Bài: Class trong Lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Class 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 **TỔNG QUAN VỀ LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**. Hôm nay chúng ta sẽ cùng tìm hiểu về **Class 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#**
- **MẢNG MỘT CHIỀU TRONG C#**

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

- Class trong C# là gì?
- Khai báo, khởi tạo và sử dụng class trong C#.
- Phương thức khởi tạo, phương thức huỷ bỏ.
- So sánh giữa Struct và Class.

Class trong C# là gì?

Class trong C# chính là cách thể hiện khái niệm về lớp trong lập trình hướng đối tượng.

Một **class** trong C# có các thành phần như:

- Thuộc tính: là các thành phần dữ liệu hay còn gọi là các biến.
- Phương thức: là các hàm thành phần thể hiện các hành vi của một đối tượng thuộc lớp.
- Phương thức khởi tạo.
- Phương thức huỷ bỏ.

Class trong C# thực chất là một kiểu dữ liệu mới do người dùng tự định nghĩa.

Khai báo, khởi tạo và sử dụng class trong C#

Cú pháp:

```
class <tên lớp>
{
        <Phạm vi truy cập> <Các thành phần của lớp>;
}
```

Trong đó:

- **<tên lớp>** là tên do người dùng đặt và tuân theo quy tắc đặt tên đã trình bày trong bài **BIẾN TRONG C#**.
- **<Phạm vi truy cập>** bao gồm các từ khoá như public, protected, private, static, . . . sẽ được trình bày trong bài **CÁC LOẠI PHẠM VI TRUY CẬP TRONG C#**.
- **<Các thành phần của lớp>** bao gồm các biến, phương thức của lớp.
  - Các biến được khai báo như khai báo biến đã học trong bài **BIẾN TRONG C#**.
  - Các phương thức (hàm) được khai báo như khai báo hàm đã học trong bài **HÀM TRONG C#**.

Ví dụ:

:

```
class Animal
    {

        public double Weight;
        public double Height;

        public void Run()
        {
            Console.WriteLine(" Animal is running. . .");
        }
    }
```

- Với khai báo trên ta đã có 1 kiểu dữ liệu mới tên là **Animal**. Và ta hoàn toàn có thể khai báo biến và sử dụng nó.
- Từ khoá **public** là chỉ phạm vi truy cập của các thành phần bên trong **class** (sẽ được trình bày chi tiết trong bài **CÁC LOẠI PHẠM VI TRUY CẬP TRONG C#**). Hiện tại bạn chỉ cần hiểu ghi **public** là để khi sử dụng ta có thể truy cập các thành phần này.
- Lớp **Animal** có 2 thuộc tính là **Weight**, **Height** và 1 phương thức là **Run**. Như vậy mọi đối tượng thuộc lớp này đều có 3 thành phần trên.
- Nếu để ý các bạn sẽ nhận ra rằng mọi phương thức có sẵn trong C# đều được nằm trong lớp nào đó (khác hoàn toàn với phong cách lập trình hướng thủ tục – phương thức độc lập với dữ liệu). Điều này có thể giải thích là do C# là ngôn ngữ thuần hướng đối tượng nên mọi thứ bên trong nó đều theo phong cách lập trình hương đối tượng.

Khởi tạo:

Bạn có thể khởi tạo 1 đối tượng thuộc lớp thông qua toán tử new.

Ví dụ:

:

```
Animal Dog = new Animal();
```

Class là kiểu dữ liệu tham chiếu vì thế đối tượng dữ liệu thực sự được lưu trên heap.

Sử dụng:

Về cơ bản thì **class** được sử dụng tương tự như **struct**. Các bạn có thể tham khảo ví dụ ở bài **STRUCT TRONG C#**.

Để gọi đến các thuộc tính bên trong lớp:

> **<tên đối tượng>. <tên thuộc tính>;**

Để gọi đến các phương thức bên trong lớp:

> **<tên đối tượng> . <tên phương thức> (<danh sách tham số nếu có>);**

Ở đây mình muốn tập trung vào cách bạn khai báo và sử dụng phương thức bên trong lớp như thế nào. Xét bài toán đơn giản sau: Khởi tạo các đối tượng thuộc lớp Animal lần lượt gọi phương thức in ra chiều cao và cân nặng của các loài động vật.

