

第六章 物件與類別



6.1 物件與類別

6.5 類別繼承

6.2 建構函式

6.6 使用主控台程式建立視窗程式

6.3 靜態成員

6.4 物件陣列

備註：可依進度點選小節



■ 傳統結構化程式設計

⇒ 對現今軟體開發技術已不敷使用。

■ 目前最流行

⇒ 物件導向技術。

■ C# 提供完整的物件導向技術如：

類別繼承、多型、介面、運算子多載、泛型
(類似C++ 樣板)...等機制，

⇒ 讓使用 C# 的程式設計師能大幅度減少程式撰寫
及增加應用程式的效能。

■ 本章主要探討物件與類別間的關係。



6.1 物件與類別

6.1.1 何謂物件與類別

物件(Object)

簡單的說就是一個東西，物件可被識別和描述、有狀態(屬性)、有行為(方法)。

類別(Class)

是物件抽象的定義，也就是說類別是定義物件的一個藍圖。



例如

Peter(物件)是一個人(類別)。

Peter 物件是屬於人這種類別。

Peter的身高是164(身高的屬性)。

Peter會走路(行為, 方法)。

如用程式碼來表示 **Peter 物件**如下：

```
Person Peter = new Person(); //建立 Peter 物件屬於 Person 類別  
Peter.Height=164;           //設定 Peter 身高屬性為 164  
Peter.Walk(10);              //Peter 執行走路方法, 並設定走 10 公里
```



- 比較用 **TextBox** 類別建立物件名稱為 **txtId** 文字方塊的敘述：

```
TextBox txtId = new TextBox();//建立 txtId 物件屬於 TextBox 類別  
txtId.Height=20;           //設定 txtId 物件的高度屬性是 20  
txtId.Clear();              //執行 txtId 的 Clear()方法  
                             //將 txtId 文字方塊的內容清為空白
```

- 建立物件前必須先定義類別。
- 在 **.NET Framework** 中提供許多內建類別大幅增加應用程式開發的速度。



6.1.2 類別的定義

■ 類別儲存在一個*.CS類別檔。

■ 寫法：

```
public class 類別名稱 {  
    [成員存取修飾詞] 欄位  
    [成員存取修飾詞] 屬性  
    [成員存取修飾詞] 方法  
}
```

C# 資料成員常用的成員存取修飾詞有：private、public、protected。



存取修飾詞

① 宣告為 **private**

該成員即為私有型態，表示該成員只能在自身類別內存取；

② 若宣告為 **public**

該成員即為公用型態，表示該成員的存取沒有限制，允許其它類別也可存取；

③ 若宣告為 **protected**

該成員即為保護型態，表示該成員能在自身類別和被繼承的子類別內進行存取。



欄位

用來儲存物件的資料以表示物件擁有的狀態，可用一般的資料型別或其他的類別來宣告。

屬性

若物件欄位的值需受到限制或管理，可先將該欄位的值宣告為 **private**，讓使用者透過 **public** 的「存取子」來取得或設定 **private** 私有型別的欄位值，存取子是一種特殊的方法，它可定義類別的唯讀或唯寫的屬性。

方法

可使用方法(或稱函式)來定義，用來表示一個類別該擁有的行為。



- 例如定義 **Person** 類別擁有 **Height**(身高)及 **Weight**(體重)的欄位及一個 **Walk**(走路)方法。

```
public class Person           //定義 Person 類別
{
    public int Height;         //Height 身高欄位
    public int Weight;         //Weight 體重欄位
    public void Walk(int x )   //走路方法
    {
        Console.WriteLine("走了{0}公里", x);
    }
}
```



6.1.3 物件的宣告與建立

■ 定義類別後，接著用 **new** 關鍵字來建立該類別的物件實體。

■ 建立物件方法有兩種：

1. 方法1 先宣告，再建立物件

類別名稱 物件變數; // 宣告物件

物件變數 = new 類別名稱(); // 建立物件實體

2. 方法2 宣告物件同時建立物件

類別名稱 物件變數 = new 類別名稱();



