

Buổi 4

Tính kế thừa

I. Đặt vấn đề

Giả sử cần giải quyết bài toán sau:

- Cài đặt class Động vật gồm thông tin là Tên và Tuổi
- Cài đặt class Chó gồm các thông tin Tên, Tuổi và Loại chó.
- Cả 2 đều có phương thức nhập và xuất thông tin.

Nếu cài đặt theo cách bình thường, chúng ta sẽ cài đặt như sau:

- Động vật:

```
class DongVat
{
    public string Ten { get; set; }

    public int Tuoi { get; set; }

    public void Nhap()
    {
        Console.Write("Nhập tên: ");
        Ten = Console.ReadLine();
        Console.Write("Nhập tuổi: ");
        Tuoi = Convert.ToInt32(Console.ReadLine());
    }

    public void Xuat()
    {
        Console.WriteLine($"{Ten} - {Tuoi}");
    }
}
```

- Chó:

```
class Cho
{
    public string Ten { get; set; }

    public int Tuoi { get; set; }

    public string Loai { get; set; }
```

```

public void Nhap()
{
    Console.Write("Nhập tên: ");
    Ten = Console.ReadLine();
    Console.Write("Nhập tuổi: ");
    Tuoi = Convert.ToInt32(Console.ReadLine());
    Console.Write("Nhập loại chó: ");
    Loai = Console.ReadLine();
}

public void Xuat()
{
    Console.WriteLine($"{Ten} - {Tuoi} - {Loai}");
}
}

```

Có thể thấy trong cách cài đặt ở trên, 2 class **DongVat** và **Cho** rất giống nhau. Về cơ bản, class **Cho** chỉ bổ sung thêm 1 thuộc tính **Loai** so với class **DongVat**. Nếu cài đặt như cách ở trên sẽ dài dòng, không tái sử dụng được code.

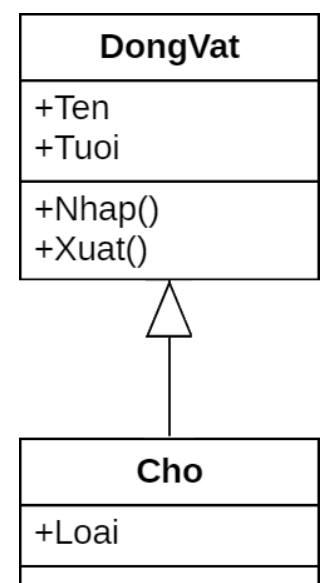
Việc giống nhau ở 2 class trên cũng đúng về mặt ngữ nghĩa: chó là 1 loài động vật. Tức là nếu xem động vật là 1 class chung, thì về cơ bản chó sẽ có tất cả thông tin, chức năng như của động vật, nhưng sẽ có thêm những thông tin, chức năng riêng của loài chó.

II. Tính kế thừa

Để thể hiện mối quan hệ giữa các class có quan hệ về mặt ngữ nghĩa cũng như có các điểm chung, lập trình hướng đối tượng hỗ trợ Tính kế thừa (**Inheritance**), cho phép cài đặt 1 lớp B có những thuộc tính, phương thức giống với 1 lớp A đã có, và bản thân lớp B cũng có thể có thuộc tính, phương thức riêng của nó.

Trong ví dụ trên, class **Cho** sẽ kế thừa từ class **DongVat**. Với mối quan hệ này, chúng ta vẽ lược đồ class (class diagram) như hình bên:

- Lớp **DongVat** được gọi là lớp cha (**parent class**) hoặc là lớp cơ sở (**base class**).
- Lớp **Cho** được gọi là lớp con (**child class**) hoặc là lớp dẫn xuất (**derived class**).
- Lớp **Cho** sẽ kế thừa các thuộc tính **Ten** và **Tuoi** từ lớp **DongVat** mà không cần phải khai báo lại. Ngoài ra, nó còn có thêm thuộc tính **Loai** của riêng nó.



