

# BÀI GIẢNG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## CHƯƠNG 5 KẾ THỪA

---

TRẦN THỊ THU THẢO

BỘ MÔN TIN HỌC QUẢN LÝ, KHOA THỐNG KÊ – TIN HỌC

TRƯỜNG ĐẠI HỌC KINH TẾ, ĐẠI HỌC ĐÀ NẴNG

THAOTRAN@DUE.EDU.VN

# NỘI DUNG

---

1. Khái niệm Kế thừa

2. Phạm vi Kế thừa

3. Gọi phương thức khởi tạo của lớp cơ sở

4. Định nghĩa phiên bản mới của lớp dẫn xuất

5. Tham chiếu thuộc lớp cơ sở

6. Thực thi trên C#



# KHÁI NIỆM KẾ THỪA

---

## □ KHÁI NIỆM

**Tính kế thừa** trong lập trình là cách 1 lớp có thể thừa hưởng lại những thuộc tính, phương thức từ 1 lớp khác và sử dụng chúng như là của bản thân mình. Hay nói cách khác, kế thừa là cơ chế cho phép định nghĩa một lớp mới (còn gọi là lớp dẫn xuất – derived class, dựa trên một lớp đã có sẵn (còn gọi là lớp cơ sở - base class). Lớp dẫn xuất có hầu hết các thành phần giống như lớp cơ sở (bao gồm tất cả các phương thức và biến thành viên của lớp cơ sở, trừ phương thức private, phương thức khởi tạo, phương thức hủy và phương thức tĩnh)

# KHÁI NIỆM KẾ THỪA

---

## □ KHÁI NIỆM

Cú pháp định nghĩa lớp dẫn xuất:

**class** <tên lớp con> : <tên lớp cơ sở>

```
{  
    // Thân lớp dẫn xuất  
}
```


**Trong đó:**

- **class** là từ khoá để khai báo lớp.
- <tên lớp con> là tên do người dùng đặt và tuân theo các quy tắc đặt tên <tên lớp cơ sở> là tên lớp mà ta muốn kế thừa các đặc tính của nó.

# KHÁI NIỆM KẾ THỪA

**Ví dụ 1: Xây dựng lớp Point2D (tọa độ trong không gian 2 chiều), từ đó mở rộng cho lớp Point3D.**

```
using System;
namespace C5Vd1
{
    class Point2D
    {
        public int x,y;
        public void Xuat2D()
        {
            Console.WriteLine("{0},{1}",x,y);
        }
    }
}
```



# KHÁI NIỆM KẾ THỪA

## Ví dụ 1:

```
class Point3D: Point2D
```

```
{
```

```
    public int z;
```

```
    public void Xuat3D()
```

```
{
```

```
        Console.WriteLine("{0},{1},{2}",x,y,z);
```

```
}
```

```
}
```

Lớp dẫn xuất Point3D kế thừa từ lớp Point2D

Lớp dẫn xuất Point3D, không cần khai báo các biến x, y nhưng trong phương thức Xuat3D(), vẫn truy cập x, y được. Thậm chí, trong hàm Main(), có thể sử dụng đối tượng p3 để gọi phương thức Xuat2D() của lớp cơ sở. Điều này chứng tỏ Point3D được kế thừa các biến x,y từ Point2D

# KHÁI NIỆM KẾ THỪA

## Ví dụ 1:

```
class PointApp
{
    static void Main(string[] args)
    {
        Point2D p2 = new Point2D();
        p2.x = 1;
        p2.y = 2;
        p2.Xuat2D();
        Point3D p3 = new Point3D();
        p3.x = 4;
        p3.y = 5;
        p3.z = 6;
        p3.Xuat3D();
        p3.Xuat2D();
    }
}
```

1,2  
4,5,6  
4,5

# PHẠM VI KẾ THỪA

---

Trong C#, không hỗ trợ đa kế thừa (1 lớp kế thừa từ nhiều lớp) nhưng lại hỗ trợ thực thi nhiều interface. Các thành phần của lớp cơ sở có được kế thừa xuống lớp dẫn xuất hay không là do phạm vi truy cập của thành phần đó là gì.

- Thành phần có phạm vi là **private** thì không được kế thừa.
- Thành phần có phạm vi là **protected**, **public** thì được phép kế thừa.

Phương thức khởi tạo và phương thức huỷ bỏ không được kế thừa.