■ 以6.1.2節 Person 類別為例，產生Person 類別的 Peter 物件實體。當建立物件時，該類別中的成員即屬於該物件所擁有，若要存取類別的 public 成員可用「.」運算子來達成。寫法：

```
Person Peter;           //宣告 Peter 物件為 Person 類別
Peter=new Person();     //建立 Peter 物件實體
Peter.Height=164;       //設定 Peter 身高為 164
Peter.Weight=65;        //設定 Peter 體重為 65
Peter.Walk(50);          //呼叫 Peter 的 Walk(50)方法
                        //結果會印出 "走了 50 公里" 訊息
Person Mary = new Person();//宣告並建立 Mary 物件，屬於 Person 類別
Mary.Height=172;        //設定 Mary 身高為 172
Mary.Weight=53;         //設定 Mary 體重為 53
Mary.Walk(20);           //呼叫 Mary 的 Walk(20)方法
                        //結果會印出 "走了 20 公里" 訊息
```



- 當建立不同物件時，每個物件都視為不同的執行個體，且每個物件都會擁有類別所定義的成員。

【結論】

物件才是類別的執行個體，
而類別是定義物件的藍圖。



範例演練

- 在 FirstClass.sln 專案新增 Product.cs 類別檔，Product 產品類別有 PartNO 編號, PartName 品名, Qty 數量三個欄位成員；以及定義 ShowInfo()方法用來顯示該項產品的編號, 品名, 數量等資訊。
最後在Program.cs 的 Main() 方法使用 Prouduct 產品類別建立兩個產品物件。

```
file:///C:/CSharp/chap06/FirstClass/...  
編號 : B001  
品名 : 變形金剛2  
數量 : 20  
=====  
編號 : P001  
品名 : 惠普HP iPAQ PDA行動手機  
數量 : -5  
=====
```



