



CHƯƠNG 3: HƯỚNG ĐỐI TƯỢNG TRONG C# (tt)

Thời gian: 120 phút

Giảng Viên: Nguyễn Tấn Thuận

Email: nguyentanthuan@dtu.edu.vn

Điện thoại: 0905626276



PHẦN 3

LỚP TRỪU TƯỢNG

INTERFACE



Nội Dung

- ❖ Lớp trừu tượng (Abstract Class)
- ❖ Lớp vô sinh (Sealed Class)
- ❖ Struct
- ❖ Giao diện (Interface)





LỚP TRỪ TƯỢNG

- ❖ Cũng là một lớp mà không muốn người khác tạo ra đối tượng từ lớp này
- ❖ Một lớp trừ tượng là một lớp được khai báo với từ khóa **abstract**
- ❖ Chứa các phương thức trừ tượng.
- ❖ Tất cả các lớp trừ tượng không thể khởi tạo đối tượng nhưng có thể được kế thừa.



LỚP TRỪU TƯỢNG

❖ Video tham khảo:

<https://www.youtube.com/watch?v=FwDtakRIILw>



LỚP TRỪU TƯỢNG

➤ Cú pháp:

```
abstract class Tên lớp{  
  
    ...  
  
}
```

➤ Ví dụ minh họa

```
abstract class Hình{  
  
    ...  
  
}
```

PHƯƠNG THỨC TRỪ TƯỢNG

- ❖ Muốn người khác phải ghi đề phương thức khi kế thừa từ lớp này
- ❖ Một phương thức trừ tượng là một phương thức được khai báo với từ khóa **abstract**

<phạm vi truy cập> abstract <kiểu trả về> tên phương thức();

❖ Lưu ý:

- Phương thức trừ tượng không có phần thân
- Các lớp con phải hiện thực các phương thức trừ tượng đó bằng cách viết chồng phương thức



LỚP TRỪ TƯỢNG

- Lớp chứa một hay nhiều phương thức trừu tượng
 - Là lớp trừu tượng.
 - Cần khai báo từ khóa abstract
- Nhìn chung có dạng như sau:

```
public abstract class Hình{  
    // Khai báo các biến  
    // Khai báo các phương thức không phải trừu tượng  
    // Khai báo các phương thức trừu tượng  
    abstract void TinhDienTich(); // Phương thức trừu tượng  
}
```




LỚP TRỪU TƯỢNG

Ví dụ: Tạo lớp Hình

```
class Hình
{
    public float TinhDienTich()
    {
        //Than cua phuong thuc
    }
    public float TinhChuVi()
    {
        //Than cua phuong thuc
    }
}
```

Vì chúng ta không biết là hình gì nên không thể triển khai phương thức do đó chúng ta sẽ tạo ra lớp trừu tượng Hình

```
abstract class Hình
{
    public abstract float TinhDienTich();
    public abstract float TinhChuVi();
}
```



LỚP TRỪU TƯỢNG

- Tạo ra lớp Hình chữ nhật thừa kế lớp trừu tượng hình. Lúc này ta đã có thể triển khai phương thức tính diện tích và tính chu vi

```
class HINHCHUNHAT :Hình
{
    double chieudai, chieurong;
    public override double TinhDienTich()
    {
        return chieudai * chieurong;
    }
    public override double TinhChuVi()
    {
        return chieudai + chieurong;
    }
}
```



LỚP TRỪU TƯỢNG

- Tương tự tạo ra lớp Hình tròn thừa kế lớp trừu tượng hình. Lúc này ta đã có thể triển khai phương thức tính diện tích và tính chu vi

```
class HINHTRON : Hình
{
    double bankinh;
    public override double TinhDienTich()
    {
        return bankinh * bankinh * 3.14;
    }
    public override double TinhChuVi()
    {
        return bankinh * 2 * 3.14;
    }
}
```



LỚP TRỪU TƯỢNG

Tuy nhiên: ta phải khai báo tất cả các phương thức trừu tượng

```
class HINHTRON : Hình
{
    double bankinh;
    public override double TinhDienTich()
    {
        return bankinh * bankinh * 3.14;
    }
}
```

'LopTruuTuongHinh.HINHTRON' does not implement inherited abstract member 'LopTruuTuongHinh.Hinh.TinhChuVi()'



LỚP TRỪU TƯỢNG

Một số lưu ý:

```
abstract class Hình
{
    int soA;
    string tenhinh;
    public string TENHINH
    {
        get { return tenhinh; }
        set { tenhinh = value; }
    }
    public abstract double TinhDienTich();
    public abstract double TinhChuVi();
    public void Nhap()
    {
        Console.WriteLine("Đây là phương thức bình thường");
    }
    public Hình() { } // Hàm khởi tạo không đối
    public Hình(int a, int b){} // Hàm khởi tạo có đối
}
```