# PHẠM VI KẾ THỪA

**Ví dụ 2:** Nếu ta định nghĩa lớp *ClassA* và *ClassB* kế thừa từ *ClassA* như sau thì câu lệnh  $x = x - 1$  sẽ bị báo lỗi

```
namespace C5Vd2
```

```
{  
    class ClassA  
    {  
        int x = 5;  
        public void XuatX()  
        {  
            Console.WriteLine("{0}",x);  
        }  
    }  
    class ClassB: ClassA  
    {  
        public void GiamX()  
        {  
            x = x - 1  
        }  
    }  
}
```

Muốn sửa lỗi phải khai báo lại thành

**public int x = 5** hoặc

**protected int x = 5**

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

---

❑ **Phương thức khởi tạo** mặc định của lớp cơ sở luôn luôn được gọi mỗi khi có 1 đối tượng thuộc lớp dẫn xuất khởi tạo. Và được gọi trước phương thức khởi tạo của lớp dẫn xuất. Nếu như lớp cơ sở có phương thức khởi tạo có tham số thì đòi hỏi lớp dẫn xuất phải có phương thức khởi tạo tương ứng và thực hiện gọi phương thức khởi tạo của lớp cơ sở thông qua từ khoá **base**.

❑ Khi đối tượng của lớp con bị huỷ thì **phương thức huỷ bỏ** của nó sẽ được gọi trước sau đó mới gọi phương thức huỷ bỏ của lớp cơ sở để huỷ những gì lớp con không huỷ được.

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

---

## □ CÚ PHÁP

```
public <tên lớp>(<danh sách tham số của lớp con>) : base(<danh  
sách tham số>)  
{  
    // Khởi tạo giá trị cho các thành phần của lớp dẫn xuất  
}
```

Trong đó:

- <tên lớp> là tên lớp con (lớp dẫn xuất).
- <danh sách tham số của lớp con> là danh sách tham số của constructor của lớp con.
- base là từ khoá để gọi đến constructor của lớp cơ sở.
- <danh sách tham số> là danh sách tham số tương ứng với constructor của lớp cha.

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

---

## □ CÚ PHÁP

```
public <tên lớp>(<danh sách tham số của lớp con>) : base(<danh  
sách tham số>)  
{  
    // Khởi tạo giá trị cho các thành phần của lớp dẫn xuất  
}
```

Trong đó:

- <tên lớp> là tên lớp con (lớp dẫn xuất).
- <danh sách tham số của lớp con> là danh sách tham số của constructor của lớp con.
- base là từ khoá để gọi đến constructor của lớp cơ sở.
- <danh sách tham số> là danh sách tham số tương ứng với constructor của lớp cha.

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

**Ví dụ 3: Xây dựng lớp Point2D, từ đó mở rộng cho lớp Point3D dùng phương thức tạo lập của lớp cơ sở**

```
using System;  
namespace C5Vd3
```

```
{  
    class Point2D  
    {  
        public int x,y;  
        public Point2D(int a, int b)  
        {  
            x = a; y = b;  
        }  
        public void Xuat2D()  
        {  
            Console.WriteLine("{0},{1}",x,y);  
        }  
    }  
}
```

Phương thức tạo lập của lớp Point2d có tham số

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

Ví dụ 3: Xây dựng lớp Point2D, từ đó mở rộng cho lớp Point3D dùng phương thức tạo lập của lớp cơ sở

```
class Point3D: Point2D
{
    public int z;
    public Point3D(int a, int b, int c):base(a,b)
    {
        z = c;
    }
    public void Xuat3D()
    {
        Console.WriteLine("{0},{1},{2}",x,y,z);
    }
}
```

Lớp Point3D dẫn xuất từ lớp Point2D, phương thức tạo lập cũng cần có tham số

Hai tham số đầu tiên dùng để khởi gán cho các biến x, y kế thừa từ lớp Point2D, tham số còn lại dùng để khởi gán cho biến thành viên z của lớp Point3D

# GỌI PHƯƠNG THỨC KHỞI TẠO CỦA LỚP CƠ SỞ

Ví dụ 3: Xây dựng lớp Point2D, từ đó mở rộng cho lớp Point3D dùng phương thức tạo lập của lớp cơ sở

```
class PointApp
{
    static void Main(string[] args)
    {
        Point2D p2 = new Point2D(1,2);
        Console.WriteLine("Toa do cua diem 2D : ");
        p2.Xuat2D();
        Point3D p3 = new Point3D(4,5,6);
        Console.WriteLine("Toa do cua diem 3D : ");
        p3.Xuat3D();
    }
}
```

```
Toa do cua diem 2D :1,2
Toa do cua diem 3D :4,5,6
```