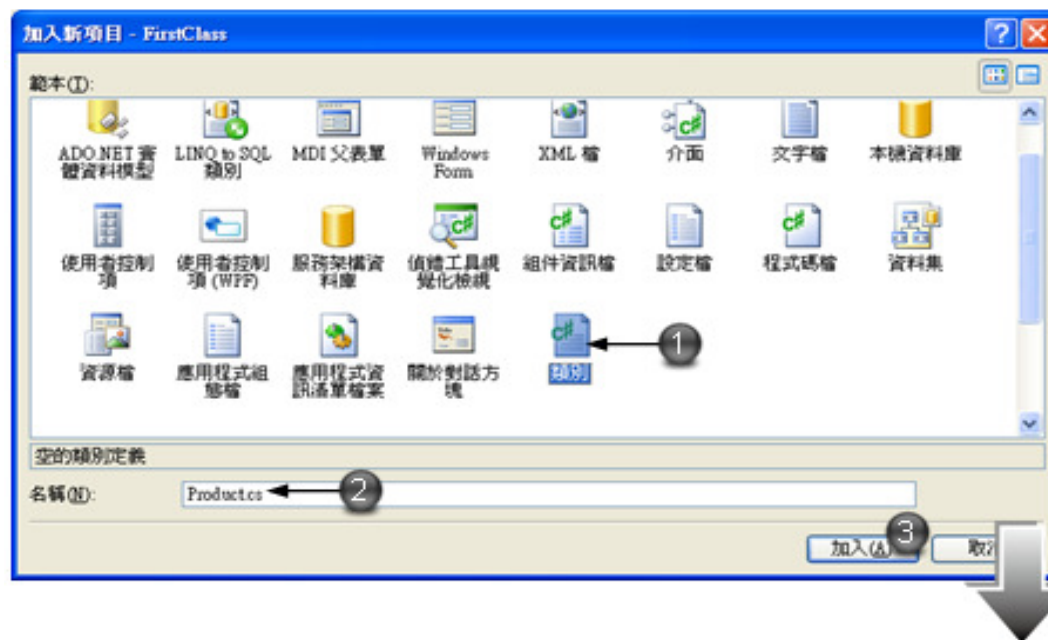
上機

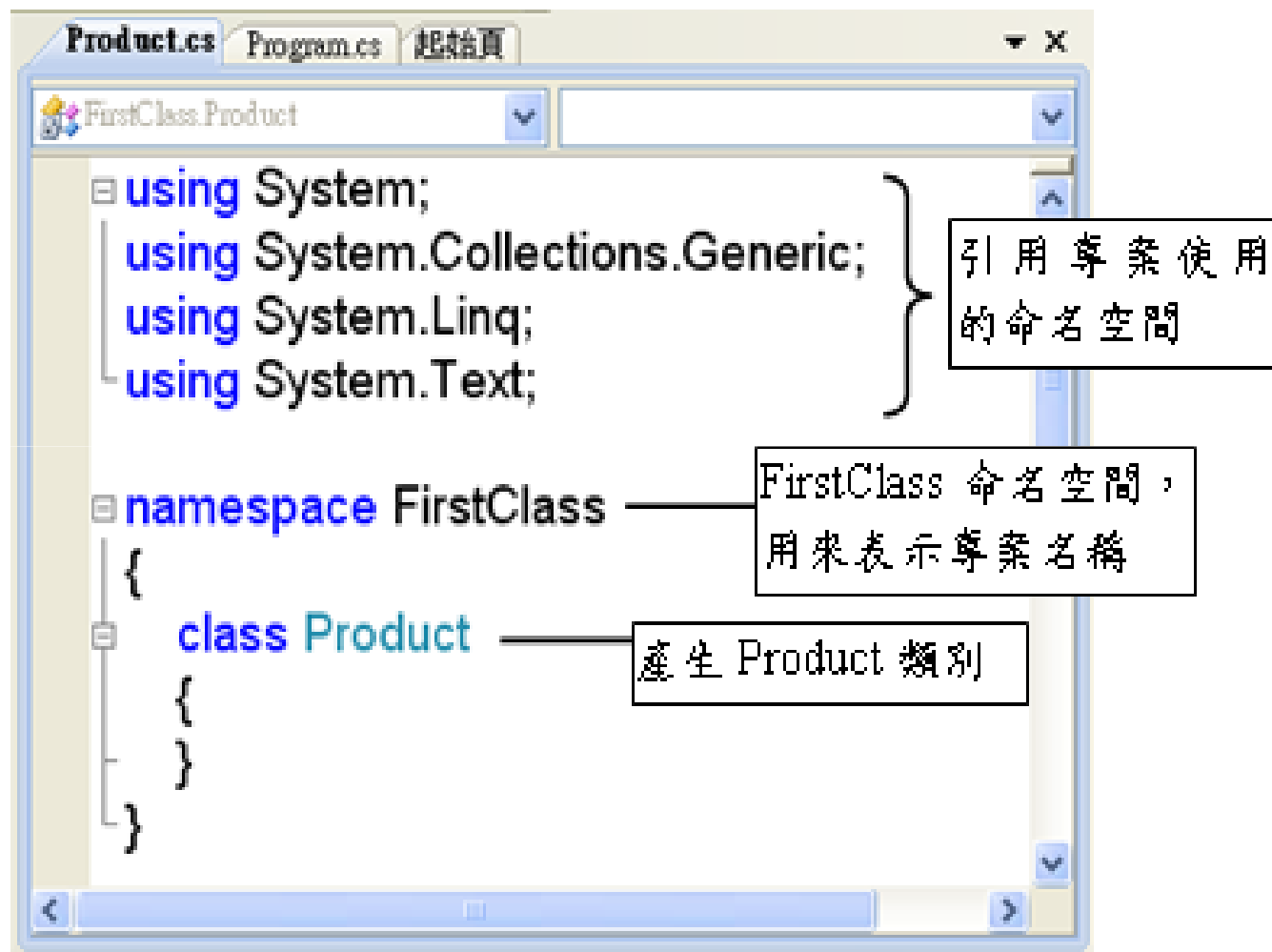
Step1 新增專案

新增「主控台應用程式」專案，專案名稱為FirstClass。

Step2 在專案中新增類別檔

執行功能表【專案(P)/加入類別(C)...】出現下圖







■ Step3 定義 Product 類別

在Product.cs程式碼中先定義 Prouduct 類別擁有PartNo 編號, PartName 品名, Qty 數量欄位, 以及可顯示產品編號, 品名, 數量的ShowInfo()方法。

