

第34章 thisとは何でしょうか



メソッドは、何らかのクラスに属しています。メソッドが呼び出されると、そのオブジェクトの参照がthisとして渡されます。

staticなメソッドでは、このようなことは起こりません。インデクサのthisも同様です。従ってstaticなインデクサが存在しないというのも納得できます。thisはどんなときに使うのでしょうか。次の例を見てください。

```
// this01.cs

using System;

class MyClass
{
    public int x;

    public void show(int x)
    {
        Console.WriteLine("x = {0}", x);
        Console.WriteLine("this.x = {0}", this.x);
    }
}

class this01
{
    public static void Main()
    {
        MyClass mc = new MyClass();
        mc.x = 100;
        mc.show(20);
    }
}
```

MyClassクラスには、インスタンスフィールドのxがあります。

このクラスには、またxというパラメータを持つshowというインスタンスメソッドがあります。

さて、showメソッドでは最初に

```
Console.WriteLine("x = {0}", x);
```

していますが、この時xはパラメータのxとなります。一方、

```
Console.WriteLine("this.x = {0}", this.x);
```

this.xとすると、「このオブジェクトのx」という意味になりフィールドのxを指すことになります。

この場合、show(int y)とでもすれば、わざわざthisを使うまでもないのですが……。

実行結果は当然次のようになります。

さて、第24章でコンストラクタのオーバーロードをやりました。あるコンストラクタがthisを使って、他のバージョンのコンストラクタを呼び出すことができます。

クラス名 (パラメータリスト) : this (パラメータリスト) { ... }

こうすると、このコンストラクタが実行される前に、this/パラメータリストと一致するコンストラクタが呼び出されます。コンストラクタが多数オーバーロードされていて、似たような処理をするときには便利かもしれません。

```
// this02.cs

using System;

class MyClass
{
    int x, y, z;

    public MyClass()
    {
        Console.WriteLine("引数なしコンストラクタが呼ばれました");
        x = 1;
    }
}
```

```

        y = 1;
        z = 1;
    }
    public MyClass(int a)
        : this()
    {
        x = a;
    }

    public MyClass(int a, int b)
        : this()
    {
        x = a;
        y = b;
    }

    public MyClass(int a, int b, int c)
    {
        x = a;
        y = b;
        z = c;
    }

    public void show()
    {
        Console.WriteLine("x = {0}, y = {1}, z = {2}", x, y, z);
    }
}

```

```

class this02
{
    public static void Main()
    {
        string strLine = "-----";

        MyClass mc1 = new MyClass();
        mc1.show();

        Console.WriteLine(strLine);

        MyClass mc2 = new MyClass(5);
        mc2.show();

        Console.WriteLine(strLine);

        MyClass mc3 = new MyClass(10, 20);
        mc3.show();

        Console.WriteLine(strLine);

        MyClass mc4 = new MyClass(100, 200, 300);
        mc4.show();

    }
}

```

MyClassクラスにはx, y, zというフィールドがあります。このフィールドのデフォルトの値を1にしたいとします。また、xのみ、x,yのみ、x,y,zのすべてをユーザーの指定した値に初期化もできるようにしたいとします。

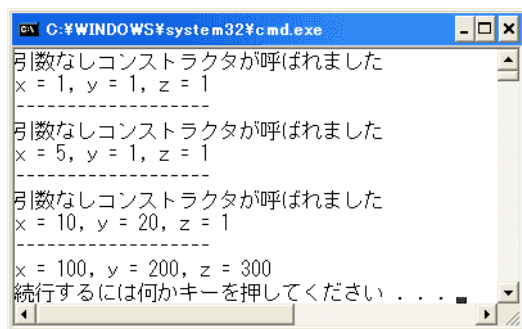
コンストラクタは、4種類のオーバーロードバージョンを用意しました。

引数なしバージョンでは、x, y, zの値を全部1で初期化しています。

さて、引数を1つだけ持つバージョンでは MyClass(int a) : this()として、まず引数なしバージョンを呼んでx,y,zの値を全部1にしています。その後xの値をユーザーが指定したaに設定しています。

同様のことを引数2個バージョンでも行っています。引数3個バージョンでは、意味がないのでthis呼び出しを行っていません。

では、実行結果を見てみましょう。



```
C:\WINDOWS\system32\cmd.exe
引数なしコンストラクタが呼ばれました
x = 1, y = 1, z = 1
-----
引数なしコンストラクタが呼ばれました
x = 5, y = 1, z = 1
-----
引数なしコンストラクタが呼ばれました
x = 10, y = 20, z = 1
-----
x = 100, y = 200, z = 300
続行するには何かキーを押してください . . .
```

自分で値を設定していないフィールドはきちんと1になっていますね。

[\[C# Index\]](#) [\[総合Index\]](#) [\[Previous Chapter\]](#) [\[Next Chapter\]](#)

Update 09/Sep/2006 By Y.Kumei

当ホーム・ページの一部または全部を無断で複写、複製、転載あるいはコンピュータ等のファイルに保存することを禁じます。