Bài 6.8: Tính chất kế thừa

- ✓ Khái niệm và đặc điểm
- ✓ Ghi đè phương thức
- ✓ Lớp Object
- ✓ Cấm kế thừa
- ✓ Ví dụ minh họa và bài tập

Khái niệm và đặc điểm

- ✓Tính chất kế thừa cho phép ta tạo một lớp mới bằng cách tái sử dụng, sửa đổi, mở rộng một lớp đã tồn tại trước đó.
- ✓Lớp mà bị lớp khác kế thừa gọi là lớp cha(father, parent, super class), lớp cơ sở(base class).
- ✓Lớp đi kế thừa lớp khác gọi là lớp con(child, sub class), lớp dẫn xuất(derived class).
- ✓Một lớp con chỉ có một lớp cha trực tiếp nhưng có thể có nhiều lớp cha gián tiếp và triển khai nhiều interface.
- ✓C# chỉ hỗ trợ đơn kế thừa với class. Để đạt được đa kế thừa, sử dụng interface.
- ✓Lớp con kế thừa tất cả các thành phần của lớp cha ngoại trừ phương thức khởi tạo(cả static và instance constructor), Finalizers.

Khái niệm và đặc điểm

- ✓ Access modifier của các thành phần trong lớp cha ảnh hưởng tới khả năng nó được nhìn thấy và truy cập từ lớp con:
  - ✓ Các thành phần private của lớp cha chỉ thuộc về lớp cha, không thể nhìn thấy và truy cập trực tiếp từ lớp con.
  - ✓ Các thành phần protected của lớp cha có thể được nhìn thấy và truy cập từ trong lớp con.
  - ✓ Các thành phần internal của lớp cha chỉ khả kiến với lớp con trong cùng assembly với lớp cha.
  - ✓ Các thành phần public có thể khả kiến với mọi lớp khác cũng như lớp con.
- ✓ Kế thừa có tính bắc cầu: nếu lớp B kế thừa lớp A, lớp C kế thừa B thì A kế thừa C.

Cú pháp kế thừa

- ✓Cú pháp kế thừa: **class DerivedClassName : BaseClassName {…}**
- ✓Trong đó:
  - ✓ DerivedClassName là tên của lớp con.
  - ✓ BaseClassName là tên lớp cha.
  - ✓ Phần thân lớp con chứa trong {} như thân lớp bình thường.
- ✓Một lớp con chỉ được kế thừa 1 lớp cha trực tiếp duy nhất.
- ✓Lớp con sẽ có thêm các đặc trưng, hành động mà lớp cha không có.
- ✓Khi đi từ lớp cha xuống lớp con, tính cụ thể hóa tăng dần.
- ✓Khi đi từ lớp con lên lớp cha, tính đơn giản hóa tăng dần.
- ✓Kế thừa thể hiện mối quan hệ is-a(là một).
- ✓Ví dụ: người là động vật, chim là động vật, động vật không xương sống là động vật.

Ví dụ

```
class Person
{
    2 references
    public string FirstName { get; set; }
    1 reference
    public string LastName { get; set; }
    1 reference
    public string MidName { get; set; }
    1 reference
    public int Age { get; set; }

    0 references
    public Person () ...

    1 reference
    public Person (string firstName, string lastName, string midName, int age)...

    0 references
    public void Sleep()...
}

3 references
class Student : Person
{
    1 reference
    public string StudentId { get; set; }
    1 reference
    public float Gpa { get; set; }

    0 references
    public Student() ...

    1 reference
    public Student(string id, float gpa)...

    0 references
    public void DoHomework(string subject)...
}
```

Các thuộc tính riêng

Hành động của riêng lớp con

Ví dụ

```
0 references
class Lesson68
{
    0 references
    static void Main()
    {
        Student student = new Student("SV1001", 3.55f);
        // truy cập đến thành phần của lớp cha:
        student.FirstName = "Long";
        student.LastName = "Nguyen";
        student.MidName = "Hoang";
        student.Age = 20;
        Console.WriteLine($"Ho va ten: {student.LastName} {student.MidName} {student.FirstName}");
        // truy cập đến thành phần riêng của lớp con
        Console.WriteLine("Ma sinh vien: " + student.StudentId);
        Console.WriteLine("Diem TB: " + student.Gpa);
    }
}

5 references
class Pe
{
```

```
C:\WINDOWS\system32\cmd.exe
Ho va ten: Nguyen Hoang Long
Ma sinh vien: SV1001
Diem TB: 3.55
Press any key to continue . . .
```

Ghi đè phương thức

- ✓Trong kế thừa, các lớp con thường có cùng hành vi nhưng khác cách triển khai của hành vi đã có trong lớp cha.
- ✓Khi muốn định nghĩa lại hành vi của lớp cha theo cách cụ thể hóa của riêng lớp con, ta phải ghi đè phương thức.
- ✓Ví dụ: động vật nói chung đều di chuyển. Nhưng cá di chuyển bằng cách bơi, mèo đi bằng 4 chân, người đi bộ hoặc đi bằng phương tiện(trên bộ, biển, trên không).
- ✓Theo mặc định, các phương thức của lớp cha là non-virtual và không cho phép ghi đè.
- ✓Để phương thức nào đó ở lớp cha cho phép ghi đè, ta thêm keyword virtual vào phần khai báo của phương thức đó. Đặt trước kiểu trả về của phương thức.
- ✓Khi ghi đè phương thức, giữ nguyên tên, kiểu trả về, tham số của phương thức ở lớp cha. Bổ sung keyword override vào khai báo phương thức và tiến hành sửa đổi nội dung trong thân phương thức theo mong muốn của lớp con.
- ✓Không thay đổi mục đích sử dụng của phương thức được override.

