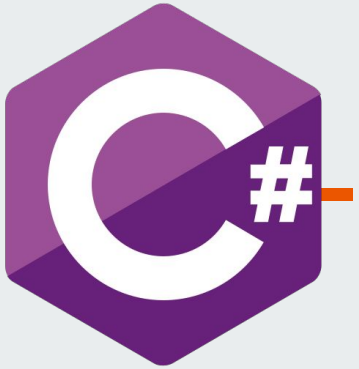


Bài 8

Lớp trừu tượng và giao diện

Nội dung

- Lớp trừu tượng
- Giao diện
- Thực thi giao diện
- Đa kế thừa giao diện
- Thực thi giao diện tường minh
- Kế thừa giao diện
- Sự khác nhau giữa lớp trừu tượng và giao diện

Lớp trừu tượng 1-3

- Lớp trừu tượng (abstract class) là một lớp cơ sở chưa hoàn thành, tức là bên trong lớp có một số phương thức trừu tượng chỉ khai báo mà chưa triển khai nội dung.
- Lớp trừu tượng không cho phép tạo thể hiện, nhưng nó cho phép kế thừa.
- Các lớp con được kế thừa phải thực thi tất cả các phương thức trừu tượng được khai báo trong lớp cơ sở, nếu không thì nó phải khai báo là abstract class.

Lớp trừu tượng 2-3

- **Cú pháp**

```
public abstract class <tên_lớp>
{
    //khai báo các thành viên của lớp
    //.....................
    //khai báo các phương thức trừu tượng
    <phạm_vi> abstract <kiểu_dữ_liệu> <tên_phương_thức>([tham_số]);
}
```

- **Lưu ý:** Khi thực thi các phương thức trừu tượng ở lớp cơ sở tại lớp con, bạn cần thêm từ khóa **override** vào tương tự việc ghi đè trong kế thừa.

Lớp trừu tượng 3-3

```
//khai báo lớp trừu tượng
public abstract class Person
{
    //phương thức không trừu tượng
    public void Speak()
    {
        Console.WriteLine("Noi bang ngon ngu Tieng Viet");
    }
    //phương thức trừu tượng
    public abstract void DoWork();
    public abstract void EnvironmentWork();
}
class Developer : Person
{
    //thực thi các phương thức trừu tượng ở lớp cơ sở
    public override void DoWork()
    {
        Console.WriteLine("Lam viec rat vat va");
    }

    public override void EnvironmentWork()
    {
        Console.WriteLine("Lam viec suot ngay ben may tinh nen rat hai cho doi mat");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Developer dev = new Developer();
        dev.Speak();
        dev.DoWork();
        dev.EnvironmentWork();
    }
}
```

Giao diện (Interface) 1-4

- **Mục đích:** Giao diện được giới thiệu để giải quyết vấn đề đa kế thừa trong C#, mặc định một lớp trong C# chỉ cho phép kế từ một lớp khác, nhưng nó có thể thực thi từ nhiều giao diện.
- **Giao diện là gì?:** Giao diện là ràng buộc, giao ước đảm bảo cho các lớp hay các cấu trúc sẽ thực thi những gì khai báo trong giao diện. Khi một lớp thực thi một giao diện, lớp này phải thực thi tất cả các thành viên của giao diện. Đây là một bắt buộc mà các lớp phải thực hiện.

Giao diện (Interface) 2-4

- Cú pháp:

```
[public] interface <tên_giao_diện>
{
    //khai báo các thành viên
}
```

```
//khai báo giao diện
public interface IStorable
{
    void Read();
    void Write(Object data);
}
```

Lưu ý

- Không được sử dụng bất kỳ phạm vi truy cập nào khi khai báo các thành viên của giao diện(mặc định là public)
- Không được sử dụng từ khóa abstract khi khai báo các thành viên giao diện (mặc định là abstract)