Chúng ta sẽ cài đặt class **Cho** như sau:

```
class Cho : DongVat
{
    public string Loai { get; set; }

    public void Nhap()
    {
        Console.WriteLine("NHẬP THÔNG TIN CHÓ");
        base.Nhap();
        Console.Write("Nhập loại chó: ");
        Loai = Console.ReadLine();
    }

    public void Xuat()
    {
        base.Xuat();
        Console.WriteLine($" - {Loai}");
    }
}
```

Trong đó:

- Để biểu diễn sự kế thừa, chúng ta dùng ký hiệu : _____
- Phương thức **Nhap()** và **Xuat()** của class **Cho** có gọi lại phương thức cùng tên của lớp cha – **DongVat** thông qua từ khóa **base.**

Tính kế thừa được sử dụng khi:

- Cần tái sử dụng mã nguồn của 1 class đã có.
- Cần mở rộng chức năng của 1 class đã có.

III. Phạm vi truy cập trong tính kế thừa

Khi 1 thành phần của class (thuộc tính, phương thức, property...) được khai báo với phạm vi **protected**, thành phần này sẽ được kế thừa bởi các class con của class này:

Access Modifier	Truy cập trong class	Truy cập ngoài class	Có thể kế thừa
private	✓	✗	✗
protected	✓	✗	✓
public	✓	✓	✓

Ví dụ: Cho class **Parent** và **Child** như sau. Class **Child** sẽ kế thừa 2 thuộc tính **lastName** và **money** cùng với phương thức **Greet()** của class **Parent**:

```
class Parent
{
    private string firstName;    // Không kế thừa

    public string lastName;

    protected int money;

    private void GoToWork() { }    // Không kế thừa

    public void Greet() { }
}

class Child : Parent { }
```

IV. Mối quan hệ giữa các class

Để có thể xây dựng được mối quan hệ kế thừa giữa 2 class A và B một cách chính xác, cần trả lời các câu hỏi sau đây:

- A và B có mối quan hệ với nhau hay không?
- Class nào là trường hợp tổng quát, class nào là trường hợp cụ thể/đặc biệt?

Đối với class là trường hợp tổng quát sẽ trở thành lớp cha, class là trường hợp cụ thể/đặc biệt sẽ trở thành lớp con. Đây được gọi là mối quan hệ tổng quát hóa (**generalization**).

Ví dụ:

Mối quan hệ	Lớp cha	Lớp con
Hình vuông là 1 trường hợp đặc biệt của hình chữ nhật	Hình chữ nhật	Hình vuông
Xe máy là 1 trường hợp cụ thể của phương tiện giao thông	Phương tiện giao thông	Xe máy
Trưởng phòng là 1 trường hợp cụ thể của nhân viên	Nhân viên	Trưởng phòng

Lưu ý: Không nên xây dựng mối quan hệ kế thừa chỉ vì muốn tái sử dụng một số thành phần của 1 class đã có, mà vẫn cần phải cân nhắc mối quan hệ về ngữ nghĩa giữa 2 class này theo 2 câu hỏi ở trên.

Giả sử cần xây dựng class Hình chữ nhật và Hình vuông với thuộc tính là độ dài cạnh và 2 phương thức tính chu vi và tính diện tích. Nếu phân tích rằng Hình vuông chỉ cần lưu

trữ độ dài 1 cạnh, còn Hình chữ nhật cần lưu trữ độ dài 2 cạnh mà xây dựng mối quan hệ Hình chữ nhật kế thừa từ Hình vuông (tức là Hình vuông là lớp cha, Hình chữ nhật là lớp con) thì sẽ không đúng về mặt ngữ nghĩa:

```
class HìnhVuong
{
    public double A { get; set; }

    public double ChuVi => A * 4;

    public double DiệnTich => A * A;

    public void Nhap()
    {
        Console.Write("Nhập cạnh A: ");
        A = Convert.ToDouble(Console.ReadLine());
    }
}

class HìnhChuNhat : HìnhVuong
{
    public double B { get; set; }

    public double ChuVi => (A + B) * 2;

    public double DiệnTich => A * B;

    public void Nhap()
    {
        base.Nhap();
        Console.Write("Nhập cạnh B: ");
        B = Convert.ToDouble(Console.ReadLine());
    }
}
```

