Programare II
Programare Orientată Obiect

Curs 8

Curs anterior

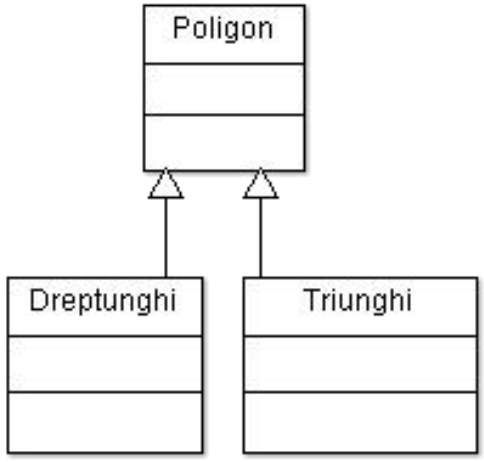
- Moştenire
- Clase derivate
- Modificatori de acces
- Constructori şi destructori
- Supraîncărcarea operatoriilor
- Supradefinirea funcţiilor
- Funcţii virtuale
- Destructori virtuali
- Polimorfism

Curs curent

- Moştenire
- Funcţii pur virtuale
- Clase abstracte
- Moştenire multiplă
- Clase de bază virtuale
- RTTI

Moştenire (inheritance)

- Definiţie
  - Moştenirea este un mecanism care permite unei clase A să moştenească atribute şi metode ale unei clase B. În acest caz obiectele clasei A au acces la membrii clasei B fără a fi nevoie să le redefinim
- Terminologie
  - Clasă de bază
    - Clasa care este moştenită
  - Clasă derivată
    - O clasă specializată a clasei de bază
  - Relaţia „kind-of” nivel de clasă
    - Triunghiul este un tip (kind-of) Poligon
  - Relaţia „is-a” nivel de obiect
    - Obiectul triungiRosu este un (is-a) Poligon



Mostenire. Funcţii virtuale

- Moştenire sintaxă
  class NumeleClaseiDerivate : modificatorDeAccess NumeleClaseiDeBază
  unde
    modificatorDeAcces specifică tipul derivării
    - private (valoare implicită)
    - protected
    - public

- Funcţii virtuale sintaxă
  - virual prototipFuncţie;
  - Cîteva caracteristici
    - Este un tip special de funcţie care determină tipul derivat corespunzător pentru o funcţie cu acelasi prototip
    - Specificarea cuvântului „virtual” în faţa funcţiei
    - Sunt funcţii membre nestatice
    - Redefinirea şi redeclararea funcţiilor virtuale în clasele derivate nu este obligatorie

FUNCȚII PUR VIRTUALE

- Definiţie
  - Sunt funcţii care sunt declarate virtuale, dar nu sunt implementate în clasa de bază
- Trebuie să fie suprascrise în toate clasele derivate, altfel rămân pur virtuale
- Sintaxă
  - virtual tipDeReturn numeFunctie (listaDeParametri) = 0;

CLASE ABSTRACTE

- Definiţie
  - Dacă o clasă conţine o funcţie pur virtuală ea se numeşte abstractă
- Clasele abstracte nu pot fi instanţiate
- Se pot utiliza pointeri la clasele virtuale
- Utile în cazul polimorfismului

CLASE ABSTRACTE. Exemplu

```
class A {
public:
 virtual void x() = 0;
 virtual void y() = 0;
};

class B : public A {
public:
 virtual void x();
};

class C : public B {
public:
 virtual void y();
};
```

Funcţii pur virtuale
=> clasă abstractă

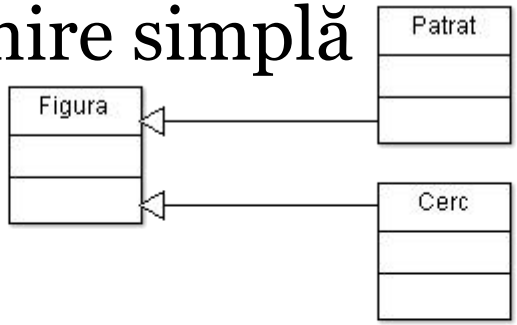
```
nt main () {
 A * ap = new C;
 ap->x ();
 ap->y ();
 delete ap;
 return 0;
```

Implementează doar funcţia virtuală x()
=> clasă abstractă

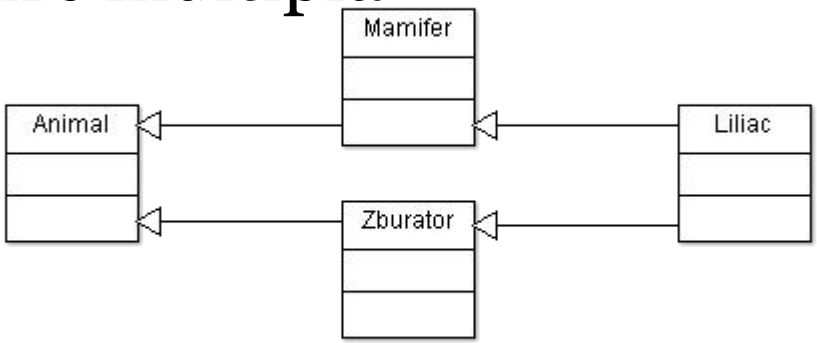
Implementează funcţiile virtuale x() şi y()
=> clasă instanţiabilă

