Visual Basic 2008程式語言 物件導向程式設計

講師：林賢達
Peter.lin@imestech.com

課程大綱

- 物件導向基礎概念
- 類別(Class)
- 類別成員(Class Member)
- 封裝
- 繼承(Inherits)

物件導向基礎概念

- 什麼是物件導向(Object Oriented)?
  - 物件導向是將問題領域中涉及的人、事、物予以物件化，透過這些物件之間的訊息傳遞，相互溝通協調，執行某項工作。
- Windows檔案管理：System.IO命名空間
  - Directory類別：建立、移動目錄和列舉子目錄
  - File類別：建立、複製、刪除、移動和開啟檔案
  - StreamReader和StreamWriter類別：以特定的編碼方式讀取檔案內容

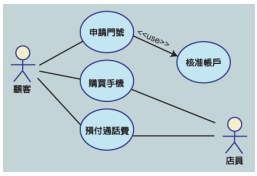
物件導向基礎概念

- 物件導向程式設計(OOP)是以物件導向分析設計(OOAD)產出的圖表為基礎
  - UML(統一建模語言) + 工具：IBM Rational Rose，Visio，SmartDraw，DynamicDraw…
  - 系統分析：功能需求 → 系統規格
    系統設計：系統規格 → 設計圖表
    程式開發：設計圖表 → 定義類別

4

物件導向基礎概念

- 使用案例(Use Case)
  - 使用案例：規劃系統功能的範圍，做什麼?不做什麼



  - 角色：參與系統運作的使用者或外部系統

5

物件導向基礎概念

- 使用案例描述
  - 申請門號描述：(1)**顧客**填寫**申請表**：**身分證號碼**、**姓名**、**出生年月日**、**住址**、**手機號碼**、**申請日期**等。(2)**店員**查核身分證明文件…
- 從案例描述中找出候選的類別
  - 名詞：物件(顧客、店員、出生年月日)或狀態(姓名、住址等)…
  - 動詞：行為，例如填寫申請表、查核身分…

6

類別

- 類別是一組具有相同資料結構和操作的物件的集合。類別對這些具有相同性質的物件的抽象，描述物件的共同特徵。
- 類別定義了一個物件的"狀態"和"行為"
  - 使用類別的私有欄位來描述與保存物件的內部狀態。
  - 使用類別中的方法去定義物件的行為。
- 類別定義了一個範本，使用此範本可以建立一個或多個物件

7

宣告類別

- 使用Class關鍵字宣告類別

```
存取修飾詞 Class 類別名稱
        Inherits Object    '存取限定子、類別名稱、父類別

        '類別成員

End Class          '類別主體 Class … End Class
```

- 類別成員
  - 資料成員：常數，欄位和屬性
  - 方法成員：方法、事件，建構函式和解構函式

8

實體化(Instatiation)

- 使用類別來建立物件的過程
  - 使用New關鍵字

```
Dim MyCar As New Car()
```

  - 沒有可存取的建構子：使用工廠模式建立物件

```
Dim fs As FileStream = File.Create("C:\123.txt")
```

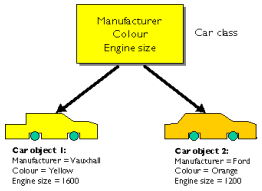
  - 使用反射(Reflection)

```
Dim t As Type = Type.GetType("Car")
…
```

9

類別 VS 物件

- 類別是在設計階段時完成，物件是類別在執行階段的實體，佔用記憶體空間且具有生命週期



類別是物件範本

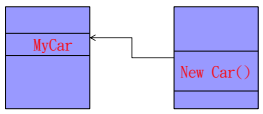
物件是類別的執行實體具有各自的狀態

10

物件 VS 物件變數(物件參考)

- 物件變數儲存的是物件的參考位址，而非物件本身

```
Dim MyCar As Car        '物件變數
MyCar = New Car()       '物件
```