- chúng ta muốn tạo một lớp cha mà vừa có các phương thức thông thường và vừa có các phương thức trừu tượng



LỚP VÔ SINH Sealed class

- ❖ Lớp không thể có lớp dẫn xuất từ nó (không có lớp con) gọi là lớp “vô sinh”, hay nói cách khác không thể kế thừa được từ một lớp “vô sinh”.
- ❖ Lớp “vô sinh” dùng để hạn chế, ngăn ngừa các lớp khác dẫn xuất từ nó.
- ❖ Để khai báo một lớp là lớp “vô sinh”, chúng ta dùng từ khóa Sealed class.
- ❖ Tất cả các phương thức của lớp vô sinh đều vô sinh, nhưng các thuộc tính của lớp vô sinh thì có thể không vô sinh.



LỚP VÔ SINH Sealed class

- ❖ Lớp không thể có lớp dẫn xuất từ nó (không có lớp con) gọi là lớp “vô sinh”, hay nói cách khác không thể kế thừa được từ một lớp “vô sinh”.
- ❖ Lớp “vô sinh” dùng để hạn chế, ngăn ngừa các lớp khác dẫn xuất từ nó.
- ❖ Để khai báo một lớp là lớp “vô sinh”, chúng ta dùng từ khóa Sealed class.
- ❖ Tất cả các phương thức của lớp vô sinh đều vô sinh, nhưng các thuộc tính của lớp vô sinh thì có thể không vô sinh.




LỚP VÔ SINH Sealed class

sealed class Student

```
{  
    string name;  
    int sid;  
    public Student(string name, int sid)  
    {  
        this.name = name;  
        this.sid = sid;  
    }  
}
```

class HutechStudent : Student


'HaGiang.HutechStudent': cannot derive from sealed type 'HaGiang.Student'



Struct

Class	Struct
Class là kiểu dữ liệu tham chiếu được lưu trong heap	Struct là kiểu dữ liệu tham trị được lưu trong Stack.
class có hỗ trợ kế thừa	Struct không hỗ trợ kế thừa
class phù hợp với các cấu trúc dữ liệu phức tạp	Khi struct được khởi tạo với từ khóa new, khi đó một constructor (hàm khởi tạo) được gọi để khởi tạo các trường trong cấu trúc.
	Khi struct được khởi tạo không dùng từ khóa new thì không có constructor(hàm khởi tạo) được gọi, do vậy người dùng cần khởi tạo tất cả các trường trước khi sử dụng



Struct

```
struct Point
```

```
{
```

```
    public int x;
```

```
    public int y;
```

```
    public Point(int x, int y) // constructor
```

```
    {
```

```
        this.x = x;
```

```
        this.y = y;
```

```
    }
```

```
    public static Point operator +(Point p1, Point p2) // operator "+"
```

```
    {
```

```
        Point p;
```

```
        p.x = p1.x + p2.x;
```

```
        p.y = p1.y + p2.y;
```

```
        return p;
```



Struct

```
class Program
```

```
{
```

```
    public static void Main()
```

```
    {
```

```
        Point p1 = new Point(5, 5); // using new keyword
```

```
        Point p2; // another way
```

```
        p2.x = 8; // assign a value for x
```

```
        p2.y = 8; // assign a value for y
```

```
        Point p3 = p1 + p2; // add
```

```
        Console.WriteLine("P1.x = {0}, P1.y = {1}", p1.x, p1.y);
```

```
        Console.WriteLine("P2.x = {0}, P2.y = {1}", p2.x, p2.y);
```