上述 Product 類別中的成員因是 public, 所以可讓物件直接以「.»運算子來存取這些欄位。



```
// FileName : Product.cs
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
04 using System.Text;
05
06 namespace FirstClass
07 {
08     class Product
09     {
10         //宣告public 公用型的PartNo編號, PartName品名欄位
11         public string PartNo, PartName;
12         public int Qty;
13         public void ShowInfo()
14         {
15             Console.WriteLine("編號 : {0}", PartNo);
16             Console.WriteLine("品名 : {0}", PartName);
17             Console.WriteLine("數量 : {0}", Qty);
18             Console.WriteLine("=====");
19         }
20     }
21 }
```



Step4 撰寫 Main() 方法

切換到 Program.cs 類別檔，在該檔中撰寫如下程式。

在 Main() 方法可用 new 關鍵字來建立屬於 Proudct 類別的物件，如要存取類別中的成員，可用「.」運算子。



```
// FileName : Program.cs
01 namespace FirstClass
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             //宣告並建立DVD物件屬於Product類別
08             Product DVD = new Product();
09             DVD.PartNo = "B001";
10             DVD.PartName = "變形金剛2";
11             DVD.Qty = 20;
12             DVD.ShowInfo();
13             Product PDA;
14             PDA = new Product();
15             PDA.PartNo = "P001";
16             PDA.PartName = "惠普HP iPAQ PDA行動手機";
17             PDA.Qty = -5;
18             PDA.ShowInfo();
19             Console.Read();
20         }
21     }
22 }
```



6.1.4 使用存取子設定屬性

- 使用**public** 欄位來表示物件的狀態或屬性非常方便
- 如果物件的屬性狀態需受到控管或限制就必須用**存取子**。
- 前範例第17行 `PDA.Qty=-5;`，
將**PDA**產品的數量設為-5。
- 若希望 **Qty** 欄位值最小不能少於0，想將類別欄位值定義在某範圍內，可用「存取子」來達成。
- 存取子是一種特殊方法，它有 **get** 存取子和 **set** 存取子兩個區塊。
- **get {...}** 用來傳回私有欄位值。
- **set{...}** 用來設定類別私有欄位值。
- 若要限制某個欄位值的範圍，可將條件規則的敘述寫在**set{...}** 區塊。



存取子語法如下：

```
public 資料型別 屬性名稱
{
    get
    {
        return 傳回值;
    }
    set
    { //設定的屬性值會傳給value，value會放入set區塊
      [程式區塊;]
    }
}
```



```
// FileName : Product.cs
01 namespace Property
02 {
03     class Product {
04         public string PartNo, PartName;
05         private int _Qty;
06         // 設定數量Qty屬性不可小於0, 若小於0則設定_Qty欄位為0
07         public int Qty {
08             get {
09                 return _Qty;
10             }
11             set {
12                 if (value < 0) value = 0;
13                 _Qty = value;
14             }
15         }
16     }
17     public void ShowInfo() {
18         Console.WriteLine("編號 : {0}", PartNo);
19         Console.WriteLine("品名 : {0}", PartName);
20         Console.WriteLine("數量 : {0}", Qty);
21         Console.WriteLine("=====");
22     }
23 }
24 }
```



```
// FileName : Program.cs
01 namespace Property
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             //宣告並建立DVD物件屬於Product類別
08             Product DVD = new Product();
09             DVD.PartNo = "B001";
10             DVD.PartName = "變形金剛2";
11             DVD.Qty = 20;
12             DVD.ShowInfo();
13             Product PDA;
14             PDA = new Product();
15             PDA.PartNo = "P001";
16             PDA.PartName = "惠普HP iPAQ PDA行動手機";
17             PDA.Qty = -5;
18             PDA.ShowInfo();
19             Console.Read();
20         }
21     }
22 }
```

```
file:///C:/CSharp/chap06/Property...
編號 : B001
品名 : 變形金剛2
數量 : 20
=====
編號 : P001
品名 : 惠普HP iPAQ PDA行動手機
數量 : 0
=====
```



6.1.5 唯讀與唯寫屬性

設定唯讀屬性做法：

- 讓設定屬性的存取子只有get {...} 區塊，表示該屬性只能傳回值而不能設定值；
- 若是唯寫屬性就是讓存取子中只有set {...} 區塊，表該屬性只能設定值而不能傳回值。
- 唯讀屬性與唯寫屬性的寫法如下：

