

CHƯƠNG 2

LỚP VÀ ĐỐI TƯỢNG

GVHD: TS. GVC LÊ THỊ THÚY NGÀ
BỘ MÔN: ĐIỀU KHIỂN HỌC

NỘI DUNG

2.1 Lớp và đối tượng

2.2 Tạo và sử dụng đối tượng

2.3 Các kỹ thuật trong xây dựng lớp

2.4 Kết tập và kế thừa

2.5 Các kiểu dữ liệu tập hợp trong C#

2.6 Ngoại lệ và xử lý ngoại lệ

2.4 KẾT TẬP VÀ KẾ THỪA

❖ Kết tập (Aggregation):

- Kết tập là tạo ra lớp mới trên cơ sở tập hợp hoặc sử dụng các đối tượng của lớp cũ đã có.
- Bản chất của kết tập:
 - Các thành phần của lớp mới (lớp toàn thể) là các đối tượng của lớp có sẵn (lớp thành phần).
 - Kết tập tái sử dụng thông qua đối tượng: Tái sử dụng các thành phần dữ liệu và các hành vi của lớp thành phần thông qua đối tượng thành phần.

2.4 KẾT TẬP VÀ KẾ THỪA

```
class diem
{
    public float x, y;
    4 references
    public diem(float x1, float y1)
    {
        x = x1;
        y = y1;
    }

    4 references
    public void hienthidiem()
    {
        Console.WriteLine("{0},{1}",x,y);
    }
}
```

```
class tugiac
{
    public diem p1, p2, p3,p4;
    1reference
    public tugiac(diem a, diem b, diem c, diem d)
    {
        p1 = a;
        p2 = b;
        p3 = c;
        p4 = d;
    }
    1reference
    public void hienthitugiac()
    {
        p1.hienthidiem();
        p2.hienthidiem();
        p3.hienthidiem();
        p4.hienthidiem();
    }
}
```

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        diem p1 = new diem(0, 0);
        diem p2 = new diem(1, 0);
        diem p3 = new diem(1, 1);
        diem p4 = new diem(0, 1);
        tugiac tg = new tugiac(p1, p2,p3,p4);
        tg.hienthitugiac();
        Console.ReadLine();
    }
}
```

```
(0,0)
(1,0)
(1,1)
(0,1)
```

2.4 KẾT TẬP VÀ KẾ THỪA

❖ Kế thừa (Inheritance):

- Kế thừa là tạo ra lớp mới trên cơ sở kế thừa từ lớp cũ đã có.
- Bản chất của kế thừa:
 - Tạo lớp mới bằng cách phát triển lớp đã có.
 - Lớp mới kế thừa những gì đã có trong lớp cũ và phát triển những tính năng mới.
- Lớp cũ: Lớp cha (parent, superclass), lớp cơ sở (**base class**).
- Lớp mới: Lớp con (child, subclass), lớp dẫn xuất (**derived class**).

2.4 KẾT TẬP VÀ KẾ THỪA

- Cú pháp kế thừa:

```
class tên_lớp_dẫn_xuất : tên_lớp_cơ_sở
```

Ví dụ: lớp SVĐH kế thừa từ lớp SV

```
class SVĐH : SV
```

2.4 KẾT TẬP VÀ KẾ THỪA

- *Từ khóa truy cập “protected” trong kế thừa:*

Từ khóa **protected** dùng để khai báo mức truy cập đối với dữ liệu thành viên của lớp theo ý nghĩa chỉ cho phép các lớp kế thừa (Derived class) từ lớp cơ sở (Base class) được quyền tác động đến những thành viên thuộc dạng này còn những lớp không thuộc cùng phả hệ thì không được phép. Nói cách khác, đối với những biến (Variable), hàm hoặc phương thức (Method) được khai báo truy cập bởi từ khóa **protected** thì tại lớp kế thừa, chúng ta có thể truy cập đến những thành viên thuộc dạng này giống như cách truy cập đối với những thành viên được khai báo **public** còn trong những lớp không phải là lớp kế thừa từ lớp cơ sở thì không được phép (trong tình huống này thì khai báo **protected** mang ý nghĩa giống như **private**).

■ Ý nghĩa của từ khóa “base” trong kế thừa:

Đây là từ khóa cho phép bạn có thể sử dụng để truy xuất đến các biến, các phương thức có trong lớp cơ sở mà lớp hiện tại đã kế thừa từ nó. Cú pháp cho phép truy cập đến thành viên trong lớp cơ sở từ lớp kế thừa được mô tả như sau:

```
class baseName
```

```
{    ...
```

```
AccessModifier return_type Base_Method (list_of_argument)
```

```
    {    }
```

```
}
```

```
class derivedName : baseName
```

```
{    base.Base_Method( ... ); // Truy xuất đến phương thức của lớp cơ sở nhờ từ khóa base và toán tử “.”
```


■ Cơ chế gọi thực thi hàm khởi tạo (constructor) trong kế thừa:

C# không cho phép kế thừa hàm khởi tạo từ lớp cơ sở giống như các phương thức thông thường khác. Tại hàm khởi tạo của lớp dẫn xuất, nếu muốn thực thi lại các lệnh trong hàm khởi tạo nào của lớp cơ sở thì bắt buộc bạn phải gọi hàm khởi tạo tương ứng (vì trong một lớp có thể có nhiều hơn một hàm khởi tạo) thông qua từ khóa **base** và toán tử “:” tại dòng khai báo hàm khởi tạo của lớp kế thừa theo cú pháp sau:

```
public deriveClassConstructor(): base(tham_số)  
  
{ ... }
```

■ Ghi đè (Overriding) một phương thức trong kế thừa:

- Overriding là một tính năng có ở hầu hết các ngôn ngữ lập trình hướng đối tượng. Trong C Sharp cũng thế, tính năng này cho phép lập trình viên có thể định nghĩa lại phương thức có trong lớp cơ sở tại lớp kế thừa. Nói cách khác, với cùng 1 phương thức đã định nghĩa trong lớp cơ sở, chúng ta có thể sử dụng lại tại lớp kế thừa bằng cách thay đổi “hành vi” của nó để phù hợp hơn đối với đối tượng của lớp kế thừa.
- Tính năng này đáp ứng cho bài toán : “Làm sao để tái sử dụng mã lệnh của lớp cơ sở (Base class) tại lớp kế thừa (Derived class) trong khi vẫn bảo toàn được những ưu điểm của việc kế thừa”

■ ***Ghi đè (Overriding) một phương thức trong kế thừa:***

- Khi thực hiện kỹ thuật Overriding phương thức tại lớp kế thừa, cần phải quan tâm đến phạm vi truy cập (Access modifier) của phương thức. Có nghĩa là khi 1 phương thức được khai báo với AccessModifier là **private** ở lớp cơ sở thì tại lớp kế thừa, bạn không thể **override** nó với AccessModifier là **public**.
- Trong C Sharp, để có thể sử dụng kỹ thuật overriding cho 1 phương thức thì tại lớp cơ sở, phương thức đó phải được khai báo là **virtual**. Một phương thức được khai báo với từ khóa **virtual** ở phía trước thì phương thức đó được gọi là phương thức ảo (virtual method) và nếu ở lớp cơ sở có chứa phương thức ảo thì tại lớp kế thừa, bắt buộc phải định nghĩa lại phương thức đó thông qua việc sử dụng từ khóa **override** ở phía trước.

■ *Ghi đề (Overriding) một phương thức trong kế thừa:*

- Nếu **override** 1 phương thức không phải là **virtual** ở lớp cơ sở thì khi biên dịch, C# sẽ đưa ra thông báo lỗi bởi vì điều này là không hợp lệ. Tuy nhiên, nếu 1 phương thức được khai báo là **virtual** tại lớp cơ sở nhưng trong lớp kế thừa, bạn “quên” không **override** lại thì khi biên dịch, C# chỉ đưa ra cảnh báo để nhắc nhở và chương trình vẫn được biên dịch thành công.
- Khi khai báo phương thức của 1 lớp, các từ khóa : **new**, **static**, **virtual** không được phép sử dụng chung với **override**.

■ Ví dụ Xây dựng lớp hình trụ kế thừa từ lớp hình tròn

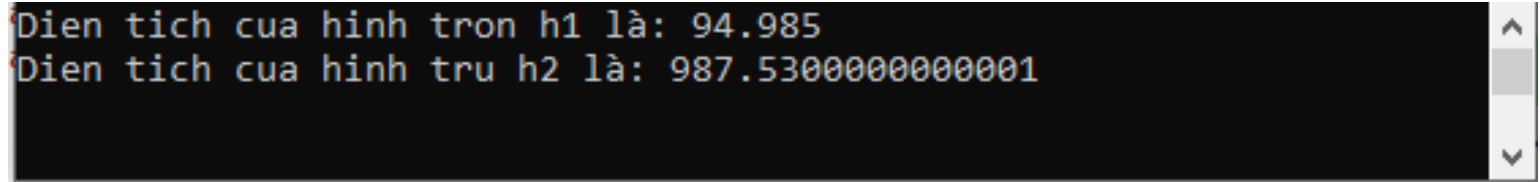
```
class hinhtron
{
    protected const double pi = 3.14;
    protected double r;
    //Phương thức khởi tạo có tham số lớp hình tròn
    public hinhtron(double bankinh)
    {
        r = bankinh;
    }
    //Định nghĩa một phương thức ảo của lớp hình tròn
    public virtual double dientich()
    {
        return (pi*r*r);
    }
}
```