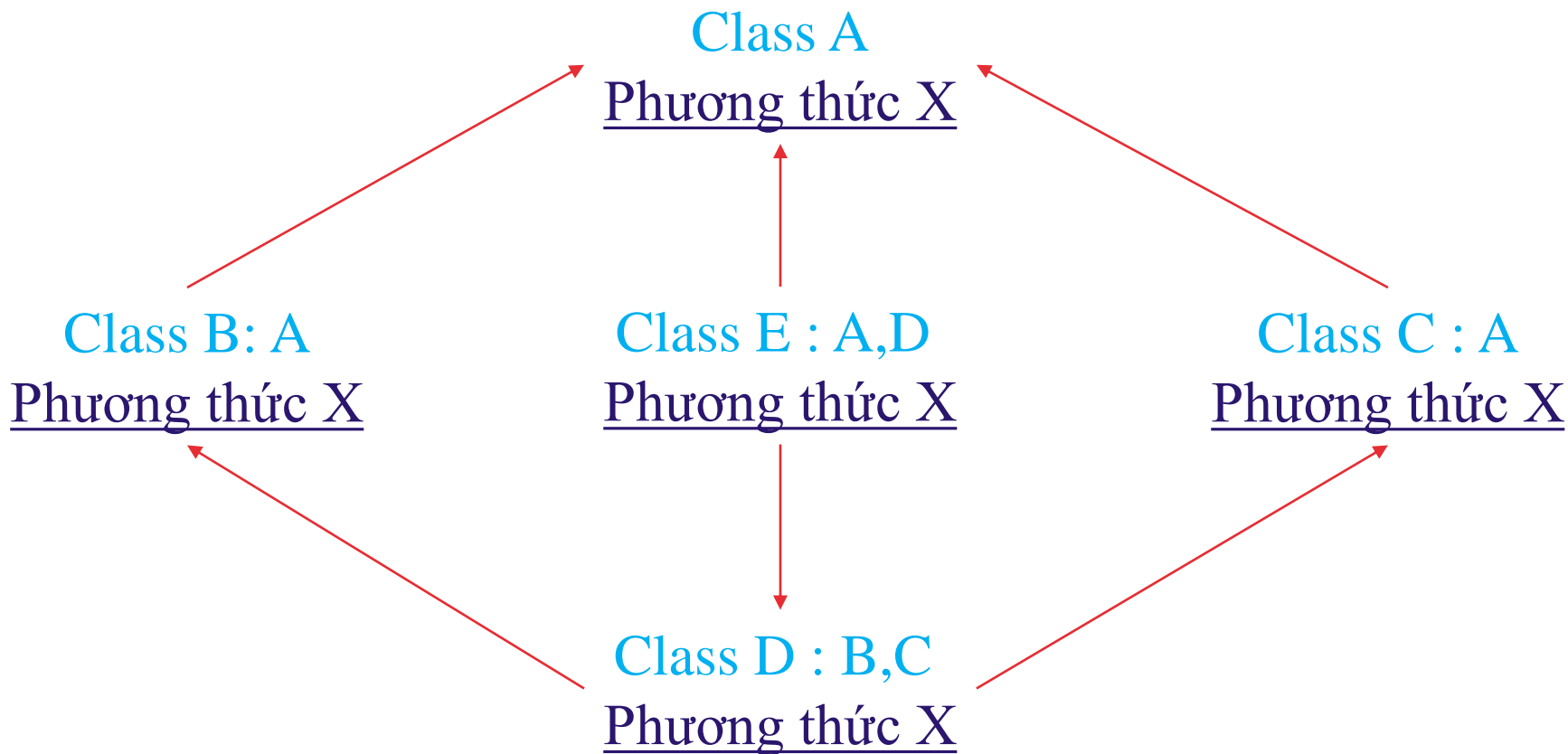
```
        Console.WriteLine("P3.x = {0}, P3.y = {1}", p3.x, p3.y);
```

```
        Console.ReadLine();
```

```
    }
```



GAO DIỆN





GIAO DIỆN

- ❖ **Giao diện được định nghĩa như là “Ký ước”**
 - Bất kỳ lớp nào triển khai giao diện phải xác định từng thành viên của giao diện đó
- ❖ **Giao diện là đối tượng loại tham chiếu không có triển khai**
- ❖ **Giao diện chứa**
 - Phương thức, thuộc tính, chỉ mục và sự kiện
- ❖ **Sử dụng khi các lớp khác nhau cần chia sẻ các phương thức phổ biến**



GIAO DIỆN

- ❖ Không thực thi bất kỳ phương thức nào
- ❖ Chỉ chứa các phương thức, thuộc tính, chỉ mục và sự kiện
- ❖ Phạm vi truy cập phương thức luôn là **PUBLIC**
- ❖ Khi một lớp thừa kế Interface thì class đó phải chạy thực thi tất cả các thành phần trong Interface đó.
- ❖ Có thể kế thừa từ nhiều Interface khác nhau
- ❖ Các phương thức trong Interface mặc định là **abstract**



GIAO DIỆN

❖ Video tham khảo:

<https://www.youtube.com/watch?v=i6zbQZT4QII>



```
interface IDrawing
{
    int X
    {
        get;
        set;
    }
    int Y
    {
        get;
        set;
    }
    void Draw(); //method for drawing
}
```




GIAO DIỆN

```
class Shape : IDrawing
```

```
{
```

```
    private int x;
```

```
    private int y;
```

```
    public int X {
```

```
        get { return x; }
```

```
        set { x = value; }
```

```
    }
```

```
    public int Y {
```

```
        get { return y; }
```

```
        set { y = value; }
```

```
    }
```

```
    public void Draw() {
```

```
        Console.WriteLine("Shape draw");
```

```
    }
```



THẢO LUẬN

❖ Sinh viên thảo luận để trả lời các câu hỏi sau:

1. Phân biệt điểm giống và khác nhau của lớp trừu tượng và giao diện
2. Khi nào nên sử dụng giao diện, khi nào nên sử dụng lớp trừu tượng



Giống nhau

- Điều không thể tạo đối tượng bên trong được
- Điều khai báo các phương thức nhưng không sử dụng chúng
- Điều bao gồm các phương thức trừu tượng
- Điều được thực thi từ các class con
- Điều có thể kế thừa từ nhiều Interface



Khác nhau

AbstractClass	Interface
Có thể kế thừa một class và nhiều interface	Chỉ có thể kế thừa từ nhiều interface
Các phương thức của Abstract class được thực thi khi sử dụng từ khóa override	Không cần
Là lựa chọn thích hợp khi vừa khai báo các phương thức thông thường vừa khai báo các phương thức trừu tượng	Thích hợp cho việc khai báo duy nhất các phương thức trừu tượng
Có thể có hàm khởi tạo	Không có



Nên sử dụng khi

AbstractClass

Là lựa chọn thích hợp khi vừa khai báo các phương thức thông thường vừa khai báo các phương thức trừu tượng

Interface

Thích hợp cho việc khai báo duy nhất các phương thức trừu tượng



GIAO DIỆN Interface

- ❖ **Cung cấp một phương pháp so sánh hai đối tượng cho của một kiểu cụ thể.**
 - Ví dụ: Student, Person, Car, Employee...
- ❖ **Vấn đề & giải pháp:**
 - nếu bạn có một mảng các đối tượng Student, bạn gọi phương thức Sort trên mảng đó, Comparable cung cấp sự so sánh các đối tượng trong khi sắp xếp.
- ❖ **Khi bạn triển khai giao diện Comparable, bạn phải thực hiện phương thức CompareTo**



GIAO DIỆN IComparable

❖ Video tham khảo:

https://www.youtube.com/watch?v=onw_BtmqSx0



GIAO DIỆN IComparable

```
class Employee: IComparable
{
    //data member
    private string name;
    private float salary;

    // constructor
    public Employee(string name, float salary)
    {
        this.name = name;
        this.salary = salary;
    }
    // method
    public void Show()
    {
        Console.WriteLine("Name: {0} \t Salary: {1}", name, salary);
    }
}
```




GIAO DIỆN IComparable

```
static void Main(string[] args)
{
    Employee[] emps = new Employee[3];
    emps[0] = new Employee("Nam", 100);
    emps[1] = new Employee("Binh", 400);
    emps[2] = new Employee("Hung", 200);
    foreach (Employee e in emps)
        e.Show();
    // after sorting by name
    Array.Sort(emps);
    foreach (Employee e in emps)
        e.Show();
    Console.ReadLine();
}
```



GIAO DIỆN IComparable

- ❖ **Vai trò của Icomparer là cung cấp các cơ chế so sánh bổ sung**
 - Ví dụ: cung cấp thứ tự lớp của bạn trên một số trường, thứ tự tăng dần và giảm dần trên cùng một trường



THẢO LUẬN





BÀI THỰC HÀNH

Nhân viên biên chế

- Phụ cấp = lương căn bản / 10
- Thực lĩnh = hệ số lương * lương căn bản + phụ cấp

Class NVBC

Hệ số lương
Lương cơ bản

Class NhanVien

Họ tên
Năm sinh
Giới tính
Số CMND
Mã NV
Nhập(): void
Xuất(): void

Class DSNV

Dictionary NhanVien
Nhập(): void
In(): void
Tổng Quỹ Lương()
Tìm()
Xóa()

Class NVHD

Mức Lương

Nhân viên Hợp đồng

- Phụ cấp = Mức lương / 10
- Thực lĩnh = Mức lương + phụ cấp

Interface ILUONG

Phụ cấp (): double
Thực lĩnh(): double



Tài liệu tham khảo

- [1] Giáo trình lập trình Winform với C#.NET Lê Trung Hiếu, Nguyễn Thị Minh Thi
- [2] Giáo trình lập trình C#.net Phạm Hữu Khang
- [3] C# Language Reference, Anders Hejlsberg and Scott Wiltamuth, Microsoft Corp.
- [4] Professional C#, 2nd Edition, Wrox Press Ltd.
- [5] Web site www.Codeproject.com
- [6] Web site www.CodeGuru.com