:

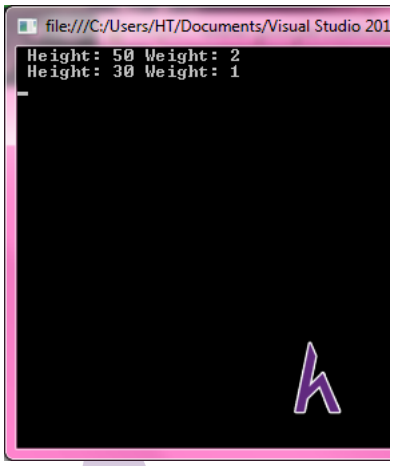
```
class Animal
    {

        public double Weight;
        public double Height;

        public void Info()
        {

            /*
                Các phương thức bên trong lớp có thể gọi đến các thành phần khác (bao gồm thuộc tính và phương thức) trong lớp đó.
                Giá trị của các thuộc tính này có thể được khởi tạo đâu đó trong lớp hoặc từ bên ngoài truyền vào.
             */

            Console.WriteLine(" Height: " + Height + " Weight: " + Weight);

        }
    }
```

Trong hàm **main**:

:

```
/*
               Khởi tạo 2 đối tượng thuộc lớp Animal là:
                + Dog có chiều cao 50cm và cân nặng 2kg.
                + Cat có chiều cao 30cm và cân nặng 1kg.
*/
            Animal Dog = new Animal();
            Dog.Weight = 2; // gán giá trị cho các thuộc tính của đối tượng
            Dog.Height = 50;


            Animal Cat = new Animal();
            Cat.Weight = 1;
            Cat.Height = 30;


            Dog.Info(); // gọi phương thức của đối tượng
            Cat.Info();
```

Kết quả khi chạy đoạn code trên:



- Bạn có thể thấy tuỳ vào giá trị của từng đối tượng mà phương thức Info in ra đúng giá trị tương ứng của đối tượng đó.

Phương thức khởi tạo, phương thức huỷ bỏ

Trong thế giới thực khi 1 sự vật nào đó được sinh ra thì nó sẽ mang sẵn trong mình những đặc điểm nhất định và mọi sự vật cùng loài với nó đều như vậy.

Vì lập trình hướng đối tượng là phương pháp giúp ánh xạ thế giới thực vào thế giới lập trình một cách dễ dàng nên từ đó đã xuất hiện 2 khái niệm **phương thức khởi tạo** và **phương thức huỷ bỏ** để thể hiện ý trên.

Phương thức khởi tạo

Phương thức khởi tạo (Constructor) là những phương thức đặc biệt được gọi đến ngay khi khởi tạo 1 đối tượng nào đó.

Đặc điểm

- Có tên trùng với tên lớp.
- Không có kiểu trả về.
- Được tự động gọi khi 1 đối tượng thuộc lớp được khởi tạo.
- Nếu như bạn không khai báo bất kỳ phương thức khởi tạo nào thì hệ thống sẽ tự tạo ra phương thức khởi tạo mặc định không đối số và không có nội dung gì.
- Có thể có nhiều constructor bên trong 1 lớp.

Có 2 loại phương thức khởi tạo:

- Phương thức khởi tạo **không đối số**:
  - Là phương thức khởi tạo không có bất kỳ tham số truyền vào nào.
  - Thường dùng để khởi tạo các giá trị mặc định cho các thuộc tính bên trong class khi khởi tạo đối tượng (giá trị mặc định này do người lập trình quyết định).
- Phương thức khởi tạo **có đối số**:
  - Là phương thức khởi tạo có tham số truyền vào. Và khi khởi tạo đối tượng để phương thức này được gọi ta cần truyền đầy đủ các tham số.
  - Thường dùng để khởi tạo các giá trị cho các thuộc thuộc tính bên trong class khi khởi tạo đối tượng (các giá trị này do người khởi tạo đối tượng truyền vào).

Ví dụ

Ta có khai báo lớp như sau:

:

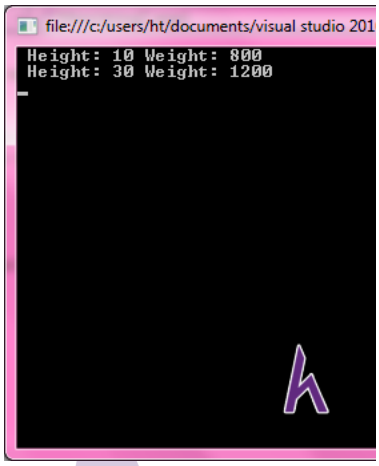
```
class Cat
    {

        public double Weight;
        public double Height;

        /*
            Constructor không có tham số
         */

        public Cat()
        {
            Weight = 800;
            Height = 10;
        }


        /*
            Constructor có tham số
         */

        public Cat(int w, int h)
        {

            Weight = w;
            Height = h;

        }


        public void Info()
        {
            Console.WriteLine(" Height: " + Height + " Weight: " + Weight);
        }
    }
```