唯讀屬性

```
public 資料型別 屬性名稱
{
    get
    {
        return ;
    }
}
```

唯寫屬性

```
public 資料型別 屬性名稱
{
    set
    {
        [程式區塊];
    }
}
```




6.1.6 自動實作屬性

- 由上例知，屬性能讓類別隱藏實作或驗證程式碼同時，以公開方式取得並設定值。
- 類別的屬性還能繫結至控制項的屬性，這是欄位所做不到的。
- 建議將類別的公開欄位修改成屬性。
- 若將上例的 **PartNo** 編號及 **PartName** 品名公開欄位改成以屬性表示
在早期 C# (C# .NET, C# 2005) 版本需用存取子分別定義 **PartNo** 屬性及 **PartName** 屬性的 **set** 和 **get** 區塊來存取 **_PartNo** 和 **_PartName** 私有欄位。



```
class Product                //定義 Product 類別
{
    private string _PartNo, _PartName;  //私有欄位
    public string PartNo                //定義 PartNo 編號屬性
    {
        get{return _PartNo ;}
        set{_PartNo = value;}
    }
    public string PartName              //定義 PartName 品名屬性
    {
        get{return _PartName ;}
        set{_PartName = value; }
    }
    .....
    .....
}
```



- 當屬性一多上式必須逐一定義屬性存取子。
- C# 2008 開始提供「自動實作屬性」來解決這個問題。
- 透過自動屬性實作讓
屬性宣告更簡明，存取子不需額外邏輯或重複撰寫程式。
- 上例改以自動實作屬性方式來定義 PartName 及
PartNo 屬性
此時編譯器會自動建立私有 (Private) 匿名支援欄位，
讓該屬性只能透過存取子的 get 和 set 區塊來進行存取。

```
class Product                //定義 Product 類別
{
    public string PartNo{get ; set ;}    //定義 PartNo 編號屬性
    public string PartName{get ; set ;} //定義 PartName 品名屬性
    .....
    .....
}
```



6.2 建構函式

6.2.1 建構函式的使用

- 前例都先建立物件，再逐一設定該物件的屬性或欄位。
- 若想在建立物件同時完成物件屬性或欄位初始值設定
⇒ 用建構函式來達成。
- **C#** 中建構函式名稱必須和類別名稱相同，建構函式的使用方式和一般函式(方法)相同，當使用 **new** 關鍵字建立物件同時即會執行該類別中的建構函式。



範例演練

延續上例在 **Product** 類別中新增一個可傳入編號, 品名, 數量三個參數的建構函式, 讓使用者可用

`Product DVD = new Product ("B001", "變形金剛2", 15);`
在建立 **Product** 類別的 **DVD** 物件同時即指定 **PartNo**, **PartName** 以及 **Qty** 屬性的初值。



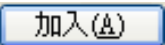


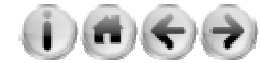
上機

■ Step1 新增專案

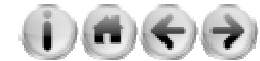
新增「主控台應用程式」專案，名稱為 Constructor。

■ Step2 建立Product 類別

- ① 執行功能表【專案(P)/加入類別(C)...】出現「加入新項目」視窗。
- ② 選取「類別」選項，再將「名稱(N)」設為 Product.cs (類別檔案名稱設為Product.cs)
- ③ 按  鈕開啟 Product.cs 檔。
- ④ 在 Product.cs 撰寫下面程式碼



```
// FileName : Product.cs
01 namespace Constructor
02 {
03     class Product
04     {
05         public string PartNo { get; set; }
06         public string PartName { get; set; }
07         private int _Qty;
08         // -----
09         public Product(string vNo, string vName, int vQty)
10         {
11             PartNo = vNo;
12             PartName = vName;
13             Qty = vQty;
14         }
15     }
16 }
```



```
15 // 設定庫存Qty屬性不可小於0, 若小於0則設定_Qty欄位為0
16 public int Qty
17 {
18     get
19     {
20         return _Qty;
21     }
22     set
23     {
24         if (value < 0) value = 0;
25         _Qty = value;
26     }
27 }
28 public void ShowInfo()
29 {
30     Console.WriteLine("編號 : {0}", PartNo);
31     Console.WriteLine("品名 : {0}", PartName);
32     Console.WriteLine("數量 : {0}", Qty);
33     Console.WriteLine("=====");
34 }
35 }
36 }
```