11

類別成員

- 常數(Constant)：代表與類別相關聯的常數值
- 欄位(Field)：類別中的變數
- 方法(Method)：負責執行類別中的計算(處理)
- 屬性(Property)：對類別中欄位值的存取邏輯
- 事件(Event)：用於說明發生了什麼事情
- 建構函式(Constructor)和解構函式(Destructor)：分別用於對類別進行實體化和摧毀物件

12

封裝

- 把屬性和方法封裝在類別中，然後使用類別建立物件(類別的執行實體)，接著使用物件存取屬性和叫用方法。
- 封裝的好處
  - 良好的封裝可以增加內聚性，降低耦合性
  - 類別內部的實現可以自由修改
  - 類別具有清晰的對外開放介面

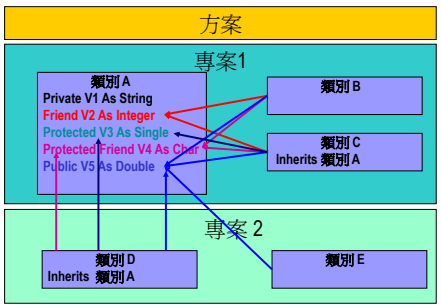
13

封裝

- 使用**存取修飾詞**來達到封裝的目的
  - Private：只允許在類別中存取之
  - Protected：只允許在類別或子類別中存取之
  - Friend：只允許在相同專案中存取之
  - Protected Friend：只允許在類別、子類別或相同專案中存取之
  - Public：任何地方均可存取之

14

封裝(續)



15

屬性

- 屬性提供了對物件內部狀態的存取。
- 屬性實現了物件的封裝性：不直接操作類別的資料內容，而是透過存取器來進行存取。
- 特性
  - 由Get/Set存取器來控制對內部狀態的存取，當取值時會叫用Get函式；賦值時則會叫用Set函式，可以在Get/Set函式提供存取邏輯
  - 內部是函式(定義)，外部則像個欄位(使用上)

16

屬性

- 使用Property程序的Get/Set函式存取私有欄位

```
存取修飾詞 Property 屬性名稱() As 資料類型
    Get                                 '取值
        Return 私有欄位
    End Get

    Set (value As 資料類型)             '賦值
        If Value<0 Then Value =0
        私有欄位 = value
    End Set

End Property
```

17

屬性

- 使用ReadOnly定義唯讀屬性

```
存取修飾詞 ReadOnly Property 屬性名稱() As 資料類型
    Get
        Return 私有欄位
    End Get
End Property
```

- 使用WriteOnly定義唯寫屬性

```
存取修飾詞 WriteOnly Property 屬性名稱() As 資料類型
    Set (value As 資料類型)
        私有欄位=value
    End Set
End Property
```

18

索引器Indexer

- 索引器是一個特殊的屬性，它提供了使用索引方式方便地存取類別的集合資料的方法。
- 索引器通常在類別中表示元素集合

```
Public Class Sale
    Private customers(10) As String

End Class
```

19

索引器

- 索引器的定義類似屬性
  - Get 存取器傳回值。Set 函式分配值。
  - value 關鍵字用於定義由Set函式分配的值。

```
Public Property Customer(Index As Integer) As String
    Get
        Return customers(Index)
    End Get

    Set (ByVal Value As String)
        Customers(Index) = Value
    End Set
End Property
```

20

索引器

- 使用索引器可以像操作陣列的方式來存取物件中的集合

```
Dim Peter As New Sale()

Peter(0) = "A客戶"
Peter(1) = "B客戶"
Peter(2) = "C客戶"

…
```

21

方法

- 方法可用於定義物件的行為
- 副程式

```
[存取修飾詞] Sub 副程式名稱(參數宣告)
    …
End Sub
```

- 函式

```
[存取修飾詞] Function 函式名稱(參數宣告) As 資料類型
    …
End Function
```

22

建構函式

- 類別的特殊方法，主要用於初始化實體的資料成員。當使用New關鍵字實體化物件時，由.NET CLR自動叫用
- 特性
  - 使用函式名稱必須是New，預設為無參數
  - 不可有傳回值
  - 個數不定(使用重載Overload)，可以有一個以上的建構函式

