

## 第24章 コンストラクタとデストラクタ



インスタンスが生成される時に、自動的に呼び出されて初期化処理を行う一種のメソッドが「コンストラクタ」です。明示的にプログラマがコンストラクタを書かなくても自動的にデフォルトのコンストラクタが呼び出されています。

コンストラクタの作り方は簡単です。クラス名と同じメソッドで、戻り値の型を記載しません。また、コンストラクタは、他のクラスから呼ばれるのでpublicとなります。

### クラス名と同じ名前 (有れば引数) {}

引数は、有っても無くてもかまいません。引数が異なれば、複数のコンストラクタを書いてもかまいません。

インスタンスを生成するとき、

```
MyClass mc = new MyClass();
```

のように書いたと思いますが、この場合は引数なしのコンストラクタが呼び出されます。プログラマがコンストラクタを書いていなければ、デフォルトのコンストラクタが呼び出されています。引数を持つコンストラクタを呼び出したければ、

```
MyClass mc = new MyClass(3);
```

のように書きます。もちろん、MyClassクラスには、プログラマが引数付きのコンストラクタを書いていなければエラーとなります。(従ってデフォルトのコンストラクタは、引数なしのコンストラクタということになります。また、デフォルトのコンストラクタは何をやっているかというと、フィールドを初期化しているのです。(前章を参照))

では、サンプルを見てみましょう。

```
// constructor01

using System;

class MyClass
{
    int x;
    int y;

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

    public MyClass(int a, int b)
    {
        x = a;
        y = b;
    }
    public MyClass()
    {
        x = 1;
        y = 1;
    }
}

class constructor01
{
    public static void Main()
    {
        MyClass mc1 = new MyClass(3, 4);
        mc1.Show();

        MyClass mc2 = new MyClass();
        mc2.Show();
    }
}
```

MyClassクラスには、xとyのprivateなインスタンスメンバがあります。

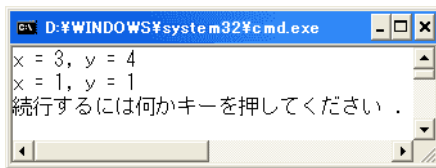
Showメソッドは、xやyの値を表示します。

また、int型の引数を2つつコンストラクタと、引数なしのコンストラクタがあります。引数付きのコンストラクタは、引数の値をxやyに代入しています。引数なしのコンストラクタはxやyの値を1に設定しています。

Mainメソッドでは、最初にnew MyClass(3, 4);というように、書いています。こうすると、引数付きのコンストラクタが呼ばれ、xに3、yに4が設定されます。

次に、new MyClass();としてインスタンスを生成しています。こうすることにより、引数なしのコンストラクタが呼ばれるはずですが。

では、実行結果を見てみましょう。



new MyClass(3, 4)でインスタンスを生成したときは、xが3, yが4になっています。

new MyClass()でインスタンスを生成したときは、xもyも1で初期化されていますね。

さて、C#では、不要になったオブジェクトは、自動的にメモリ上から解放されます。C/C++では、プログラマが自分で解放しなくてはなりませんでした。(うっかり忘れると、メモリーリークとなりユーザーから怒られます。)

C#では、自動的に不要なオブジェクトを解放することを「ガベージコレクション」といいます。メモリ上に散らばっている不要物(ゴミ=ガベージ)を集めるという意味でしょうか。

この、ガベージコレクションが起る直前に、デストラクタという一種のメソッドが呼び出されます。しかし、ガベージコレクションがいつ起るのかは予測不能で、デストラクタを自分で書いても、それがいつ実行されるのかわからないのでは、困りますね。C++では、デストラクタには、メモリの解放関係のコードを書きます。しかし、C#では自動でメモリが解放されるので、わざわざデストラクタを自分で書く必要性はほとんどないかもしれません。

一応デストラクタは、

**~クラス名 () { ... }**

のように書きます。また、デストラクタは引数をとりません。publicも不要です。

// destructor01.cs

using System;

class MyClass

```
{
    public int x;

    ~MyClass()
    {
        Console.WriteLine("デストラクタが呼ばれました ({0})", x);
    }
    public MyClass(int a)
    {
        x = a;
    }
}
```

class destructor01

```
{
    public static void Main()
    {
        MyClass[] mc = new MyClass[10];

        for (int i = 0; i < 10; i++)
        {
            mc[i] = new MyClass(i);
            Console.WriteLine("インスタンスを生成しましたmc[{0}]", i);
        }

    }
}
```

MyClassクラスには、インスタンスフィールドxがあります。

デストラクタ(~MyClass)が呼ばれると、その時のxの値を表示します。

コンストラクタは、呼ばれるときの引数でxを初期化しています。

さて、Mainメソッドを見てください。

**MyClass[] mc = new MyClass[10];**

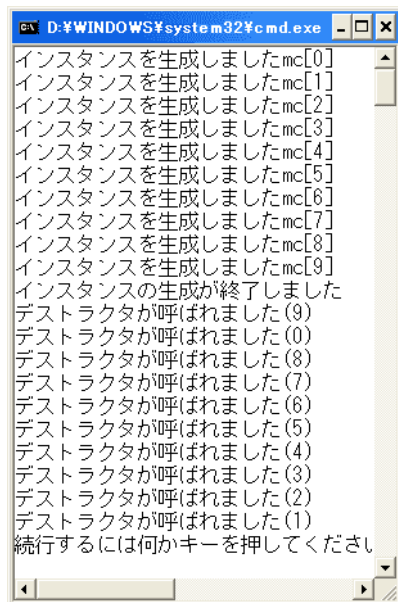
は、ちょっと見慣れない形ですね。でも

**int[] arr = new int[10];**

ならどうでしょうか。これと同じです。参照変数の配列を10個用意しているのですね。

mc[0]からmc[9]に、MyClassオブジェクトの参照を代入しています。それぞれのインスタンスはフィールドxの値が0, 1, ...9です。

これで、プログラムは終了です。プログラム終了時にオブジェクトのメモリを解放するはずなのでデストラクタも呼ばれるはずですが、



```
D:\WINDOWS\system32\cmd.exe
インスタンスを生成しましたmc[0]
インスタンスを生成しましたmc[1]
インスタンスを生成しましたmc[2]
インスタンスを生成しましたmc[3]
インスタンスを生成しましたmc[4]
インスタンスを生成しましたmc[5]
インスタンスを生成しましたmc[6]
インスタンスを生成しましたmc[7]
インスタンスを生成しましたmc[8]
インスタンスを生成しましたmc[9]
インスタンスの生成が終了しました
デストラクタが呼ばれました(9)
デストラクタが呼ばれました(8)
デストラクタが呼ばれました(7)
デストラクタが呼ばれました(6)
デストラクタが呼ばれました(5)
デストラクタが呼ばれました(4)
デストラクタが呼ばれました(3)
デストラクタが呼ばれました(2)
デストラクタが呼ばれました(1)
続行するには何かキーを押してください
```

必ずしも、オブジェクトが生成された順番にデストラクタが呼ばれるというものでもありません。

---

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

Update 30/Aug/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。