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

---

## **Nhận xét chung:**

Khi cần định nghĩa hai lớp mà chúng có chung một vài đặc trưng, chức năng thì những thành phần đó nên được đặt vào một lớp cơ sở. Sau đó hai lớp này sẽ kế thừa từ lớp cơ sở đó và bổ sung thêm các thành phần của riêng chúng. Ngoài ra, lớp dẫn xuất còn có quyền định nghĩa lại các phương thức đã kế thừa từ lớp cơ sở nhưng không còn phù hợp với nó nữa.



# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

---

Lớp dẫn xuất kế thừa hầu hết các thành viên của lớp cơ sở vì vậy trong bất kỳ phương thức nào của lớp dẫn xuất ta có thể truy cập trực tiếp đến các thành viên này (mà không cần thông qua một đối tượng thuộc lớp cơ sở). Tuy nhiên, nếu lớp dẫn xuất cũng có một thành phần **X** (biến hoặc phương thức) nào đó trùng tên với thành viên thuộc lớp cơ sở thì trình biên dịch sẽ có cảnh báo dạng như sau:

```
"keyword new is required on 'LớpDẫnXuất.X' because  
it hides inherited member on 'LớpCơSở.X' "
```

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

---

Trong lớp dẫn xuất, khi khai báo một thành phần trùng tên lớp thành phần trong lớp cơ sở thì trình biên dịch hiểu rằng người dùng muốn che dấu các thành viên của lớp cơ sở và yêu cầu người dùng đặt từ khóa **new** ngay câu lệnh khai báo thành phần đó. Nếu phương thức của lớp dẫn xuất muốn truy cập đến thành phần **X** của lớp cơ sở thì phải sử dụng từ khóa **base** theo cú pháp: **base.X**

**Cú pháp:**

```
public new void <Tên phương thức>
{
    ....
}
```

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

## Ví dụ 4: Viết chương trình quản lý thông tin Xe

```
using System;
namespace C5Vd4
{
    class XeHoi
    {
        protected int TocDo;
        protected string BienSo;
        protected string HangSX;
        public XeHoi(int td, string BS, string HSX)
        {
            TocDo = td; BienSo = BS; HangSX = HSX;
        }
        public void Xuat()
        {
            Console.WriteLine("Xe: {0}, Bien so: {1}, Tocdo: {2}
kmh", HangSX, BienSo, TocDo);
        }
    }
}
```

Lớp XeHoi có phương thức Xuat() xuất ra các thông tin như Biển số, Tốc độ, Hãng sản xuất

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

## Ví dụ 4: Viết chương trình quản lý thông tin Xe

```
class XeCar: XeHoi
{
    int SoHanhKhach;
    public XeCar(int td, string BS, string HSX, int SHK):
base(td,BS,HSX)
    {
        SoHanhKhach = SHK;
    }
    public new void Xuat()
    {
        base.Xuat();
        Console.WriteLine(",{0} cho ngoi", SoHanhKhach);
    }
}
```

Phương thức Xuat() của lớp XeHoi không còn phù hợp với lớp XeCar. Cần định nghĩa một phiên bản mới của phương thức Xuat()

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

## Ví dụ 4: Viết chương trình quản lý thông tin Xe

```
class XeTai: XeHoi
{
    int TrongTai;
    public XeTai(int td, string BS, string HSX, int TT):
base(td,BS,HSX)
    {
        TrongTai = TT;
    }
    public new void Xuat()
    {
        base.Xuat();
        Console.WriteLine(", trong tai {0} tan", TrongTai);
    }
}
```

Sử dụng từ khóa base để đại diện cho lớp cơ sở và gọi đến các thành phần của lớp cơ sở

# ĐỊNH NGHĨA PHIÊN BẢN MỚI TRONG LỚP DẪN XUẤT

## Ví dụ 4: Viết chương trình quản lý thông tin Xe

```
public class Program
{
    static void Main(string[] args)
    {
        XeCar c = new XeCar(150, "43A-45235", "Toyota", 24);
        c.Xuat();
        XeTai t = new XeTai(150, "43A-98235", "Benz", 12);
        t.Xuat();
    }
}
```

Xe: Toyota, Bien so: 43A-45235, Tocdo: 150 kmh, 24 cho ngoi  
Xe: Benz, Bien so: 43A-98235, Tocdo: 150 kmh, trong tai 12 tan

# THAM CHIẾU THUỘC LỚP CƠ SỞ

---

Một tham chiếu thuộc lớp cơ sở có thể trỏ đến một đối tượng thuộc lớp dẫn xuất nhưng nó chỉ được phép truy cập đến các thành phần được khai báo trong lớp cơ sở.