23

建構函式

- 定義建構函式。即使沒有定義它，編譯器也會自動為類別提供一個預設的建構函式

```
Public Class 類別名稱
    Public Sub New()
        MyBase.New()
    End Sub

End Class
```

- 當實體化物件時，由.NET CLR叫用

```
Dim 物件變數 As New 類別名稱()
```

24

建構函式

- 重載建構函式
  - 不可使用Overloads關鍵字

```
Public Sub New (FirstName As String, LastName As String )
        MyBase.New( FirstName, LastName)
        …
End Sub
```

- 建構函式之間的叫用

```
Me.New(…)
```

25

解構函式

- 使用Finalize函式，加上Overrides (覆寫)
  - 用於釋放資源
  - 不可有參數和傳回值
  - 當物件被Garbage Collection回收時，會自動被叫用

```
Protected Overrides Sub Finalize()
    '釋放資源
    MyBase.Finalize()
    …
End Sub
```

26

使用特性(Attribute)

- 使用角括弧(<>)提供特殊的資料
- 可用於類別、方法和屬性
- 例如標題、版本、Web服務、組件、安全性和自訂設定…等

```
<WebMethod()> _
Public Sub HelloWorld()
        …
End Sub
```

27

建立物件

- 宣告後初始化(使用New關鍵字)

```
Dim C As  Customer
C = New Customer()
```

- 宣告同時初始化，使用預設建構函式

```
Dim C As Customer = New Customer()
Dim C As New Customer()
```

- 宣告同時初始化，使用重載建構函式

```
Dim C As Customer = New Customer(1)
Dim C As New Customer(1)
```

28

摧毀物件

- GC (資源回收器，Garbage Collection)
  - 當記憶體不夠時，GC將回收不再被使用物件佔用的資源
  - 使用 *物件變數* = ***Nothing*** 標示不再被使用
  - 使用GC.Collect()強制回收資源

29

摧毀物件

- 定義Dispose方法，用於自行釋放資源

```
Public Sub Overridable Dispose()
    '關閉資料庫連線
    …
End Sub
```

- 在Client端叫用Dispose方法

```
Dim C As New Customer(1)
…
C.Dispose()
```

30

共用(Shared)成員

- 可建立全域變數、全域方法或常數(隱含的靜態成員)
- 共用成員不屬於任何物件，可以使用類別名稱直接存取

```
Dim Radius As Decimal = 10

Dim Area As Decimal = Math.PI * Math.Pow(Radius, 2)
```

- 若將類別中的某個成員宣告為Shared，該成員稱為共用成員。

31

共用成員

- 共用成員 VS 實體成員
  - 共用成員直接由類別存取；實體成員由物件存取。
  - 共用成員是屬於類別：實體成員則屬於類別的實體—物件。
- 共用方法為類別所有，因此不用建立類別的實體就可以叫用。例如System.Math類別。

32

共用資料成員

- 允許一個類別的多個實體共用相同的欄位
- 使用Shared關鍵字

```
存取修飾詞 Class 類別名稱
    Public Shared 欄位名稱 As 資料類型
    …

End Class
```

- 直接使用**類別名稱**來存取Shared欄位

```
類別名稱.欄位名稱        '無需建立物件
```

33

共用方法成員

- 使用Shared關鍵字

```
存取修飾詞 Class 類別名稱
    Public Shared Sub 方法名稱(參數宣告)
        …
    End Sub
End Class
```

- 使用**類別名稱**來存取Shared方法成員

```
類別名稱.方法名稱(引數)          '無需建立物件
```

- 在類別中只能存取Shared欄位

34

命名空間(Namespace)

- 使用命名空間來組織類別(邏輯的儲存位置)
  - 增強可讀性，例如System.IO
  - 降低類別名稱的衝突

```
NameSpace 命名空間名稱1
    存取修飾詞 Class 類別名稱
        …
    End Class
End NameSpace
```

```
命名空間名稱1.類別名稱
命名空間名稱2.類別名稱
```

