

## Bài 7

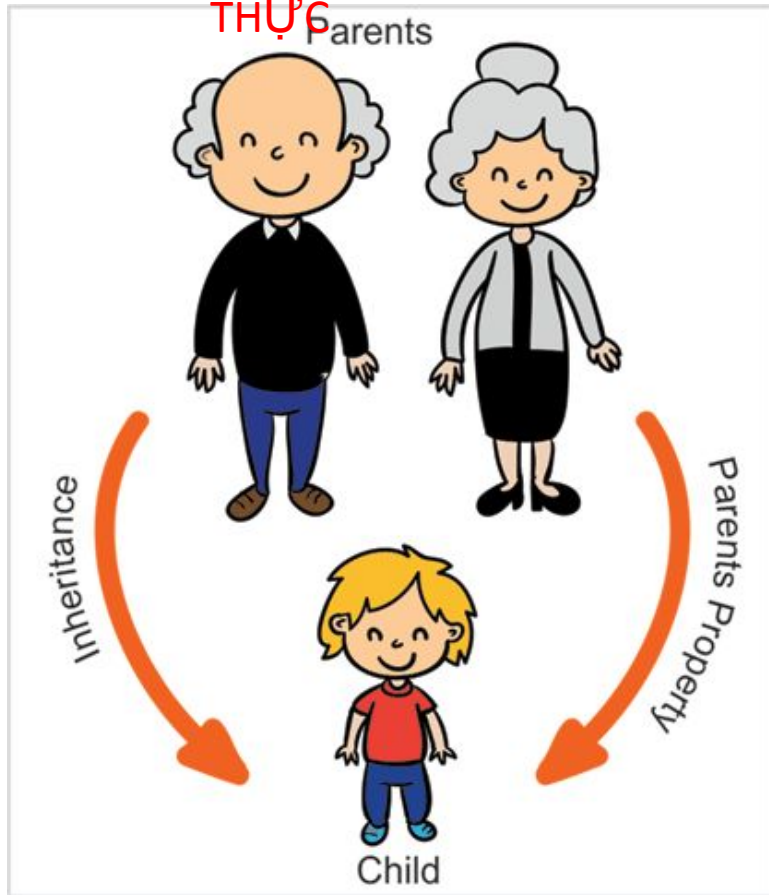
# Kế thừa và đa hình

# Mục tiêu

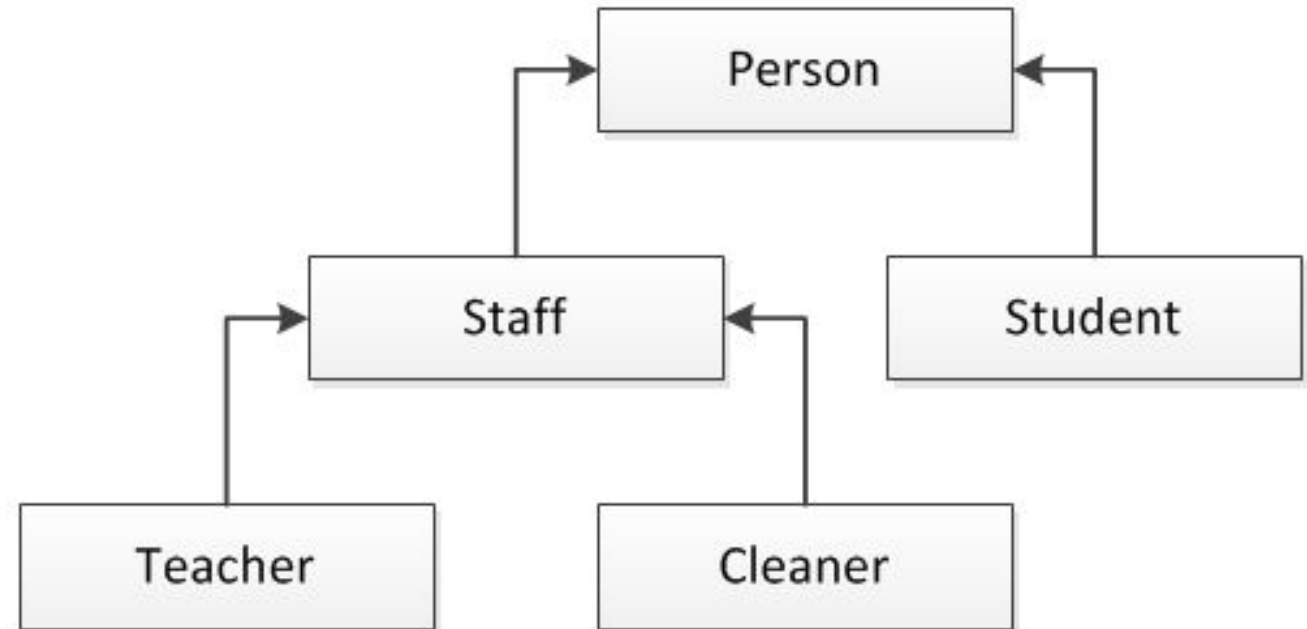
- Giải thích về kế thừa
- Tạo lớp cơ sở và lớp dẫn xuất
- Ghi đè phương thức
- Lớp cô lập
- Giải thích về đa hình

# Kế thừa 1-2

THỂ GIỚI  
THỰC



TRONG LẬP TRÌNH OOP



# Kế thừa 2-2

- **Kế thừa (Inheritance)** cho phép một lớp (class) có thể kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa.
- Lớp cho lớp khác kế thừa gọi là lớp cơ sở (Base class, Parent Class, Super class)
- Lớp kế thừa những thành viên từ lớp khác gọi là lớp con (Child class, Derived class).
- Derived class được kế thừa tất cả những thành phần của base class ngoại trừ những thành phần là private.

# Tạo lớp cơ sở và lớp dẫn xuất

```
//Tạo lớp cơ sở
class Animal
{
    public void Eat()
    {
        Console.WriteLine("Dong vat an mot vai thu");
    }
    public void DoSomething()
    {
        Console.WriteLine("Dong vat lam mot vai thu");
    }
}
//Tạo lớp con kế thừa từ Animal
class Cat : Animal
{
    public void Actions()
    {
        //Các phương thức kế thừa từ lớp cha
        Eat();
        DoSomething();
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Cat c = new Cat();
        c.Actions();
    }
}
```

# Bổ từ truy cập Protected

- Bổ từ “protected” sử dụng cho các thành viên của lớp cơ sở, nó quy định các thành viên đó chỉ được truy ở trong lớp đó và trong lớp dẫn xuất.

```
//Lớp cơ sở
class Animal
{
    //khai báo các thành viên protected
    protected string food;
    protected string activity;
}
//lớp con
class Cat : Animal
{
    public Cat(string food, string activity)
    {
        //sử dụng các thành viên protected ở lớp cha
        base.food = food;
        base.activity = activity;
    }
    //hiển thị thông tin các thành viên protected ở lớp cha
    public void Show()
    {
        Console.WriteLine("Meo an:" + food);
        Console.WriteLine("Meo :" + activity);
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Cat c = new Cat("Fish", "Run and Jump");
        c.Show();
    }
}
```

# Từ khóa “base”, từ khóa “new” 1-2

- Từ khóa “base” cho phép bạn truy cập tới các biến và phương thức của lớp cơ sở từ lớp dẫn xuất, do khi kế thừa các biến hoặc phương thức ở lớp cơ sở có thể bị định nghĩa trùng tên ở lớp dẫn xuất.
- Từ khóa “new” ngoài việc dùng như một toán tử để khởi tạo đối tượng, nó còn có thể được sử dụng như một bổ từ trong C#. Khi lớp con kế thừa từ lớp cha và tạo ra một phương thức giống tên phương thức lớp cha, lúc này từ khóa new được sử dụng để tạo ra một phiên bản mới của phương này so với phương thức lớp cha, hay chúng ta có thể nói phương thức này sẽ làm ẩn và thay thế phương thức lớp cha.

# Từ khóa “base”, từ khóa “new” 2-2

```
class Animal
{
    public void Eat() { Console.WriteLine("Animal eating"); }
    public void Sound() { Console.WriteLine("Animal sounding"); }
}
class Dog : Animal
{
    //sử dụng từ khóa new để ẩn đi phương thức Eat ở lớp cha
    public new void Sound()
    {
        //sử dụng từ khóa base truy cập phương thức lớp cha
        base.Sound();
        Console.WriteLine("Dog barking");
    }
    //sử dụng từ khóa new để ẩn đi phương thức Eat ở lớp cha
    public new void Eat() {
        //gọi phương thức lớp cha
        Console.WriteLine("Dog eating");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Dog d = new Dog();
        d.Eat();
        d.Sound();
    }
}
```



# Gọi constructor lớp cha 1-2

- Các phương thức khởi tạo sẽ được gọi khi thể hiện của lớp được tạo, trong C# bạn không thể kế thừa phương thức khởi tạo giống các phương thức thường, tuy nhiên bạn có thể gọi constructor lớp cơ sở trong lúc tạo constructor của lớp con.