Ở Ví dụ 4, với các lớp XeHoi, XeCar như trên, ta có thể định nghĩa hàm Main() như sau:

# THAM CHIẾU THUỘC LỚP CƠ SỞ

```
static void Main(string[] args)
{
    XeCar c = new XeCar(150, "49A-4444", "Toyota", 24);
    c.Xuat();
    Console.WriteLine();
    Console.WriteLine("Tham chieu cua lop co so XeHoi  
co the tro den doi tuong thuoc lop dan xuat XeCar");
    Console.WriteLine("Nhưng chỉ có thể gọi hàm xuất  
tuong ung voi XeHoi");
    XeHoi h = c;
    h.Xuat();
}
```

Xe: Toyota, Bien so: 49A-4444, Tocdo: 150 kmh, 24 cho ngoi

Tham chieu cua lop co so XeHoi co the tro den doi tuong thuoc lop dan xuat XeCar  
Nhưng chỉ có thể gọi hàm xuất tuong ung voi XeHoi  
Xe: Toyota, Bien so: 49A-4444, Tocdo: 150 kmh



# BÀI TẬP ÔN TẬP CHƯƠNG

---

**Bài 1.** Xây dựng lớp hình tròn với các thuộc tính (properties): **bán kính, đường kính, diện tích.**

**a.** Xây dựng lớp hình cầu kế thừa từ lớp hình tròn. Lớp này che dấu đi các thuộc tính: diện tích (dùng từ khóa new) đồng thời bổ sung thêm thuộc tính: **thể tích.**

- Diện tích hình cầu tính bán kính R được tính theo công thức

$$4 * PI * R^2$$

- Thể tích hình cầu tính bán kính R được tính theo công thức

$$4/3 * PI * R^3$$

**b.** Tương tự, xây dựng lớp hình trụ tròn kế thừa từ lớp hình tròn với các thuộc tính: chu vi mặt đáy, diện tích mặt đáy, diện tích xung quanh, diện tích toàn phần, thể tích.

# BÀI TẬP ÔN TẬP CHƯƠNG

---

**Bài 2a.** Xây dựng Lớp **People** gồm:

- Thuộc tính: **ID**, **Hoten**, **Tuoi**, **Diachi** để lưu lần lượt các giá trị mã số, tên, tuổi và địa chỉ
- Phương thức trong lớp **People** gồm: hàm **Nhap()** và **Xuat()**, **hàm khởi tạo (có hoặc không có tham số)** và **hàm hủy** (nếu có)

❖ Tạo Lớp **Students** Kế thừa từ lớp **People**

Lớp **Students** sẽ có thêm:

- Thuộc tính **Term** để lưu tên học phần, **TP1**, **TP2**, **TP3** lần lượt là điểm thành phần 1, 2, 3
  - Phương thức **GPA()** để tính điểm trung bình môn học và xếp loại theo thang điểm tín chỉ của môn học đó.
- ❖ **Xuất ra màn hình tất cả thông tin của sinh viên**

# BÀI TẬP ÔN TẬP CHƯƠNG

---

## ❖ Bài 2b. Tạo Lớp Lecture Kế thừa từ lớp People

Lớp Lecture sẽ có thêm:

- Thuộc tính **Kinhnghiem** để lưu số năm kinh nghiệm, **Hocvi**, **Chucvu** để lưu học vị (ThS/TS/...), chức vụ (Trưởng BM, Trưởng Khoa, ...) của Giảng viên
  - Phương thức **Sapxep()** để sắp xếp giảng viên theo số năm kinh nghiệm giảng dạy
- ❖ Xuất ra màn hình tất cả thông tin của các Giảng viên được sắp xếp theo số năm kinh nghiệm từ nhiều đến ít.