Ví dụ

```
1 reference
class Father
{
    // other members
    // ...
    0 references
    public void DoSomething(string something)
    {
        // statements
    }
}

0 references
class Child: Father
{
    0 references
    public override void DoSomething(string something) // error code CS0506
    {
        // do something here
    }
}
```

Ví dụ

```
1 reference
class Father
{
    // other members
    // ...
    1 reference
    public virtual void DoSomething(string something)
    {
        // statements
    }
}

0 references
class Child : Father
{
    1 reference
    public override void DoSomething(string something) // ok
    {
        // do something here
    }
}
```

Ghi đè phương thức

✓Trong phương thức ghi đè của lớp con ta có thể gọi tới phương thức gốc ở lớp cha qua cú pháp: **base.MethodName(params);**

```
1 reference
class Father
{
    // other members
    // ...
    2 references
    public virtual void DoSomething(string something)
    {
        // statements
    }
}

0 references
class Child : Father
{
    2 references
    public override void DoSomething(string something) // ok
    {
        base.DoSomething(something); // gọi phương thức gốc của lớp cha
        // do something here
    }
}
```

Ví dụ

```
class Shape // lớp cha
{
    0 references
    public string Name { get; set; }
    2 references
    public virtual double Area()... // tính diện tích

    2 references
    public virtual double Perimeter()... // tính chu vi
}

0 references
class Circle : Shape // lớp hình tròn kế thừa Shape
{
    3 references
    public double Radius { get; set; }

    1 reference
    public override double Area()
    {
        return Math.PI * Radius * Radius;
    }

    1 reference
    public override double Perimeter()
    {
        return 2 * Math.PI * Radius;
    }
}
```

Ví dụ

```
0 references
class Rectangle : Shape // lớp hình chữ nhật kế thừa Shape
{
    2 references
    public double Width { get; set; }
    2 references
    public double Height { get; set; }

    1 reference
    public override double Perimeter()
    {
        return 2 * (Width + Height);
    }

    1 reference
    public override double Area()
    {
        return Width * Height;
    }
}
```

Lớp Object

- ✓Trong thế giới C#, lớp Object là lớp cha, tổ tiên của tất cả các lớp khác.
- ✓Theo mặc định khi ta tạo lớp mới, lớp cha ngầm định của nó sẽ là Object nếu ta không chỉ rõ lớp cha trực tiếp một cách tường minh.
- ✓Do kế thừa từ Object nên mọi phương thức của lớp Object luôn sẵn dùng trong tất cả các đối tượng được tạo ra.
- ✓Lớp con có thể ghi đè các phương thức sau đây của lớp Object:
  - ✓ Equals: để so sánh sự tương đương của hai đối tượng.
  - ✓ Finalize: thực hiện dọn dẹp trước khi một đối tượng bị thu hồi bộ nhớ.
  - ✓ GetHashCode: sinh một giá trị số tương ứng với giá trị của đối tượng nhằm hỗ trợ sử dụng bảng băm.
  - ✓ ToString: tạo và trả về chuỗi kí tự mô tả thông tin một đối tượng của lớp.

Ví dụ ghi đè phương thức của lớp Object

```
class Student
{
    4 references
    public string Id { get; set; }
    2 references
    public string FullName { get; set; }
    2 references
    public float Gpa { get; set; }

    0 references
    public Student(string id, string fullName, float gpa)...

    0 references
    public override bool Equals(object obj)
    {
        if(obj.GetType() != this.GetType())
        {
            return false;
        }
        var other = (Student)obj;
        return Id.CompareTo(other.Id) == 0; // mã sinh viên trùng nhau
    }

    0 references
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }

    0 references
    public override string ToString() => $"Student[Id={Id}, FullName={FullName}, Gpa={Gpa}]";
}
```

Cấm kế thừa

✓Để một lớp nào đó không được phép kế thừa, ta sử dụng keyword sealed.

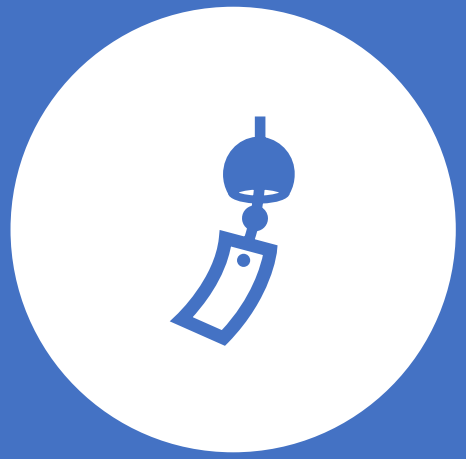
✓Các lớp tiện ích là ví dụ điển hình của những lớp không cho phép lớp khác kế thừa nó.

```
1 reference
sealed class Utils
{
    0 references
    public double Add(double a, double b)...

    0 references
    public string[] SplitWords(string input)...

    0 references
    public bool IsPrimeNumber(int n)...
}

0 references
class ChildUtils : Utils // error!
{
    // statements
}
```



Nội dung tiếp theo

Mối quan hệ thành phần(has a)