Thực thi giao diện

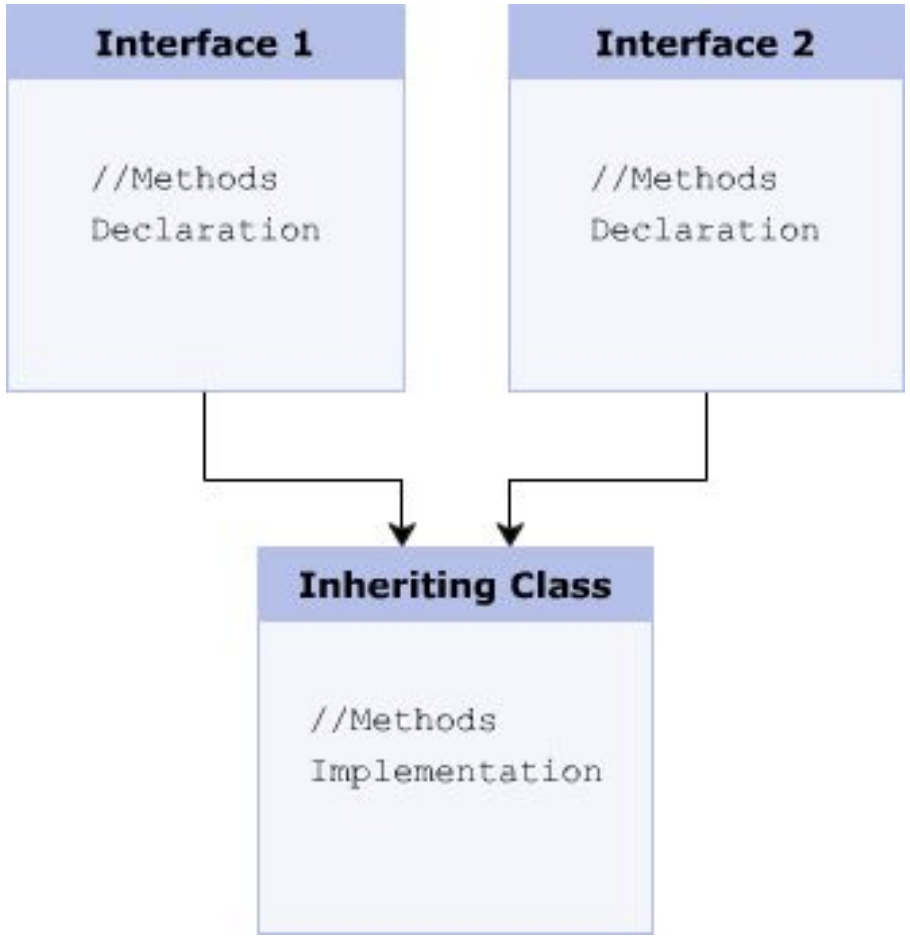
```
//khai báo giao diện
public interface IStorable
{
    void Read();
    void Write(Object data);
}
//khai báo lớp thực thi từ giao diện
public class Document : IStorable
{

    #region "thực thi các phương thức từ giao diện IStorable"
    public void Read()
    {
        Console.WriteLine("Doc du lieu tu tai lieu");
    }

    public void Write(object data)
    {
        Console.WriteLine("Ghi du lieu " + data + " vao tai lieu");
    }
    #endregion
}
```

Giao diện và đa kế thừa 1-2

- Một lớp có thể thực thi từ nhiều giao diện, khi thực thi thì mỗi giao diện cách nhau bởi dấu phẩy.
- Một lớp thực thi từ nhiều giao diện thì phải thực thi tất cả các phương thức trừu tượng được khai báo trong các giao diện
- Từ khóa **override** không được sử dụng trong khi thực thi các phương thức trừu tượng của giao diện.



