POO - Moștenire

Moștenire

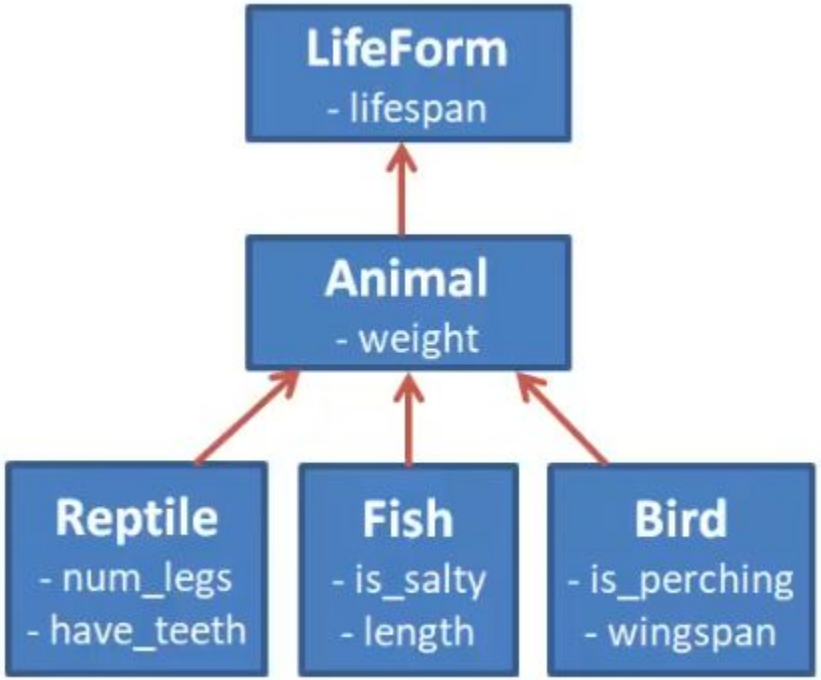
Moștenire - concept

Unul dintre cei 4 piloni ai POO. Prin moștenire putem să creăm o clasă care să aibă la bază o altă clasă, care să ne permită să ne folosim de membrii primei clase. Moștenirea este folosită pentru a nu avea cod duplicat.

Moștenirea are la bază conceptul de apartenență (IS-A).

Astfel se introduc 2 concepte noi:

- Subclasă (sub class) - clasa care moștenește de la o altă clasă (derived class, child)
- Superclasă (superclass) - clasa de la care se moștenește (base class, parent)


LifeForm
- lifespan
Animal
- weight
Reptile
- num_legs
- have_teeth
Fish
- is_salty
- length
Bird
- is_perching
- wingspan

Tipuri de Moșteniri

Single Inheritance

class subclass_name : access_mode
base_class_name

{

//body of subclass

};

```
#include <stdio.h>
#include <iostream>
#include <string.h>

using namespace std;

class Base{
public:
    void hello(){
        cout << "Hello from the Base side" << endl;
    };
};

class Derived : public Base{
public:
    void hello2(){
        cout << "Hello from the Derived side" << endl;
    };
};

int main(){
    Base clasa1;
    clasa1.hello();
    Derived clasa2;
    clasa2.hello();
    clasa2.hello2();
    return 0;
}
```

Multilevel Inheritance

O clasă poate fi creată dintr-o clasă care a fost la rândul ei derivată din altă clasă.

```
8  #include <stdio.h>
9  #include <iostream>
10 #include <string.h>
11
12 using namespace std;
13
14 class GrandParent{
15 public:
16     int age;
17     string name;
18     void show(int age , string name){
19         this->age = age;
20         this->name = name;
21         cout << this->age << " " << this->name << endl;
22     }
23 };
24
25 class Parent : public GrandParent{
26 };
27
28 class Child : public Parent{
29 };
30
31 int main(){
32     GrandParent bunic;
33     Parent parinte;
34     Child copil;
35     bunic.show(90, "Popescu");
36     parinte.show(60, "Moraru");
37     copil.show(15, "Moraru");
38     return 0;
39 }
```

Multiple Inheritance

O clasă poate moșteni 2 clase.

class subclass_name : access_mode base_class_name , access_mode base_class name

{

  //body of subclass

};

```
8  #include <stdio.h>
9  #include <iostream>
10 #include <string.h>
11
12 using namespace std;
13
14 class Father{
15 public:
16     string eyeColor = "Caprui";
17 };
18
19 class Mother{
20 public:
21     string hairColor = "Brunet";
22 };
23
24 class Child : public Father , public Mother{
25
26 };
27
28 int main(){
29     Child copil;
30     cout << copil.eyeColor << " " << copil.hairColor << endl;
31     return 0;
32 }
```

În Java nu există multiple inheritance.

