クラスプログラミングの応用
ポリモーフィズムの理解

株式会社CSK
教育サービス事業部
大井 卓爾
OOI, Takuji

Technology Tools

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ❑ Other:

Level

★★☆☆☆

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥F01_02ディレクトリに収録しています。

¥ORDERWINAPP2_1
クラスのインスタンス化（3パターン）
¥ORDERWINAPP2_2
コンストラクタを使ったサンプル
¥ORDERWINAPP2_3
引数のある場合とない場合のサンプル
¥ORDERWINAPP2_4
ガベージコレクションのサンプル
¥ORDERWINAPP2_5
共有変数／共有メソッドサンプル
¥ORDERWINAPP2_6
継承サンプル
¥ORDERWINAPP2_7
オーバーライドサンプル
¥ORDERWINAPP2_8
ポリモーフィズムサンプル

はじめに

本稿では「クラスプログラミングの基本」を踏まえて、クラスの応用的な使い方を解説します。

とくに、オブジェクト指向において重要な要素である“ポリモーフィズム”について解説します。これを理解すると、オブジェクト指向のメリットがはっきり理解できます。

しかし、ポリモーフィズムを理解するには、いくつかの前提となる知識が必要になります。前半はその前提となるクラスの使い方や動作などについて解説し、後半はポリモーフィズムそのものについて解説したいと思います。

オブジェクトとメモリの関係

オブジェクトを作成した際の、メモリがどのように使われるかを知ることは重要です。まずは、いくつかのサンプルコードを紹介して、メモリがどう使われているかを確認してみましょう。

Orderクラスと用語補足

たとえば、リスト1のようなOrderクラスがあるとします。Orderクラスには、Noという名前で変数が宣言されています。Noのようにクラス内でメソッドの外側に宣言された変数のことを「メンバ変数」と呼びます。また、メソッド内に宣言された変数は「ローカル変数」と呼びます。これらの用語は、この後の解説でも多く出てきますので、今のうちに整理しておいてください。

■Orderクラスをインスタンス化（その1）

Orderクラスをインスタンス化するには、以下のようなコードを使います。

```
Dim objOrder As New Order
objOrder.No = 100
Msgbox(objOrder.No)
```

この場合は、図1のようなイメージで

リスト1：Orderクラス

```
Public Class Order
  ' メンバ変数（データ）
  Public No As String
End Class
```

メモリ上にオブジェクトが作成されます。

オブジェクト参照変数のobjOrderには、オブジェクトそのものが格納されるわけではなく、メモリ上のアドレスが格納されます。アドレスは「メモリ上の場所情報」で、アドレスを参照することにより、実際のオブジェクトの場所を特定しています。

実際のオブジェクトには、Noメンバ変数の値「100」が格納されます。実行結果は当然ですが“100”と表示されます。

■Orderクラスをインスタンス化（その2）

では、次に以下のコードを確認しましょう。

```
Dim objOrder As New Order
objOrder.No = 100
Dim objOrder2 As New Order
objOrder2.No = 200
Msgbox(objOrder.No & " " & objOrder2.No)
```

この場合は、図2のようなイメージでメモリ上にオブジェクトが作成されます。オブジェクトは、オブジェクトごとにメンバ変数やメソッドを持つイメージであるため、オブジェクトごとに領域がメモリ上に確保されます[注1]。そのため、objOrderのNoメンバ変数「100」と、objOrder2のNoメンバ変数「200」はそれぞれメモリ上の別の領域に値が格納されます。

したがって、実行結果は「100 200」と表示されます。

注1）メソッドはオブジェクトごとに動作しますが、コード（処理）は同じなので、メソッドのコードはオブジェクトごとにはメモリ上に作成されません。

図1：メモリ上に作られたOrderクラスのオブジェクト

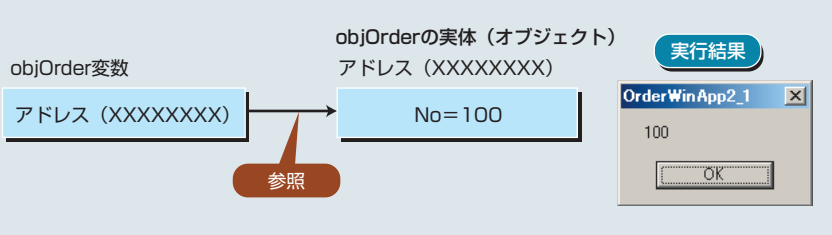
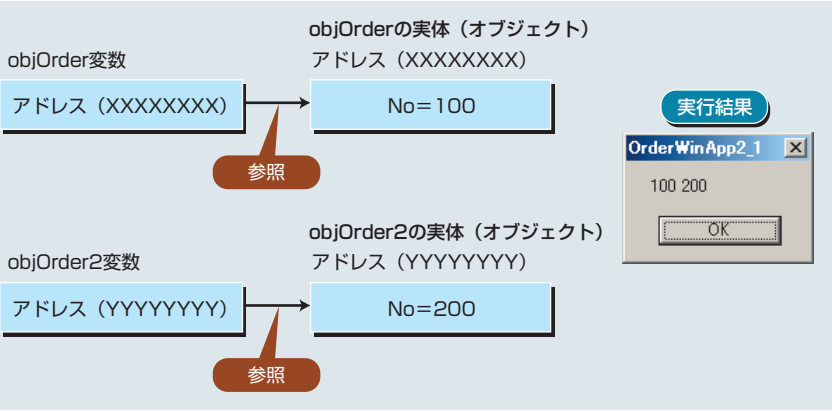


図2：メモリ上には2つのオブジェクト



■Orderクラスをインスタンス化（その3）

では、次に以下のコードを確認します。

```
Dim objOrder As New Order
objOrder.No = 100
Dim objOrder2 As New Order
objOrder2.No = 200
objOrder2 = objOrder
Msgbox(objOrder.No & " " & objOrder2.No)
```

「Orderクラスをインスタンス化（その2）」のサンプルコードと違う点は、5行目の、

```
objOrder2 = objOrder
```

です。

これはどのような動作になるのでしょうか？　この場合は、図3のようなイメージになります。objOrder2のオブジェクト参照変数に代入されるのは、objOrderのオブジェクト参照変数に格納されているアドレスなのです。したがって、objOrder2のオブジェクト参照変数は、objOrderのオブジェクトを参照することになります。

すなわち、どちらのオブジェクト参照変数も同じobjOrderのオブジェクトを参照する結果になるため、実行結果は「100 100」と表示されます。このことを認識していないと、誤ってバグを生んでしまう可能性もありますので、十分な注意が必要です。

なお、今回のようにクラスを元にオブジェクトを作成した場合は、メモリ確保の方法は「参照型」になりますが、たとえば「Integer型」で変数を作成した場合は「値型」となります。詳しくはMicrosoft Visual Studio .NET ドキュメントの「値型と参照型」をご覧ください。