Giao diện và đa kế thừa 2-2

```
//Định nghĩa giao diện 1
interface IMachine
{
    void Start();
    void Stop();
}
//Định nghĩa giao diện 2
interface IComputer
{
    void CheckBios();
    void LoadOS();
}
```

```
//Định nghĩa lớp thực thi từ 2 giao diện
class HPComputer : IMachine, IComputer
{
    public void CheckBios()
    {
        Console.WriteLine("Checking Bios");
    }

    public void LoadOS()
    {
        Console.WriteLine("Loading Operating System");
    }

    void IMachine.Start()
    {
        Console.WriteLine("Start computer");
    }

    void IMachine.Stop()
    {
        Console.WriteLine("Stop computer");
    }
}
```

Thực thi giao diện tường minh 1-2

- Khi các giao diện khai báo các phương thức thành viên trùng tên nhau, lúc này lớp thực thi từ các giao diện đó phải thực thi tường minh các phương thức trùng tên nhau, trường hợp này gọi là thực thi giao diện tường minh

Thực thi giao diện tường minh 2-2

```
//ví dụ thi giao diện tường mình
interface ITerrestrialAminal
{
    void Eat();
}
interface IMarineAnimal
{
    void Eat();
}
```

```
class Crocodie : ITerrestrialAminal, IMarineAnimal
{
    //thực thi tường minh 2 phương thức của 2 giao diện
    void ITerrestrialAminal.Eat()
    {
        Console.WriteLine("Eat on Terrestrial");
    }
    void IMarineAnimal.Eat()
    {
        Console.WriteLine("Eat on Marine");
    }
    //public các phương thức ra ngoài khi cần sử dụng
    public void EatTerrestrial()
    {
        ITerrestrialAminal ta = this;
        ta.Eat();
    }
    public void EatMarine()
    {
        IMarineAnimal ma = this;
        ma.Eat();
    }
}
```

Toán tử is và as

Kế thừa giao diện

- Một giao diện có thể kế thừa từ một hoặc nhiều giao diện khác, nhưng không thể thực thi chúng, việc thực thi sẽ được giao cho các lớp.

```
//Định nghĩa giao diện Tồn kho
interface IInventoryItem
{
    //có thể khai báo cả thuộc tính trong giao diện
    string ProductName { get; set; }
    int QuantityInStock { get; set; }
}
//Định nghĩa giao diện cho thuê
interface IRentaltable: IInventoryItem
{
    void Rent();
    void ReturnRental();
}
//Định nghĩa giao diện mua hàng kế thừa từ 2 giao diện trên
interface IPurchasable : IRentaltable, IInventoryItem
{
    void Purchase();
}
```

```
//Tạo lớp thực thi từ giao diện mua hàng
public class VehicleModel : IPurchasable
{
    public decimal DealerFee { get; set; }
    //thực thi thuộc tính
    public string ProductName { get ; set; }
    public int QuantityInStock { get; set; }
    //thực thi tất cả các phương thức từ 2 giao diện mà nó kế thừa
    public void Purchase()
    {
        QuantityInStock -= 1;
        Console.WriteLine("This vechicle has been purchased");
    }
    public void Rent()
    {
        QuantityInStock -= 1;
        Console.WriteLine("This vehicle has been rented");
    }
    public void ReturnRental()
    {
        QuantityInStock += 1;
        Console.WriteLine("This vechicle has been returned");
    }
}
```

So sánh lớp trừu tượng và giao diện

Abstract Classes	Interfaces
Lớp trừu tượng có thể kế thừa từ một lớp hoặc nhiều giao diện	Giao diện có thể kế thừa từ nhiều giao diện nhưng không thể kế thừa từ lớp
Lớp trừu tượng có thể có phương thức triển khai nội dung	Giao diện chỉ chứa các phương thức hoàn toàn trừu tượng
Phương thức trong lớp trừu tượng được thực thi sử dụng từ khóa override	Phương thức trong giao diện không cần sử dụng từ khóa override
Lớp trừu tượng là tùy chọn tốt khi bạn cần thực thi các phương thức chung và khai báo phương thức trừu tượng	Giao diện là tùy chọn tốt khi bạn cần khai báo các phương thức trừu tượng
Lớp trừu tượng có thể khai báo constructor và destructor	Giao diện không thể khai báo constructors hoặc destructor

HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH