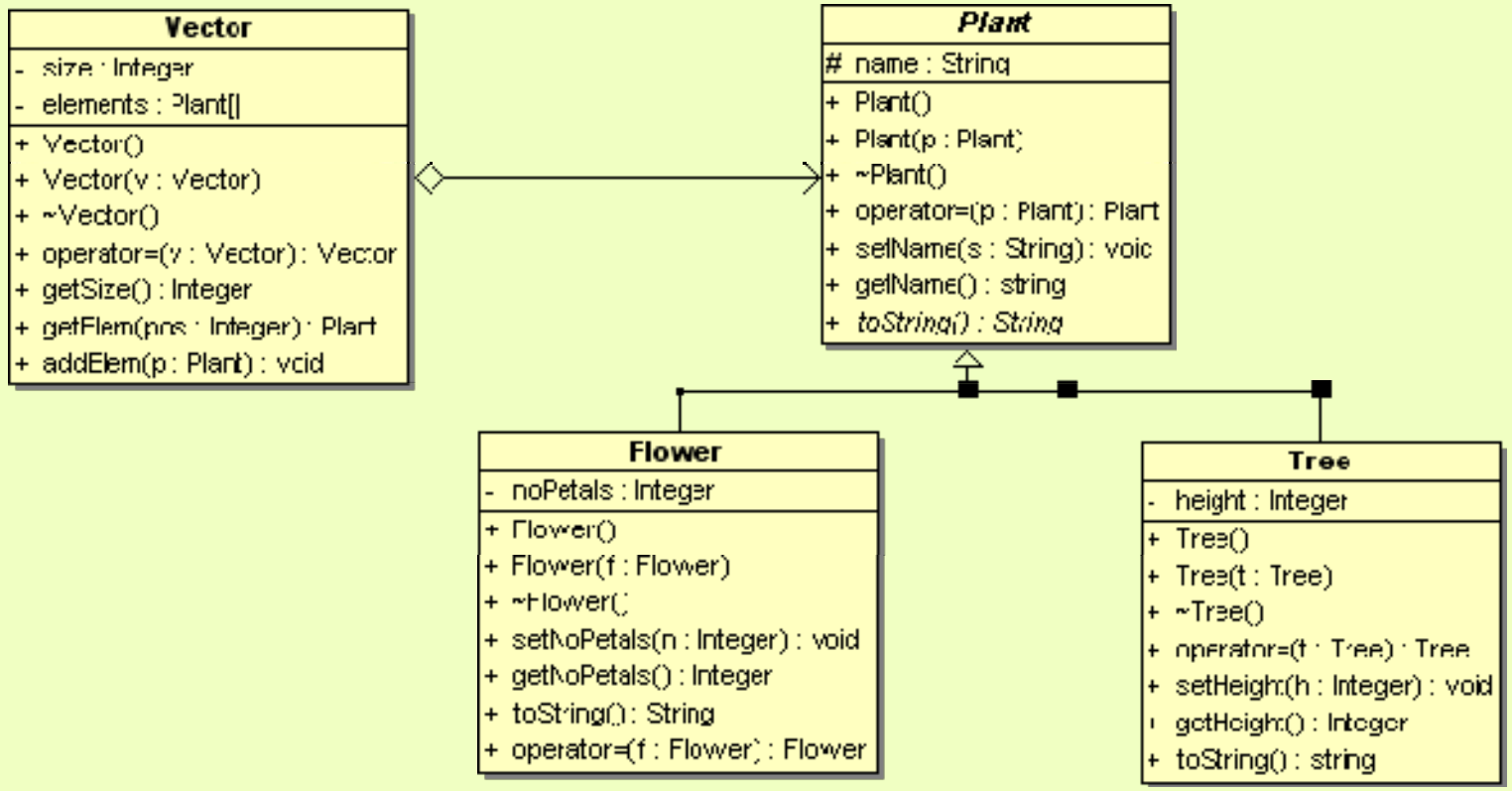


Polimorfism

- Avantaje
  - Reutilizarea codului
  - Extensibilitate
- Funcţii polimorfice
  - Parametrii sunt pointeri la obiecte
- Structuri de date polimorfice
  - void*
  - Clase abstracte

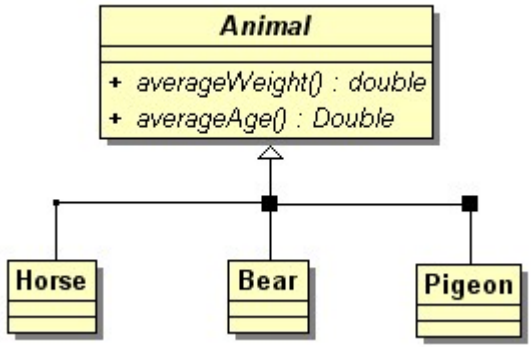
Polimorfism

- A se consulta exemplul din directorul *inheritanceVectorPolymorphic*



Temă

- Implementaţi următoarea diagramă UML:
  - Fiecare animal se caracterizează prin:
    - Greutate şi greutate medie (a acelei categorii)
    - Vârstă şi vârstă medie (a acelei categorii)
  - Creaţi mai mutle animale şi afişaţi informaţiile despre ele.



Cursul următor

- Clase
  - Polimorfism
  - Interfeţe
  - Colecţii cu elemente generice