- 類別實際的儲存位置是組件(dll)，與命名空間無直接關係

35

命名空間

- 完全引用
  - 從命名空間的最外層開始引用

```
Dim c As NS1.Class1
```

- 匯入命名空間
  - 在程式碼檔上方使用**Imports**關鍵字匯入要使用的命名空間

```
Imports NS1
Imports Other = NS2      '命名空間的別名(Alias)
Dim c1 As Class1
Dim c2 As Other.Class1
```

36

繼承

- 繼承是使用已存在的類別為基礎建立新類別的技術
- 子類別從父類別獲取其成員(程式碼和資料)，而且可以定義自己的成員
- VB.NET 只支援單一繼承，子類別有且僅有一個父類別，也就是繼承關係是一對一
- 使用繼承我們可以衍生出無數個子類別，子類別還可以建立自己的子類別

37

繼承

- 為什麼要使用繼承?
  - 原始碼重用，減少撰寫相同的程式碼
  - 設計重用，在父類別定義公開的成員，然後由其衍生新的子類別，子類別可以有自己的成員
- 如果沒有明確指定繼承哪一個父類別，則VB.NET預設繼承自Object
- 通常用於表示"is a"的關係，如果A類別是從B類別中繼承而來，即A是B的子類別，則我們可以說："class A is-a class B"

38

繼承

- 使用Inherits宣告類別子類別繼承自父類別

```
存取修飾詞 Class 子類別名稱                          '語法 1
    Inherits 父類別名稱
End Class

存取修飾詞 Class 子類別名稱 : Inherits 父類別名稱  '語法 2

End Class
```

39

繼承

- Overridable方法，宣告允許被子類別覆寫的方法，使用Override實作覆寫
- MustOverride方法，只能在抽象類別中使用
- 使用MustInherit宣告的類別必須被繼承，不能被實體化，又稱為抽象類別
- 使用NotInheritable宣告的類別不能被繼承(防止繼承)

40

繼承：*Demo NumbericTextBox*

- 只允許輸入數字的文字方塊
- 步驟
  - 繼承自System.Windows.Forms.TextBox
  - 覆寫OnKeyPress方法，檢查輸入字元

```
If (Not Char.IsDigit(e.KeyChar)) Then
        e.Handled = True
End If
```

41

使用MyBase

- 在子類別中使用父類別的方法成員
- 可以叫用Public、Protected和Friend方法
- MyBase不是一個真正的物件

```
存取修飾詞 Class 子類別名稱
    Inherits 父類別名稱
    Public Overrides Function 方法名稱 (參數宣告)
        ...
        MyBase.方法名稱(參數)
    End Function
End Class
```

42

遮蔽(Shadows)

- 隱藏父類別的方法

```
Class Parent
    Public Sub Say(word As String)          '預設NotOverridable
        …
    End Sub
End Class
```

```
Class Child
        Inherits Parent
        Public Shadows Sub Say()        'Clienr端只能叫用此方法
            …
        End Sub
End Class
```

43

遮蔽

- 遮蔽測試

```
Dim MySon As New Child()

MySon.Say("hello")      '拋出例外
MySon.Say()             '正常
```

44

課程大綱

- 多形(Polymorphism)
  - 重載(Overload)
  - 覆寫(Overrides)
- 介面(Interface)
- 委派(Delegate)
- 事件(Event)

45

多形

- 同一個操作作用在不同的類別實體(物件)，不同的類別有不同的解釋，產生不同的執行結果
- 多個類別具有同名方法，但有不同的操作

樂器可以演奏歌曲，不同的樂器有不同的演奏方式，例如：

鋼琴：用手彈鍵盤
提琴：用手拉琴弓
長笛：用嘴吹口管

46

多形(Polymorphism)

- 實作方式
  - 類別：使用**重載**(Overload)來實現多形。在編譯時，根據參數類型和傳回類型等資訊來決定執行哪一個操作
  - 繼承：透過子類別**覆寫**(Overrides)父類別的Overridable方法來實作多形。在執行時，才會根據實際情況來決定執行哪一個操作
  - 介面：類別實作在介面所定義的成員。一個類別可實作多個介面