TIPURI DE MOȘTENIRE ÎN C++

- Moștenire simplă



- Moștenire multiplă



MOŞTENIRE MULTIPLĂ

- Definiţie
  - Moştenirea este multiplă dacă o clasă are două sau mai multe clase de bază

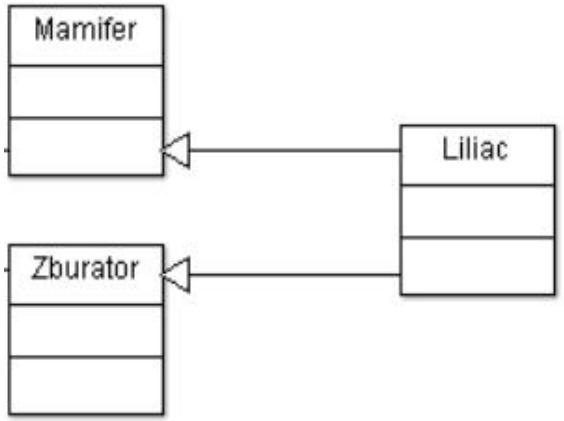
- Sintaxă
  - class ClasaDerivată : [modificatorDeAcces] ClasaDeBaza1,
    [modificatorDeAcces] ClasaDeBaza2,
    ...
    [modificatorDeAcces] ClasaDeBazaN {

    };

- Creşte flexibilitatea ierarhilor de clase → ierarhii în formă de graf

MOȘTENIRE MULTIPLĂ. EXEMPLU

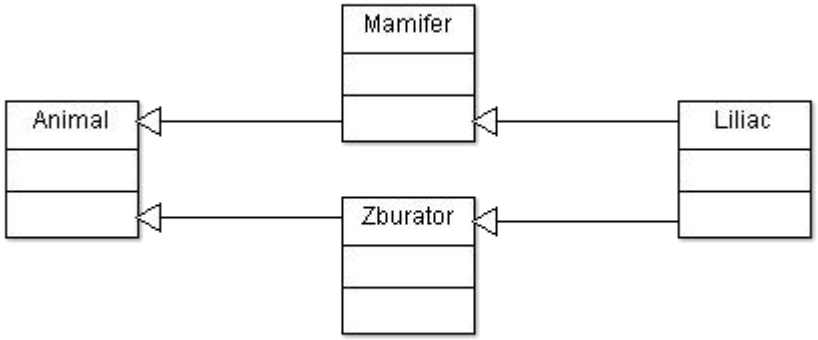
```
class Mamifer {
...
};

class AnimalInaripat {
...
};

class Liliac: public Mamifer, protected AnimalInaripat {
...
Liliac(...):AnimalInaripat(...), Mamifer(...) {
...
}
...
};
```



MOȘTENIRE MULTIPLĂ. EXEMPLU

```
class Animal {
int varsta;
...
};
class Mamifer: public Animal {
...
};
class AnimalInaripat: public Animal {
...
};
class Liliac: public Mamifer, protected
AnimalInaripat {
...
};
```

```
int main(){
  Liliac l;
  l.varsta; //eroare accesare
  ambiguă l.Manifer::varsta=7;
   l.AnimalInaripat=10;
}
```



MOŞTENIRE MULTIPLĂ. EXEMPLU

- Problema: o instanţă a clasei de bază Animal este inclusă de două ori clasa derivată Liliac (una de la clasa Mamifer şi una de la clasa AnimalÎnaripat), ceea ce duce la:
  - Pierderi la alocarea spaţiului de memorie (toate atributele sunt duplicate)
  - Ambiguităţi: probleme de accesare a membrilor clasei Animal

CLASE DE BAZĂ VIRTUALE

- Definiţie
  - Dacă o clasă de bază este declarată ca şi clasă de bază virtuală, atunci într-o ierarhie de tip diamant clasa de bază este instanţiată o singură dată

- Sintaxă

```
class clasaDerivată : [modificatorDeAcces] virtual
clasaDerivata {
...
}
```

CLASE DE BAZĂ VIRTUALE

```
class Animal { … };

class Mamifer: public virtual Animal { … };

class AnimalInaripat: public virtual Animal { … };

class Liliac: public Mamifer, protected AnimalInaripat
{
  Liliac(…):Animal(…), Manifer(…), AnimalInaripat(…)
{
     …….
  }
…
};
----------
Liliac l;
l.varsta;
```

- Constructorul clasei de bază trebuie apelat explicit
- Paşi pentru iniţializarea unui obiect:
  - Apelează constructorul clasei de bază virtuale
  - Apelează constructorii claselor de bază în ordinea declarări lor
  - Iniţializarea membrilor clasei derivate
  - Iniţializarea obiectului derivat

RUNTIME TYPE INFORMATION - RTTI

- Este o facilitate a limbajului care permite identificarea tipului variabilelor la execuţie
- Pentru a funcţiona clasele trebuie să fie polimorfice, să conţină cel puţin o funcţie virtuală
- Include
  - dynamic_cast<>
  - typeid
- Incluse in biblioteca typeinfo
  - #include <typeinfo>

RTTI – DYNAMIC CAST

```
class Animal {
virtual ~Animal();
};
class Mamifer : public Animal {
virtual ~Mamifer();
};

int main() {
Animal a;
Mamifer m;
Animal *pa =&m;
if (dynamic_cast<Mamifer*> (pa) != 0) {
Mamifer *pm = (dynamic_cast<Mamifer*> (pa) ;
Mamifer *p1=(Mamifer*)pa;
}
}
```

CURS VIITOR

- Şabloane