```
class hinhtru:hinhtron //Lớp hình trụ kế thừa từ lớp hình tròn
{
    public double h;
    //Phương thức khởi tạo có tham số lớp hình trụ
    //kế thừa từ phương thức khởi tạo có tham số của lớp hình tròn
    public hinhtru(double bankinh, double chieucao):base(bankinh)
    {
        h = chieucao;
    }
    //Định nghĩa lại phương thức tính diện tích của lớp hình trụ
    //kế thừa từ lớp hình tròn
    public override double dientich()
    {
        return (2*pi*r*(h+r));
    }
}
```

■ Ví dụ Xây dựng lớp hình trụ kế thừa từ lớp hình tròn

class Program

```
{  
    static void Main(string[] args)  
    {  
        hìnhtron h1 = new hìnhtron(5.5); //tạo đối tượng h1 thuộc lớp hình tròn, có bán kính là 5.5  
        hìnhtru h2 = new hìnhtru(8.5, 10); //tạo đối tượng h2 thuộc lớp hình trụ, có bán kính là 8.5 và chiều cao 10  
        Console.WriteLine("Diện tích của hình tròn h1 là: {0}", h1.dientich());  
        Console.WriteLine("Diện tích của hình trụ h2 là: {0}", h2.dientich());  
        Console.ReadLine();  
    }  
}
```



```
Diện tích của hình tròn h1 là: 94.985  
Diện tích của hình trụ h2 là: 987.53000000000001
```

BÀI TẬP MỤC 2.4

Bài 1: Ứng dụng Console để thực hiện các công việc sau:

- Xây dựng lớp sinh viên:

+ thuộc tính: họ tên, năm sinh, quê quán, mã sinh viên.

+ phương thức: khởi tạo, nhập thông tin.

- Xây dựng lớp sinh viên đại học kế thừa lớp sinh viên:

Bổ sung:

+ thuộc tính: điểm năm 1, điểm năm 2, điểm năm 3, điểm năm 4, điểm năm 5.

+ phương thức: tính điểm trung bình (điểm trung bình = $(\text{điểm năm 1} + \text{điểm năm 2} + \text{điểm năm 3} + \text{điểm năm 4} + \text{điểm năm 5}) / 5$), in thông tin gồm cả điểm trung bình.

Yêu cầu:

a. Thực hiện nhập thông tin, in thông tin (tất cả các thông tin) của m ($m > 3$) đối tượng thuộc lớp sinh viên đại học.

b. Hiển thị danh sách các sinh viên có điểm trung bình dưới 5.

BÀI TẬP MỤC 2.4

Bài 2: Ứng dụng Console để thực hiện các công việc sau:

- Xây dựng lớp Công nhân viên chức:

+ thuộc tính: họ tên, lương cơ bản, hệ số lương.

+ phương thức: khởi tạo, nhập thông tin

- Xây dựng lớp Cán bộ phòng ban kế thừa lớp Công nhân viên chức:

Bổ sung:

+ thuộc tính: đơn vị công tác.

+ phương thức: tính tổng lương ($\text{tổng lương} = \text{lương cơ bản} * \text{hệ số lương}$), in thông tin gồm cả tổng lương.

- Xây dựng lớp Giảng viên kế thừa lớp Công nhân viên chức:

Bổ sung:

+ thuộc tính: bộ môn quản lý, số năm công tác.

+ phương thức: tính tổng thu nhập ($\text{tổng thu nhập} = \text{lương cơ bản} * (\text{hệ số lương} + \text{số năm công tác} / 50)$), in thông tin gồm cả tổng thu nhập.

BÀI TẬP MỤC 2.4

Bài 2 (tiếp): Ứng dụng Console để thực hiện các công việc sau:

Yêu cầu:

- Thực hiện nhập thông tin, in thông tin của n ($n > 2$) đối tượng thuộc lớp Cán bộ phòng ban.
- In ra màn hình danh sách Cán bộ phòng ban có đơn vị công tác là Phòng Đào tạo với đầy đủ các thông tin trên.
- Thực hiện nhập thông tin, in thông tin của m ($m > 2$) đối tượng thuộc lớp Giảng viên.
- In ra màn hình danh sách Giảng viên có tổng thu nhập dưới 10 triệu với đầy đủ các thông tin trên.