Đối với cách cài đặt mối quan hệ kế thừa như ở trên, ngoài việc Hình chữ nhật tái sử dụng thuộc tính cạnh A của Hình vuông, chúng ta không tái sử dụng được thêm bất kỳ thứ gì khác (kể cả việc tính chu vi và diện tích).

Trong khi đó, Hình vuông là 1 trường hợp đặc biệt của Hình chữ nhật, hay nói cách khác, Hình vuông cũng có thể coi là Hình chữ nhật, do đó cách tính chu vi và diện tích của Hình chữ nhật vẫn áp dụng được cho Hình vuông.

Vậy, cách phân tích đúng như sau: Hình vuông sẽ kế thừa từ Hình chữ nhật (tức là Hình chữ nhật là lớp cha, Hình vuông là lớp con):

```

class HìnhChuNhat
{
    public double A { get; set; }

    public double B { get; set; }

    public double ChuVi => (A + B) * 2;

    public double DienTich => A * B;

    public void Nhap()
    {
        Console.Write("Nhập chiều dài: ");
        A = Convert.ToDouble(Console.ReadLine());
        Console.Write("Nhập chiều rộng: ");
        B = Convert.ToDouble(Console.ReadLine());
    }
}

class HìnhVuong : HìnhChuNhat
{
    public double B { get; set; }

    public void Nhap()
    {
        Console.Write("Nhập cạnh: ");
        A = Convert.ToDouble(Console.ReadLine());
        B = A;
    }
}

```

Với cách cài đặt này, class Hình vuông sẽ tái sử dụng được cách tính chu vi và diện tích từ class Hình chữ nhật.

V. Kế thừa phương thức khởi tạo

Giả sử có 2 class như sau:

- Class `Parent` có thuộc tính `Name` và phương thức khởi tạo mặc định, phương thức khởi tạo kèm tham số:

```

class Parent
{
    public string Name { get; set; }
    public Parent()
    {
        Name = "Anonymous";
    }
}

```

```

    public Parent(string name)
    {
        Name = name;
    }
}

```

- Class **Child** kế thừa từ **Parent**, có thêm thuộc tính **Money** và phương thức khởi tạo mặc định, phương thức khởi tạo kèm tham số:

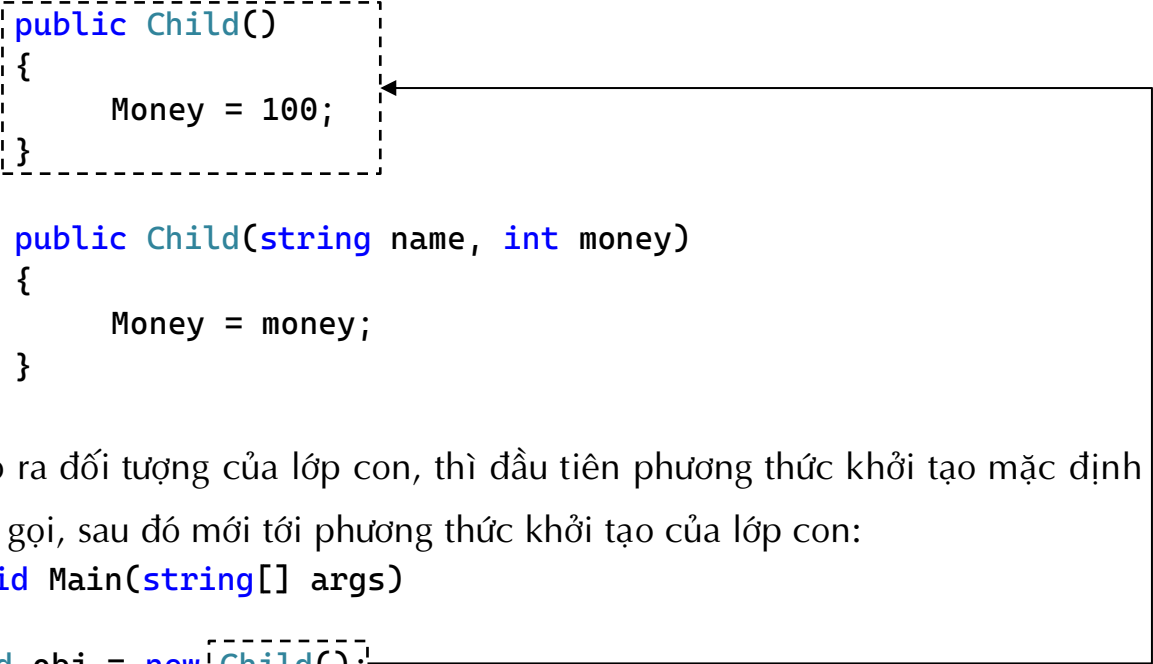
```

class Child : Parent
{
    public int Money { get; set; }

    public Child()
    {
        Money = 100;
    }

    public Child(string name, int money)
    {
        Money = money;
    }
}

```



Nếu chúng ta tạo ra đối tượng của lớp con, thì đầu tiên phương thức khởi tạo mặc định của lớp cha sẽ được gọi, sau đó mới tới phương thức khởi tạo của lớp con:

```

static void Main(string[] args)
{
    Child obj = new Child();
    Console.WriteLine(obj.Name); // Xuất ra Anonymous
    Console.WriteLine(obj.Money); // Xuất ra 100
}

```

Trong ví dụ trên, mặc dù chúng ta tạo ra đối tượng **obj** bằng phương thức khởi tạo mặc định của class **Child**. Phương thức này chỉ khởi tạo thuộc tính **Money** là 100, nhưng thuộc tính **Name** của nó vẫn có giá trị là *Anonymous*. Điều này chứng minh là khi đối tượng **obj** được tạo ra thì phương thức khởi tạo mặc định của class **Parent** đã được gọi.

Giả sử chúng ta dùng phương thức khởi tạo kèm tham số để tạo ra đối tượng của lớp con như sau:

```

static void Main(string[] args)
{
    Child obj = new Child("Ricky", 20000);
    Console.WriteLine(obj.Name); // Xuất ra Anonymous
    Console.WriteLine(obj.Money); // Xuất ra 20000
}

```

Trong ví dụ trên, chúng ta tạo ra đối tượng `obj` có 2 thông tin là *Ricky* và *20000*, nhưng do trong quá trình tạo đối tượng `obj`, phương thức khởi tạo mặc định của lớp cha được gọi, nên thuộc tính `Name` có giá trị là *Anonymous*.

Để phương thức khởi tạo kèm tham số của lớp con có thể kế thừa từ phương thức khởi tạo kèm tham số của lớp cha, chúng ta khai báo kèm từ khóa `base` như sau:

```
public Child(string name, int money) : base(name)
{
    Money = money;
}
```

Gọi phương thức khởi tạo
kèm tham số của lớp cha

Với cách khai báo như thế này, phương thức khởi tạo kèm tham số của lớp con sẽ gọi phương thức khởi tạo kèm tham số của lớp cha trước khi thực hiện. Ngoài ra, chúng ta cũng có thể điều khiển được phương thức khởi tạo của lớp con sẽ kế thừa từ phương thức khởi tạo nào của lớp cha.