47

重載(Overload)

- 在一個類別中的方法可以有多個不同的版本：有相同的函式名稱，但參數宣告(個數、順序和資料類型)必須不同

```
Public Function GetOrderItem() As OrderItem
        Return New OrderItem()
End Function
```

```
Public Function GetOrderItem(ItemID As Integer) As OrderItem
        Return Me.OrderItems(ItemID)
End Function
```

- 使用參數類型來確認叫用哪一個方法
- 若重載繼承而來的方法需使用Overrides

48

覆寫(Overrides)

- 子類別為了滿足個別需求重新定義某個方法的不同實作，也就是在子類別中覆寫從父類別繼承而來的方法
- 使用Overrides關鍵字來覆寫父類別的Overridable方法
  - 方法名稱，參數宣告和傳回類型都必須相同
  - 執行時期將確定叫用物件是什麼類別的實體，並叫用適當的Overridable方法

49

覆寫(Overrides)

- 父類別可使用下列繼承修飾詞宣告方法
  - Overridable：此方法允許被子類別覆寫，例如Object的Finalize方法
  - MustOverride：此方法必須被子類別覆寫
  - NotOverridable：此方法不可被子類別覆寫，此為預設值
- 子類別：使用Overrides覆寫父類別的Overridable方法

50

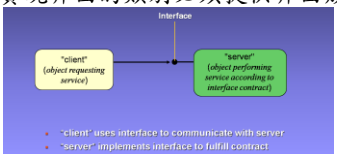
Override VS Overload

	Override	Overload
位置	存在於繼承關係的類別	同一個類別
方法名稱	相同	相同
參數宣告	相同	必須不同
傳回類型	相同	可以不同

51

介面

- 介面為類別提供了藍圖(blueprint)，只提供定義，但不含實作部分
- 實現介面的類別必須提供介面成員的實作



- 介面本身可以從多個基底(父)介面衍生而來

52

介面

- 定義方法、屬性和事件的簽章(Signature)
- 使用Interface關鍵字

```
Interface 介面名稱

    Function 方法名稱1(參數宣告) As 資料類型
    Sub方法名稱2()
    Sub方法名稱2(參數宣告)

End Interface
```

- 介面也可以繼承，使用Inherits

53

委派

- 委派是函式的封裝，代表某一類函式。這些函式都具有相同的簽章：相同的參數類型和傳回值類型。
- 委派的實體則代表一個具體的函式。
- 委派類型指定它所代表的方法之傳回類型和參數宣告，也就是它可以代表具有相同的參數列和傳回類型的任何一個方法

54

委派

- 為什麼要使用委派？提供更靈活的方法叫用
  - 用於非同步回呼(Asynchronous Callback)：由於產生委派實體是一個物件，所以可以將其作為參數來傳遞，也可以將其賦值給屬性。此時非同步方法便可以將一個委派作為參數，並且以後可以叫用該委派。
  - 多執行緒程式設計中使用委派來指定啟動一個執行緒時叫用的方法
  - 提供事件處理機制，使用委派指定某個事件的處理函式

55

委派

- 委派的宣告方式與方法的宣告方式相似
  - 使用Delegate宣告一個委派物件，宣告時需指定委派所代表函式的參數宣和傳回值類型

```
存取修飾詞 Delegate Function 委派類型名稱(參數宣告) As 資料類型
```

  - 委派類型均繼承自Delegate或MulticaseDelegate
- 代表一組具有相同參數宣告與傳回值類型的函式

```
Public Function 方法名稱(參數宣告) As 資料類型
    …
End Function
```

56

委派

- 使用委派：先宣告委派，再產生實體並指定所代表具體函式
  - 使用New關鍵字建立委派的實體，同時在括號內指定一個具體的函式

```
Dim 委派物件As 委派類型
委派物件 = New 委派類型(AddressOf 函式名稱)
```