# Gọi constructor lớp cha 2-2

```
//Lớp cơ sở
class Animal
{
    string name;
    //constructor có tham số
    public Animal(string name)
    {
        this.name = name;
    }
}
//lớp con
class Canime : Animal
{
    public Canime() : base("Lion")//gọi constructor lớp cơ sở
    {
        Console.WriteLine("Derived Canime");
    }
}
```

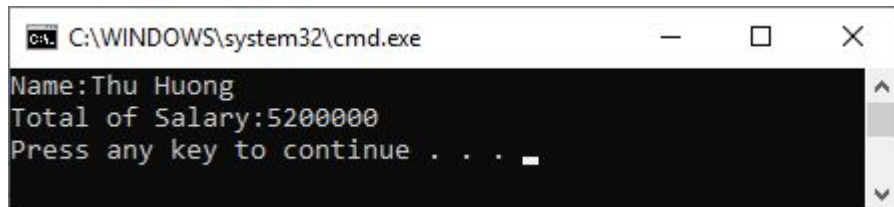
```
//Lớp cha
class Person
{
    private string id;
    private string name;
    private string identityCard;
    public Person(string id, string name, string identityCard)
    {
        this.id = id;
        this.name = name;
        this.identityCard = identityCard;
    }
}
//Lớp con
class Staff:Person
{
    double salary;
    public Staff(string id, string name, string identityCard, double salary)
        :base(id,name,identityCard) //gọi constructor lớp cha
    {
        this.salary = salary;
    }
}
```

# Ghi đè phương thức 1-2

- Ghi đè phương thức (Overriding method) là khi lớp con định nghĩa một phương thức giống hệt lớp cha về mặt hình thức như tên, danh sách tham số, kiểu trả về nhưng nội dung thì khác nhau.
- Để thực thi ghi đè phương thức, bạn cần khai báo phương thức trong lớp cơ sở sử dụng từ khóa **virtual**. Trong lớp dẫn xuất, bạn cần khai báo phương thức ghi đè với từ khóa **override**. Đây là điều bắt bắt buộc.

# Ghi đè phương thức 2-2

```
public class Employee
{
    //khai báo tên và lương cơ bản
    protected string name;
    protected double basepay;
    //khai báo constructor
    public Employee(string name, double basepay)
    {
        this.name = name;
        this.basepay = basepay;
    }
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    // khai báo phương thức ảo có thể được ghi đè ở lớp con
    public virtual double CalculatePay()
    {
        return basepay;
    }
}
```



```
cmd.exe C:\WINDOWS\system32\cmd.exe
Name:Thu Huong
Total of Salary:5200000
Press any key to continue . . . _
```

```
// khai báo lớp nhân viên bán hàng kế thừa từ lớp Employee.
public class Sale : Employee
{
    // khai báo thêm trường thưởng
    private double salesbonus;
    // Khai báo constructor và gọi constructor lớp cha
    public Sale(string name, double basepay,
        double salesbonus) : base(name, basepay)
    {
        this.salesbonus = salesbonus;
    }
    // Ghi đè phương thức CalculatePay
    public override double CalculatePay()
    {
        return basepay + salesbonus;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Employee sale = new Sale("Thu Huong", 5000000, 200000);
        Console.WriteLine("Name:"+sale.Name );
        //CalculatePay của lớp Sale sẽ được gọi
        Console.WriteLine("Total of Salary:" + sale.CalculatePay());
    }
}
```

# Lớp cô lập (Sealed class)

- Lớp sealed là lớp không cho phép các lớp khác kế thừa, khi khai báo lớp bạn bổ sung từ khóa sealed vào trước từ khóa class.

```
//khai báo lớp sealed
sealed class Product
{
    private string id;
    private string name;
    private int price;
    private int quantity;
    public Product(string id, string name, int price, int quantity)
    {
        this.id = id;
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
}
//biên dịch máy sẽ báo lỗi dòng này
class Pen : Product
{
}
}
```

# Tính đa hình

- Tính đa hình trong OOP thể hiện ở 2 loại phương thức Override và Overload

STT	Nạp chồng phương thức	Ghi đè phương thức
1	Nạp chồng phương thức được sử dụng để tăng tính có thể đọc của chương trình	Ghi đè phương thức được sử dụng để cung cấp trình triển khai cụ thể của phương thức mà đã được cung cấp bởi lớp cha của nó
2	Nạp chồng phương thức được thực hiện bên trong lớp	Ghi đè phương thức xuất hiện trong hai lớp mà có mối quan hệ IS-A (kế thừa)
3	Trong Nạp chồng phương thức, tham số phải khác nhau	Trong Ghi đè phương thức, tham số phải là giống nhau
4	Nạp chồng phương thức là ví dụ của đa hình tại compile time	Ghi đè phương thức là ví dụ của đa hình tại runtime
5	Trong Java, Nạp chồng phương thức không thể được thực hiện bởi thay đổi kiểu trả về của phương thức. Kiểu trả về có thể là giống hoặc khác trong Nạp chồng phương thức. Nhưng bạn phải thay đổi tham số	Kiểu trả về phải là giống hoặc covariant trong Ghi đè phương thức

# HỎI ĐÁP







# TRẢI NGHIỆM THỰC HÀNH