OOP. Laboratorul 6

Prof. Marian Lupașcu. Document editat de Tudor Raluca

Martie 2020

Moștenire

Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

În clasa derivată

- Când construiesc derivata se apelează prima dată constructorul bazei și abia după constructorul derivatei (prima dată se construiește partea bazei - se păstrează în această ordine - baza + derivata după).
- Putem apela **constructor** și **pentru clasa de bază** doar în LISTA DE INIȚIALIZARE. *(Notă: este un fel de super keyword din Java, dar care se scrie doar pe prima linie.)*
- Dacă nu punem în lista de inițializare B(..) înainte să începem să setam datele membre pentru clasa derivată, atunci se ia constructorul fara parametri(!!!) din clasa de bază.
- Dacă nu avem constructor fără parametri și nu apelam nimic in clasa derivată, atunci nu știe și crapă!!! (no matching for B b)

Alte observaţii importante

Dacă moștenim mai multe clase, ordinea de apelare a constructorilor importantă ESTE CEA DIN PRIMA LINIE!!!

```cpp
#include <bits/stdc++.h>

using namespace std;

class B1 {
public:
    B1() {
        cout << "B1\n";
    }
};

class B2 {
public:
    B2() {
        cout << "B2\n";
    }
};

// ORDINEA DE APELARE A CONSTRUCTORILOR E CEA DE AICI
class D : public B1, public B2 { // ORDINEA ACEASTA ESTE SFANTA
public:
    D() : B2(), B1(){ // nu conteaza ordinea de aici, dar este fff util
        cout << "D\n";
    }
};

int main() {
    D d; // at! ordinea - B1, B2, D
    return 0;
}
```

Listing 1: Ordinea de apelare a constructorilor - moştenire multiplă

Object Slicing

B
|
D

D d;
B b = (B)d; // deci așa merge doar în ierarhie!!

=> Când îmi construiesc obiectul din clasa derivata, mai întâi se construiește partea clasei de bază și putem să extragem doar acea parte prin object slicing.

La examen - Atenție la obiectele globale!!

```
class C{
public:
  C(){cout << "CC\n";}
};

class B{
public:
  B(){cout << "CB\n";}
}c;

class D : public B{
public:
    D(){cout << "CD\n";}
    C c;
};

class E : public D{
public:
    E(){cout << "CE\n";}
}b;

int main()
{
    E e1, e2;
    B b;
}
```

Listing 2: Subiect Examen

Compilează? DA
Ce se va afişa?
Vezi fişierul exemplu-examen-1.cpp pentru explicaţie.