Tìm hiểu từ khóa this và static trong c#

Tìm hiểu từ khóa this và static trong c#

Trong bài học này chúng ta cùng tìm hiểu về từ khóa this và static trong C#. Vậy ý nghĩa của từng từ khóa như thế nào.

1. This

- Trước khi tìm hiểu về từ khóa this là gì, chúng ta xem ví dụ sau đây, giả sử chúng ta cho tham số của hàm xây dựng cùng tên với dữ liệu thành viên của lớp.

```
using System;
namespace ConsoleApp1
{
    class People
    {
        int old;
        string name;
        double height;
        public People(int old, string name, double height)
        {
            Console.WriteLine("\n---Goi ham xay dung co 3 tham so---");
            old = old;
            name = name;
            height = height;
        }
        public void Show()
        {
            Console.WriteLine("Old: " + old + ",\nName: " + name + ",\nHeight: " + height);
        }
    }
```

```
class Program
 {
     static void Main(string[] args)
     {
         People p1 = new People(20, "Nguyen Van A", 180);
         p1.Show();
         Console.ReadKey();
     }
 }
}
```

- kết quả là old = 0, name = , height = 0.
- Từ kết quả trên chúng ta thấy kết quả không như chúng ta mong đợi (đáng lẻ kết quả phải là old = 20, name = Nguyen Van A, height = 180).
- Như vậy khi tên tham số trùng với tên của dữ liệu thành viên của lớp, thì chương trình không hiểu nó là dữ liệu thành viên mà nó hiểu đây là tham số truyền vào.
- Chúng ta sẽ thay đổi đoạn code trên như sau:

```
using System;
namespace ConsoleApp1
{
    class People
    {
        int old;
        string name;
        double height;
        public People(int old, string name, double height)
        {
            Console.WriteLine("\n---Goi ham xay dung co 3 tham so---");
            this.old = old;
            this.name = name;
            this.height = height;
        }
        public void Show()
        {
            Console.WriteLine("Old: " + old + ",\nName: " + name + ",\nHeight: " + height);
        }
    }
```

```
 class Program
    {
        static void Main(string[] args)
        {
            People p1 = new People(20, "Nguyen Van A", 180);
            p1.Show();
            Console.ReadKey();
        }
    }
}
```

kết quả là old = 20, name = Nguyen Van A, height = 180

Như vậy chúng ta hiểu từ khóa this trong c# một cách đơn giản như sạu:

Từ khóa this trong c# được sử dụng để tham chiếu đến thể hiện (instance) hiện tại của lớp. Nó cũng được sử dụng để phân biệt giữa các tham số phương thức và các dữ liệu thành viên của lớp nếu cả hai đều có cùng tên.

2. Static

Trước khi tìm hiểu về từ khóa static trong c# là gì, chúng ta cùng xem ví dụ sau đây, giả sử chúng ta có lớp People (People đang đề cập ở đây là người Việt Nam) có thuộc tính quốc tịch (*nationality*).

```
using System;
namespace ConsoleApp1
{
    class People
    {
        int id;
        string name;
        string nationality;
        public People(int id, string name, string nationality)
        {
            this.id = id;
            this.name = name;
            this.nationality = nationality;
        }
        public void Show()
        {
            Console.WriteLine("Id: " + id + ",\nName: " + name + ",\nationality: " + nationality);
        }
    }
```

```
    class Program
    {
        static void Main(string[] args)
        {
            People p1 = new People(1, "Nguyen Van A", "Viet Nam");
            People p2 = new People(2, "Nguyen Van B", "Viet Nam");
            People p3 = new People(3, "Nguyen Van C", "Viet Nam");
            Console.ReadKey();
        }
    }
}
```

- Như vậy, mỗi lần tạo đối tượng sẽ cấp 3 vùng nhớ cho mỗi đối tượng đó là id, name, nationality. Lớp chúng ta đang xét là lớp người Việt Nam, thì tất cả đối tượng được tạo ra cùng chung giá trị là Việt Nam. Nếu có 100 đối tượng hoặc 10000 đối tượng được khởi tạo thì tốn 100 hoặc 10000 biến nationality được tạo ra.
- Từ khóa static trong C# giúp giải quyết vấn đề lãng phí bộ nhớ nói trên.
- Static ở đây là thuộc về lớp, không thuộc về thể hiện của lớp. Vì vậy khi khởi tạo đối tượng thì biến static không được khởi tạo theo đối tượng

Static Field

Một trường được khai báo có từ khóa static trước tên trường, được gọi là trường tĩnh (*static field*). Không giống như trường bình thường của đối tượng là nhận bộ nhớ mỗi lần bạn tạo đối tượng, chỉ có một bản sao của trường tĩnh được tạo trong bộ nhớ. Nó được chia sẻ cho tất cả các đối tượng.

```
using System;
namespace ConsoleApp1
{
    class People
    {
        int id;
        string name;
        static string nationality = "Viet Nam";
        public People(int id, string name)
        {
            this.id = id;
            this.name = name;

        }
        public void Show()
        {
            Console.WriteLine("Id: " + id + ",\nName: " + name + ",\nationality: " + nationality + "\n\n");
        }
    }
```

```
    class Program
    {
        static void Main(string[] args)
        {
            People p1 = new People(1, "Nguyen Van A");
            p1.Show();
            People p2 = new People(2, "Nguyen Van B");
            p2.Show();
            People p3 = new People(3, "Nguyen Van C");
            p3.Show();
            Console.ReadKey();
        }
    }
}
```

- Do trường tĩnh là thuộc về lớp, nên muốn truy xuất hay thay đổi giá trị của trường tĩnh chúng ta không cần phải khởi tạo đối tượng. Chỉ cần lấy tên lớp chấm tên trường static.

Static class

- static class cũng giống như lớp khác, tuy nhiên nó không có thể hiện. Nghĩa là chúng ta không tạo được đối tượng từ static class.
- Một số điểm cần lưu ý:
- static class chỉ chứa thành viên là static
- static class không có thể hiện
- static class không chứa hàm xây dựng
- chúng ta vẫn có thể gọi và sử dụng thành viên của static class

```
using System;
namespace ConsoleApp1
{
    public static class MyMath
    {
       public static float PI = 3.14f;
        public static int squared(int n) { return n * n; }
    }
```

```
class Program
 {
    static void Main(string[] args)
    {
       int squared;
       float pi = MyMath.PI;
       squared = MyMath.squared(10);
       Console.WriteLine("squared: " + squared);
       Console.WriteLine("\nPI: " + pi);
       Console.ReadKey();
    }
 }
```