  - 使用委派實體，並傳入給委派所代表方法的參數來叫用其代表的函式

```
委派物件.Invoke (引數)
```

57

委派

- 使用Delegate叫用來排序陣列
  - SortAscending遞增排序： 2, 1, 3 ⇨ 1, 2, 3
  - SortDescending遞減排序： 2, 1, 3 ⇨ 3, 2, 1
- 宣告Delegate類別

```
Delegate Sub SortMethod (ByRef  IntArray() As Integer)
```

58

委派

- 定義SortAscending與SortDescending方法

```
Private Sub SortAscending (ByRef IntArray() As Integer)
      Array.Sort(IntArray)
End Sub
```

```
Private Sub SortDescending (ByRef IntArray() As Integer)
      Array.Sort(IntArray)
      Array.Reverse(IntArray)
End Sub
```

59

委派

- 建立委派實體並指定所代表的具體函式，然後叫用函式

```
Private Sub SortArray(ByVal Ascending As Boolean,  ByRef  IntArray()
    As Integer)
    Dim del As SortMethod
    If (Ascending = True) Then
       del = New SortMethod(AddressOf SortAscending)
    Else
       del = New SortMethod(AddressOf SortDescending)
    End If
    del.Invoke(IntArray)
End Sub
```

60

事件

- 使用委派來實作事件處理機制，稱為事件委派
- 事件委派的基本形式

```
Public Delegate Sub  EventHandler (object sender As object, e As EventArgs)
```

  - sender As object: 來源物件
  - e As EventArgs，System.EventArgs類別的實體或衍生類別實體，包含事件的相關訊息

61

事件

- .NET Framework定義了許多的事件委派

```
Public Delegate Sub KeyEventHandler(sender As object, e As KeyEventArgs)

Public Delegate Sub MouseEventHandler(sender As object, e As MouseEventArgs)
```

- 自訂事件委派

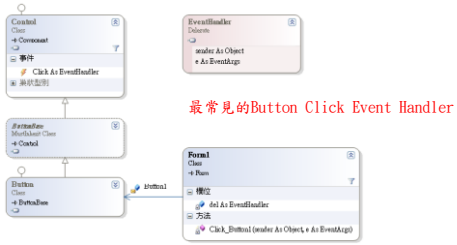
```
Public Delegate Sub NameEventHandler(source As object,  e As
    NameEventArgs)
```

  - 必須定義NameEventArgs類別

62

委派與事件

- .NET Framework的事件委派實例



最常見的Button Click Event Handler

63

事件

- 使用Event關鍵字宣告事件名稱，它的類型為事件委派

```
Public Event OnNameHandler As NameEventHandler
```

  - 或者

```
Public Event OnNameHandler (source As object, e As
    NameEventArgs)
```

- 使用RaiseEvent關鍵字觸發事件

```
RaiseEvent OnNameListEvent(Me, New NameListEventArgs())
```

64

事件

- 使用WithEvents宣告此物件帶有事件

```
Protected WithEvents MyList As New NameList()
```

- 事件處理函式，參數宣告必須與事件委派一致

```
Private Sub MyList_Added (ByVal Source As Object, ByVal Args
    As NameListEventArgs)
End Sub
```

- 建立事件與事件處理函式的關聯
  - 宣告方式或動態方式

65

事件

- 宣告方式

```
Private Sub MyList_Added (ByVal Source As Object, ByVal Args
    As NameListEventArgs) Handles MyList.OnNameHandler
End Sub
```

- 動態方式

```
Dim EventHandlerDelegate As New
    NameListEventHandler(AddressOf MyList_Added)
AddHandler MyList.OnNameHandler, EventHandlerDelegate
```

  - 或者

```
AddHandler MyList.OnNameListEvent, AddressOf MyList_Added
```

66

事件

- 使用RemoveHandler動態取消關聯

```
RemoveHandler MyList.OnNameListEvent, EventHandlerDelegate
```

- 或者

```
RemoveHandler  MyList.OnNameListEvent, AddressOf
    MyList_Added
```

67

Q & A