Trong hàm **main**:

:

```
/*
                Để ý khi bạn new Cat bạn không truyền tham số vào thì constructor mặc định (constructor không tham số)
             sẽ tự động được gọi để khởi tạo các giá trị mặc định cho các thuộc tính của đối tượng BlackCat.

 */
            Cat BlackCat = new Cat();

            BlackCat.Info(); // In ra thông tin Weight, Height có thực sự được khởi tạo giá trị chưa.

            /*
                Lần này ta khởi tạo đối tượng và có truyền 2 tham số như vậy constructor có tham số sẽ được gọi.
                Như vậy các thuộc tính Weight, Height sẽ không mang giá trị mặc định nữa mà sẽ được gán theo giá trị mà người dùng
             truyền vào.
             */

            Cat WhiteCat = new Cat(1200, 30);
            WhiteCat.Info(); // In ra thông tin Weight, Height để xem có theo mong muốn chưa.
```

Kết quả khi chạy đoạn code trên:



Lưu ý

Khi 1 đối tượng được khởi tạo chỉ có 1 **constructor** phù hợp nhất được gọi cho dù trong lớp có nhiều **constructor**.

Qua ví dụ bạn có thể thấy cách ta truyền tham số khi khởi tạo đối tượng sẽ gian tiếp ám chỉ constructor nào được gọi vì thế bạn cần **truyền đúng** và **truyền đủ** các tham số để C# có thể gọi đúng constructor như ý muốn.

Các tham số của **constructor** thường sẽ mang các giá trị tương ứng cho các thuộc tính bên trong lớp vì thế bạn nên:

- Khai báo chúng có cùng kiểu dữ liệu với thuộc tính tương ứng.
- Đặt tên chúng gợi nhớ đến thuộc tính tương ứng để tránh nhầm lẫn.
- Không nên khai báo dư tham số sẽ dẫn đến khó hiểu cho người dùng.

Phương thức huỷ bỏ

Phương thức huỷ bỏ (destructor) là phương thức đặc biệt được gọi đến trước khi 1 đối tượng bị thu hồi.

Đặc điểm

- Có tên trùng với tên lớp nhưng để phân biệt với constructor thì ta thêm dấu "~" vào trước tên lớp.
- Không có kiểu trả về.
- Được tự động gọi khi 1 đối tượng thuộc lớp kết thúc "vòng đời" của nó thông qua bộ thu dọn rác tự động GC (Garbage Collection).
- Nếu bạn không khai báo destructor thì C# sẽ tự động tạo ra 1 destructor mặc định và không có nội dung gì.
- Chỉ có 1 destructor duy nhất trong 1 lớp.

Vì bộ GC của C# có cơ chế tự động phát hiện đối tượng không còn được sử dụng nữa và thực hiện thu hồi vùng nhớ của nó nên bạn không cần phải viết tường minh việc huỷ vùng nhớ của nó. Việc bạn có thể làm viết những thứ bạn muốn làm khi đối tượng bị huỷ vào đây thôi!

So sánh giữa Struct và Class

Nhìn chung **struct** và **class** có khá nhiều điểm chung nhưng class có phần mạnh hơn. Hãy còn xem tại sao class lại mạnh hơn và mạnh hơn ở điểm nào nhé!

Struct	Class
Kiểu tham trị	Kiểu tham chiếu
Không có destructor	Có destructor
Không thể khai báo phương thức khởi tạo mặc định (không đối số)	Có thể khai báo phương thức khởi tạo mặc định
Không có khả năng kế thừa	Có thể kế thừa

Ngoài ra nếu 1 class là tham số truyền vào thì mọi thay đổi bên trong hàm sẽ tự động cập nhật ra bên ngoài cho dù không có từ khoá out hoặc ref (xem lại bài **TỪ KHÓA REF VÀ OUT**) còn struct thì không.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Class trong C# là gì?
- Khai báo, khởi tạo và sử dụng class trong C#.
- Phương thức khởi tạo, phương thức huỷ bỏ.
- So sánh giữa Struct và Class.

Bài sau chúng ta sẽ tìm hiểu về **CÁC LOẠI PHẠM VI TRUY CẬP TRONG C#**.

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**".