VI. Sealed class

Khi 1 class được khai báo kèm từ khóa `sealed`, class này không thể được kế thừa:

```
sealed class Vehicle { ..... }

class Car : Vehicle { ..... } // Lỗi: 'Car': cannot derive from
                               // sealed type 'Vehicle'
```

VII. Bài tập

Thực hiện các bài tập sau, mỗi bài nằm trong 1 project của solution **OOP_Buoi05**. Có thể sử dụng Object Initializer để tạo đối tượng kèm giá trị thay vì nhập từ bàn phím.

Bài 1: Viết chương trình cho phép người dùng chọn và nhập vào thông tin độ dài cạnh của 1 hình chữ nhật hoặc 1 hình vuông. Xuất ra màn hình thông tin hình vừa nhập, kèm theo chu vi và diện tích của nó.

Bài 2: Viết chương trình cho phép người dùng chọn và nhập vào thông tin của 1 hình elip hoặc 1 hình tròn:

- Thông tin của hình elip là độ dài 2 bán trục
- Thông tin của hình tròn là độ dài bán kính

Xuất ra màn hình thông tin hình vừa nhập, kèm theo chu vi và diện tích của nó.

Gợi ý: Công thức tính chu vi và diện tích của hình elip lần lượt là:

Chu vi	Diện tích
$C_{\text{elip}} = 2\pi \sqrt{\frac{r_1^2 + r_2^2}{2}}$	$S_{\text{elip}} = \pi r_1 r_2$

Bài 3: Trong công ty ABC có 3 loại nhân viên sau:

- Nhân viên thường: có các thông tin Họ tên, Tuổi, Số ngày công, Lương cơ bản. Lương của nhân viên được tính như sau:

$$\text{Số ngày công} \times \text{Lương cơ bản.}$$

- Trưởng phòng: có các thông tin Họ tên, Tuổi, Số ngày công, Lương cơ bản, Phụ cấp chức vụ. Lương của trưởng phòng được tính như sau:

$$\text{Số ngày công} \times \text{Lương cơ bản} + \text{Phụ cấp chức vụ}$$

- Giám đốc: có các thông tin Họ tên, Tuổi, Số ngày công, Lương cơ bản, Phụ cấp chức vụ và thâm niên. Lương của Giám đốc được tính như sau:

$$30.000.000 + \text{Phụ cấp chức vụ} + \text{Thưởng}$$

$$\text{Nếu thâm niên} \geq 10 \text{ năm thì thưởng } 10.000.000$$

$$\text{Ngược lại thưởng } 3.000.000$$

Viết chương trình cho phép người dùng nhập vào thông tin của 1 nhân viên trong công ty. Xuất ra màn hình tất cả thông tin của nhân viên đó, kèm theo mức lương.

Bài 4: Công ty điện lực ABC cần quản lý thông tin khách hàng thuộc 1 trong 2 loại sau:

- Khách hàng Việt Nam: có các thông tin Mã khách hàng, Họ tên, Ngày lập hóa đơn, Đối tượng khách hàng (Sinh hoạt, Sản xuất, Kinh doanh), Số điện tiêu thụ (kW), Đơn giá (đ/kW), Định mức (kW). Cách tính tiền điện:

- Nếu số điện \leq Định mức: Thành tiền = Số điện \times Đơn giá

- Ngược lại: Mỗi kW vượt định mức tính giá gấp 2.5 lần.

- Khách hàng nước ngoài: có các thông tin Mã khách hàng, Họ tên, Ngày lập hóa đơn, Quốc tịch, Số điện tiêu thụ (kW), Đơn giá (đ/kW). Cách tính tiền điện:

$$\text{Thành tiền} = \text{Số điện} \times \text{Đơn giá} \times 2.5$$

Viết chương trình cho phép người dùng nhập vào thông tin của 1 khách hàng. Xuất ra màn hình tất cả thông tin của khách hàng đó, kèm theo tiền điện.