Constructori

Constructori

Prin moștenire, clasa de derivată nu preia constructorii, copy constructorul sau destructorul clasei de bază.

Apelurile de constructor sunt înlănțuite, ceea ce înseamnă că înainte de a se inițializa obiectul copil, mai întâi se va inițializa obiectul părinte. În cazul în care părintele este copil la rândul lui, se va inițializa părintele lui (până se va ajunge la părintele suprem – root).

Dacă nu avem constructor default atunci trebuie să folosim următoarea structură:

```
class subclass_name : access_mode base_class_name, access_mode base_class
{
  public:
        sublcass_name (type param1, type param2): base_class_name{param1},base_class_name{param2}{
        }
};
```

```
14 class Father{
15 public:
16     Father(string x){
17         cout << "Constructor tata a fost apelat" << endl;
18         eyeColor = x;
19     }
20 protected:
21     string eyeColor;
22 };
23
24 class Mother{
25 public:
26     Mother(string x){
27         cout << "Constructor mama a fost apelat" << endl;
28         hairColor = x;
29     }
30 protected:
31     string hairColor;
32
33 };
34
35 class Child : public Father , public Mother{
36 public:
37
38     Child(string x , string y):Father{x}, Mother{y}{
39         cout << "Constructor copil a fost apelat" << endl;
40         cout << this->eyeColor << " " << this->hairColor;
41     }
42 };
43
44 int main(){
45     Child copil("Caprui","Brunet");
46     return 0;
47 }
```

Modificări în caz că avem constructor care nu e default

Specificatori de acces (Access specifiers)

Specificatori de access

Atunci când o clasă moștenește pe o alta membrii sunt accesibili astfel:

- Public: nu are restricții
- Protected: în interiorul clasei părinte și clasei copil
- Private: doar în interiorul clasei

Exemplu specificator de acces

```cpp
using namespace std;

class Human {
public:
    int legsNumber = 0;
protected:
    string country = "Default";
private:
    string name = "Default";
};

class Romanian : public Human{ //inheritence
public:
    string getCountry(){
        return this->country;
    }
    string city;
};

int main(){
    Human om;
    om.legsNumber = 2;
    Romanian roman;
    roman.city = "Bucharest";
    cout << roman.getCountry() << endl;
    cout << roman.city << endl;
    cout << roman.legsNumber << endl;
    roman.legsNumber = 1;
    cout << om.legsNumber << endl;
    cout << roman.legsNumber << endl;
    return 0;
}
```

Tip de moștenire

Specificator de acces la clasa de bază	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Ascuns	Ascuns	Ascuns

Tipuri de moșteniri

```cpp
#include <stdlib.h>
#include <iostream>

using namespace std;

class One
{
public:
    int a;
protected:
    int b;
private:
    int c;
};

class Two : public One
{
    // a  public
    // b  protected
    // c  not accessible from Two
};

class Three : protected One
{
    // a protected
    // b protected
    // c not accessible from Three
};

class Four : private One    // 'private' is default for classes
{
    // a private
    // b private
    // c not accessible from Four
};
```

Summary

De Reținut

Unul dintre cei 4 piloni ai POO. Prin moștenire putem să creăm o clasă care să aibă la bază o altă clasă, care să ne permită să ne folosim de membrii primei clase. Moștenirea este folosită pentru a nu avea cod duplicat.

Subclasă (sub class) - clasa care moștenește de la o altă clasă (derived class, child)

Superclasă (superclass) - clasa de la care se moștenește (base class, parent)

Prin moștenire, clasa de derivată nu preia constructorii, copy constructorul sau destructorul clasei de bază.

Moștenire prin public membrii rămân la fel, protected transforma public în protected, iar private transformă public și protected în private.

Exerciții

Exerciții

1. **6p** Realizați o clasă Wine, care să conțină atributele *type* și *origin* (ambele string și să fie protected). Clasa trebuie să aibă:

- Un constructor default
- Un constructor care va inițializa toți parametrii
- Metode de get pentru cei doi parametrii

Din clasa Wine derivați alte 3 clase printr-o moștenire private. Fiecare va avea un tip diferit de etichetă pe sticlă

- Tipul A: etichetă pătrată (private *length*)
- Tibul B: etichetă rotundă și transparentă (private *radius*)
- Tipul C: în formă de triunghi dreptunghic (private *base* și *height*)

Fiecare clasă va avea constructori default și constructori cu parametrii care să seteze fiecare atribut.

Fiecare clasă trebuie să aibă o metodă de a întoarce aria etichetei, a tipului și a originii.

2. **3p** Modelați din natură un exemplu de moștenire multiplă (multiple inheritance) care să conțină constructor default.