.NET Framework ネームスペースの旅

すべての根源よりすべてのクラスに向けて

Episode 2

System.Collections ネームスペース編

グレープシティ株式会社
アドバイザリースタッフ
矢沢 久雄 *YAZAWA, Hisao*

はじめに

みなさんは、オブジェクト指向プログラミングがお好きですか？　覚えるべきテクニックが多過ぎて、はっきりいって面倒くさいですよね。私も、かつてはオブジェクト指向プログラミングが嫌いでした。ところが、ある程度コツがわかると「おお！　これは便利じゃないか」と思えるようになりました。今では、オブジェクト指向プログラミングに対して、好き嫌いではなく「便利だ」というだけの印象をもっています。前回の連載で説明した“継承”“オーバーライド”“オーバーロード”などのテクニックも、難しいとか面倒くさいというものではなく、使えばそれなりに便利なものなのです。難しく考えずに、気楽にゆきましょう。

今回は、「System.Collections」ネームスペースを旅します。このネームスペースの中にあるクラスを使いこなすには、“コレクション”および“インターフェイス”というテクニックを覚えなければなりません。それらがなぜ便利なのかがわかったなら、みなさんのプログラミングで大いに活用してください。

おっと、いけない。この連載のテーマは、オブジェクト指向プログラミングじゃなかったですね。でもいいでしょう。.NETでプログラミングするには、Visual C#（以下C#）とVisual Basic .NET（以下VB.NET）のどちらを使っても、オブジェクト指向プログラミングを避けて通れないのですから。

本稿で前提となるもの

OS	Windows XP（SP2）
開発環境	Visual Studio.NET .NET Framework 1.0.3705.288（SP2）

初級　中級　上級

この記事で解説しているサンプルプログラムは、付録CD-ROMの¥DMAG¥NAMESPACEフォルダ以下に収録しています。

¥CSOBJARRAY	：異なるオブジェクトをひとつのコレクションに格納する（C#版）
¥VBOBJARRAY	：異なるオブジェクトをひとつのコレクションに格納する（VB.NET版）
¥CSINTERFACE	：インターフェイスの定義と実装したクラス（C#版）
¥VBINTERFACE	：インターフェイスの定義と実装したクラス（VB.NET版）
¥CSARRAYLIST	：ArrayListクラスの使い方（C#版）
¥VBARRAYLIST	：ArrayListクラスの使い方（VB.NET版）
¥CSSTACK	：Stackクラスの使い方（C#版）
¥VBSTACK	：Stackクラスの使い方（VB.NET版）
¥CSQUEUE	：Queueクラスの使い方（C#版）
¥VBQUEUE	：Queueクラスの使い方（VB.NET版）
¥CSCOLLECTION	：オリジナルのコレクションを使う（C#版）
¥VBCOLLECTION	：オリジナルのコレクションを使う（VB.NET版）

コレクションとは何か？

まず「コレクション」が何であるかを説明しましょう。コレクションとは、オブジェクトのメモリアドレスの配列のことです。クラスがメモリにロードされたものをオブジェクト（またはクラスのインスタンス）と呼びます。プログラムの中に複数のオブジェクトがあるなら、それらのメモリアドレス（メモリ上の位置を表わす数値）を配列にまとめて整理したものがコレクションです。異なる種類のクラスのオブジェクトであっても、メモリアドレスの値は32ビット整数値（32ビット環境の場合）で同じなので、ひとつのコレクションにまとめることができます（図1）。

C#ではobject型の配列、VB.NETではObject型の配列で、メモリアドレスの配列すなわちコレクションを作成できます。これらのデータ型はSystem.Objectクラスの別名です。整数型や浮動小数点数型などの通常の（オブジェクトではない）データもコレクションに格納できます。ボクシングによって、自動的にSystem.Int32クラスのインスタンスやSystem.Doubleクラスのインスタンスに変換されるからです。C#やVB.NETのプログラミング上の約束事として、ベースクラスをデータ型とした変数には、サブクラスのインスタンスを代入できるようになっています。System.Objectクラスは、あらゆるクラスのベースクラスです。したがって、System.Objectクラスをデータ型とした変数には、あらゆるクラスのインスタンスを代入できます。

リスト1は、さまざまなデータ型の変数をコレクションに格納するコンソールアプリケーションの一部です。コレクションに格納された時点で、変数のデータ型は、System.Int32クラスやSystem.Doubleクラスにボクシングされ、それがSystem.Objectクラスの配列に代入されます。コレクションからデータを取り出して利用する場合には、「キャスト（型変換）」を行ないます。C#では、

```
i = (int)obj;
```

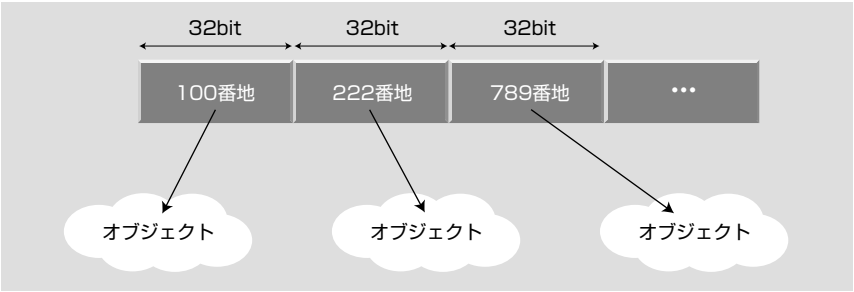
のように、変数名の前にカッコで囲んでデータ型やクラス名を指定することでキャストを行ないます。VB.NETでは、

```
i = CType(obj, Integer)
```

のようにCType関数を使ってキャストを行ないます。キャストが必要なのは、コレクションから取り出した時点では、オブジェクトのデータ型がSystem.Objectクラスになっているからです。

コレクションは、さまざまなデータ型の変数や、さまざまな種類のクラスのインスタンスを一元管理したい場合に便利です。もちろん、同じデータ型の変数や、同じ種類のクラスのインスタンスを一元管理することもできます。ただし、みなさんがSystem.Objectクラスの配列を使うことは滅多にないでしょう。なぜなら、System.Collectionsネームスペースに、コレクションを実現するクラス群があるからです。これらのクラスは、内部的にSystem.Objectクラスの配列をもっているだけでなく、配列のサイズを動的に変更したり、配列の要素をソートやサーチするなどの便利な機能をもっています。

図1：コレクションのイメージ



インターフェイスとは何か？

次に「インターフェイス」が何であるかを説明しましょう。インターフェイスとは、ひとつ以上のメソッド（関数）のプロトタイプをまとめて名前を付けたものです。メソッドのプロトタイプとは、メソッド名、戻り値のデータ型、引数の数とデータ型だけが定義され、メソッドの処理内容が記述されていないもののことです。たとえば、MyFuncAおよびMyFuncBという2つのメソッドをまとめたIMyInterfaceという名前のインターフェイスは、リスト2のように定義されます。「interface