クラスプログラミングの実践
関連技術と.NETへの応用

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_03ディレクトリに収録しています。

¥ORDERWINAPP3_1：
インターフェイスサンプル

¥ORDERWINAPP3_2：
抽象クラスサンプル

¥AUTHENTICATION：
「クラスライブラリ」プロジェクト

¥ORDERWINAPP：
「Windowsアプリケーション」プロジェクト

はじめに

前項では、クラスプログラミングの応用を解説しました。ポリモーフィズムの利便性がわかると、オブジェクト指向への理解がより深まります。本稿では、ポリモーフィズムに関連する技術と、.NETでの実装時のポイントなどを中心に解説をしてゆきます。

インターフェイス

ポリモーフィズムはインターフェイスを使用することで実現できます。

インターフェイスは、プロパティやメソッドの定義の集まりです。あくまでも定義だけなので、プロパティやメソッドの処理（コード）を実装することはできません。

インターフェイスは、クラスに実装して使用します。インターフェイスを実装するクラスでは、インターフェイスに定義されているプロパティやメソッドの処理を記述する必要があります(図1)。

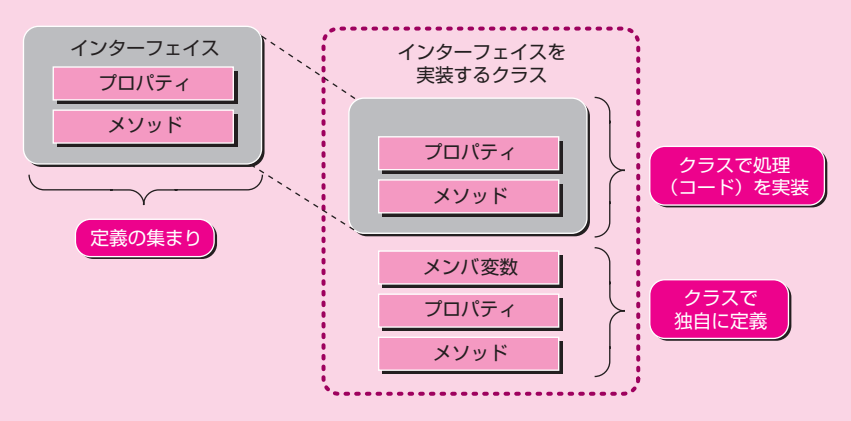
インターフェイスは「Interface～End Interface」内に定義を記述します。リスト1では、IOrderというインターフェイスを定義しており、ExecuteメソッドとCancelメソッドの定義を含めています。各メソッドにはそれぞれ、「No」というString型の引数を定義していますが、処理やEnd Subが記述されていないことを確認できます。

クラスでインターフェイスを実装するには、「Implements」キーワードを使用し、インターフェイス名を指定します。リスト2がインターフェイスを実装するクラスの例です。Orderクラスと VipOrderクラスでは、IOrderインターフェイスを実装しています。コードのポイントは、各クラスの2行目です。

```
Implements IOrder
```

このコードにより、IOrderインターフェイスを実装することができます。インターフェイスを実装したクラスでは、インターフェイスで定義されているプロパティやメソッドの処理を実装

図1：インターフェイス



する必要があります。OrderクラスやVipOrderクラス内のExecuteメソッドやCancelメソッドでは、Implementsキーワードが使用されており、インターフェイス内のどのメソッドの処理を実装しているのかを表わしています。全体のイメージは図2になりますので、確認しておいてください。

インターフェイスを使用する側のコードがリスト3になります。

まず、IOrderインターフェイスを使用して変数の宣言をしています。

```
Dim obj As IOrder
```

次にフォーム上の「VIP専用注文をする」のチェックボックスによって、どのクラスをインスタンス化するのかが決定されます。

```
If chkVip.Checked = False Then
  obj = New Order
Else
  obj = New VipOrder(txtVipName.Text)
End If
```

この各クラスのインスタンス化が、新しい内容になります。IOrderインターフェイスを使用して宣言された変数objに対して、インターフェイスを実装しているOrderクラスまたはVipOrderクラスのオブジェクトを格納しています。

つまり、特集記事「クラスプログラミングの応用」の“継承”を使用したポリモーフィズムと同じようなことが実現できるのです（図3）。「インターフェイスを実装するクラスは、インターフェイスの定義を含む」ということになるので（図1）、「インターフェイス型の変数から、インターフェイスを実装するクラスのオブジェクトを参照する」ことができるのです。

そして、

リスト1：インターフェイス（IOrderインターフェイス）

```
Public Interface IOrder
   ' メソッドの定義
   ' 定義のみで処理や「End Sub」は書かない
   Sub Execute(ByVal No As String)
   Sub Cancel(ByVal No As String)
End Interface
```

リスト2：インターフェイスを実装するクラス

```
' Orderクラス（インターフェイスを実装するクラス）
Public Class Order
   Implements IOrder  ' インターフェイスを実装
   ' メソッド
   Public Sub Execute(ByVal No As String) _
     Implements IOrder.Execute
       MsgBox(No & "の注文を実行しました。")
   End Sub
   ' メソッド
   Public Sub Cancel(ByVal No As String) _
     Implements IOrder.Cancel
       MsgBox(No & "の注文を取消しました。")
   End Sub
End Class

' VipOrderクラス（インターフェイスを実装するクラス）
Public Class VipOrder
   Implements IOrder  ' インターフェイスを実装
   ' 変数
   Private VipName As String ' VipOrderクラスの独自の定義
   ' コンストラクタ
   Public Sub New(ByVal sVipName As String)
      VipName = sVipName
   End Sub
   ' メソッド
   Public Sub Execute(ByVal No As String) _
     Implements IOrder.Execute
       MsgBox(No & VipName & "様の、VIP注文を実行しました。")
   End Sub
   ' メソッド
   Public Sub Cancel(ByVal No As String) _
     Implements IOrder.Cancel
       MsgBox(No & "の注文を取消しました。")
   End Sub
End Class
```