Bài: Đa hình trong Lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Đa hình trong Lập trình hướng đối tượng

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở các bài học trước, chúng ta đã cùng nhau tìm hiểu về **KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG** . Hôm nay chúng ta sẽ cùng tìm hiểu về **Đa hình trong C#**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- **BIẾN** và **KIỂU DỮ LIỆU** trong C#
- **TOÁN TỬ TRONG C#**
- **CÂU ĐIỀU KIỆN TRONG C#**
- **CẤU TRÚC CƠ BẢN CỦA VÒNG LẶP TRONG C#**
- **CẤU TRÚC HÀM CƠ BẢN TRONG C#**
- **TỔNG QUAN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**
- **CLASS TRONG C#**
- **CÁC LOẠI PHẠM VI TRUY CẬP TRONG C#**
- **KẾ THỪA TRONG C#**

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Khái niệm tính đa hình.
- Từ khoá virtual và từ khoá override.
- Lớp trừu tượng và phương thức thuần ảo.

Khái niệm tính đa hình

Tính đa hình là hiện tượng các đối tượng thuộc các lớp khác nhau có thể hiểu cùng 1 thông điệp theo các cách khác nhau.

Một ví dụ về đa hình trong thực tế. Ta có 3 con vật: chó, mèo, lợn. Cả 3 con vật này đều là động vật. Nhưng khi ta bảo cả 3 động vật kêu thì con chó sẽ kêu gâu gâu, con mèo sẽ kêu meo meo và con heo sẽ kêu ẹt ẹt.

Trong ví dụ trên 3 con vật: chó, mèo, lợn xem như là các đối tượng. Việc ta bảo 3 động vật kêu chính là thông điệp. Rõ ràng cả 3 con vật có thể hiểu cùng 1 thông điệp là kêu theo các cách khác nhau.

Để thể hiện được **tính đa hình**:

- Các lớp phải có quan hệ kế thừa với cùng 1 lớp cha nào đó.
- Phương thức đa hình phải được ghi đè (override) ở các lớp con (sẽ được trình bày ngay sau đây).

Từ khoá virtual và từ khoá override

Virtual là từ khoá dùng để khai báo 1 phương thức ảo (phương thức ảo là phương thức có thể ghi đè được).

Override là từ khoá dùng để đánh dấu phương thức ghi đè lên phương thức của lớp cha.

Lưu ý:

- Chỉ có thể ghi đè lên phương thức virtual hoặc abstract (sẽ trình bày ngay sau đây).
- Tính đa hình chỉ được thể hiện khi đã ghi đè lên phương thức của lớp cha.

Ví dụ minh hoạ:

Ta có 3 lớp Animal, Cat, Dog. Trong đó Cat và Dog kế thừa từ lớp Animal. Trong các lớp đều có phương thức **Speak()**.

C#:

```
class Animal
    {

        public void Speak()
        {
            Console.WriteLine(" Animal is speaking. . .");
        }
    }


    class Cat : Animal
    {
        public void Speak()
        {
            Console.WriteLine(" Cat is speaking. . .");
        }
    }


    class Dog : Animal
    {
        public void Speak()
        {
            Console.WriteLine(" Dog is speaking. . .");
        }
    }
```

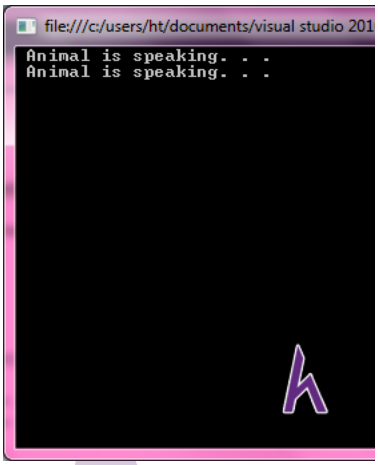
Ta có đoạn chương trình trong hàm **main** như thế này:

C#:

```
Animal cat = new Cat();

            Animal dog = new Dog();
            cat.Speak();
            dog.Speak();
```

Ta mong muốn chương trình sẽ gọi đúng phương thức **Speak()** của lớp đã được cấp phát vùng nhớ. Nhưng thực tế không phải vậy.



Lúc này ta cần phải override phương thức **Speak()** của lớp cha (lớp Animal) và để override được thì ta cần khai báo phương thức **Speak()** của lớp cha là phương thức ảo (virtual).

C#:

```
class Animal
    {
        public virtual void Speak()
        {
            Console.WriteLine(" Animal is speaking. . .");
        }
    }



    class Cat : Animal
    {
        public override void Speak()
        {
            Console.WriteLine(" Cat is speaking. . .");
        }
    }


    class Dog : Animal
    {
        public override void Speak()
        {
            Console.WriteLine(" Dog is speaking. . .");
        }
    }
```

Chạy lại hàm **main** trên ta được:

Đây cũng chính là ví dụ sử dụng tính đa hình.

Ta thấy 2 đối tượng dog, cat được cấp phát 2 vùng nhớ thuộc 2 lớp 2 khác nhau nhưng khi cùng gọi phương thức **Speak()** thì đối tượng tham chiếu đến vùng nhớ của lớp nào sẽ được gọi đúng phương thức của lớp đó.

Lớp trừu tượng và phương thức thuần ảo

Phương thức thuần ảo là 1 phương thức ảo và không có định nghĩa bên trong.

Lớp trừu tượng là lớp chứa phương thức thuần ảo.

Abstract là từ khoá dùng để khai báo 1 lớp trừu tượng hoặc 1 phương thức thuần ảo.

Xét lại ví dụ trên, Ở đây ta xem lại phương thức **Speak()** của lớp Animal ta nhận thấy phần định nghĩa của phương thức này chỉ là hình thức sau đó cũng sẽ bị các lớp kế thừa ghi đè lên.

Việc định nghĩa nội dung phương thức không có tác dụng gì vậy tại sao ta lại phải định nghĩa chúng?

Câu trả lời đã được C# giải đáp qua từ khoá abstract. Ở đây ta sử dụng abstract để nhấn mạnh 2 điều:

- Phương thức **Speak()** có thể ghi đè (override).
- Phương thức **Speak()** không có định nghĩa gì bên trong.

Để khai báo lớp trừu tượng và phương thức thuần ảo ta chỉ cần thêm khoá abstract vào trước tên lớp và tên phương thức.

C#:

```
abstract class Animal
    {
        /*
            Khai báo phương thức thuần ảo nên không cần định nghĩa nội dung cho phương thức
         */

        public abstract void Speak();

    }
```

Khi chạy chương trình mọi thứ vẫn ra đúng như mong muốn.

Lưu ý:

- Khi kế thừa 1 lớp trừu tượng bạn bắt buộc phải override tất cả các phương thức thuần ảo nhằm đảm bảo tính hợp lệ cho chương trình.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Khái niệm tính đa hình.
- Từ khoá virtual và từ khoá override.
- Lớp trừu tượng và phương thức thuần ảo.

Bài sau chúng ta sẽ tìm hiểu về **INTERFACE TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".