Step3 撰寫Main()方法

```
// FileName : Program.cs
01 namespace Constructor
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             // Product cpu = new Product(); 無法使用
08             // 使用建構函式建立物件時並給予初值
09             Product cpu = new Product("B001", "變形金剛2", 15);
10             cpu.ShowInfo();
11             Console.Read();
12         }
13     }
14 }
```



6.2.2 建構函式的多載

- 若希望在建構物件同時，也能呼叫不帶引數或帶引數的建構函式，必須使用建構函式的多載。
- 可定義建構函式，並以不同的引數個數，不同引數的資料型別來加以區隔不同的建構函式。
- 建構函式多載和一般的方法多載(函式多載)使用方式相同。



範例演練

延續上例在 Product 類別中再新增一個不帶引數的建構函式，當使用者使用

Product 物件名稱 = new Proudct();

呼叫不帶引數的Product() 建構函式時，不帶引數的Product() 建構函式會將”送審中”、”品名未定”、0 指定給 Product 類別的 PartNo, PartName, Qty 屬性。最後再加入 ShowInfo()方法多載(函式多載)，透過此方法可設定PartNo, PartName, Qty並顯示產品資訊。

```
file:///C:/CSharp/chap06/OverLoads/...
編號 : 送審中
品名 : 品名未定
數量 : 0
=====
送審中的品名更新後...
編號 : G001
品名 : 火影忍者-件
數量 : 10
=====
編號 : B001
品名 : 變形金剛2
數量 : 15
=====
```



上機

Step1 新增專案

新增主控台應用程式專案，名稱設為 OverLoads。

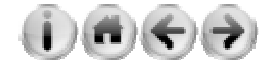
Step2 建立 Product 類別

① 執行功能表的【專案(P)/加入類別(C)...】指令
新增 Product.cs 類別檔。

② 在 Product.cs 撰寫下面程式碼



```
// FileName : Product.cs
01 namespace OverLoads
02 {
03     class Product
04     {
05         public string PartNo{get;set;}
06         public string PartName { get; set; }
07         private int _Qty;
08         public Product()
09         {
10             PartNo = "送審中";
11             PartName = "品名未定";
12             Qty = 0;
13         }
14     }
15 }
```



```
14     public Product(string vNo, string vName, int vQty)
15     {
16         PartNo = vNo;
17         PartName = vName;
18         Qty = vQty;
19     }
20     // 設定庫存Qty屬性不可小於0, 若小於0則設定_Qty欄位為0
21     public int Qty
22     {
23         get
24         {
25             return _Qty;
26         }
27         set
28         {
29             if (value <= 0) value = 0;
30             _Qty = value;
31         }
32     }
```



```
33    //此ShowInfo()多載方法可顯示產品資訊
34    public void ShowInfo()
35    {
36        Console.WriteLine("編號：{0}", PartNo);
37        Console.WriteLine("品名：{0}", PartName);
38        Console.WriteLine("數量：{0}", Qty);
39        Console.WriteLine("=====");
40    }
41    //此ShowInfo()多載方法可設定產品的編號, 品名, 數量，並同時顯示資品資訊
42    public void ShowInfo(string vNo, string vName, int vQty)
43    {
44        PartNo = vNo;
45        PartName = vName;
46        Qty = vQty;
47        ShowInfo();
48    }
49 }
50 }
```



Step3 撰寫 Main() 方法

```
// FileName : Program.cs
01 namespace OverLoads
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             Product Game = new Product(); //無參數之建構函式
08             Game.ShowInfo();
09             Console.WriteLine("送審中的品名更新後...");
10             Game.ShowInfo("G001", "火影忍者-伴", 10);
11             //使用此建構函式建立物件時並給予編號,品名,數量的初值
12             Product DVD = new Product("B001", "變形金剛2", 15);
13             DVD.ShowInfo();
14             Console.Read();
15         }
16     }
17 }
```




6.2.3 物件初始設定式

- 當使用 `new` 建立物件，此時會呼叫指定的多載建構函式
- 若初始化物件的屬性很多，要定義多個多載建構函式。
- 如：Employee 員工類別含有：
EmpID 編號、EmpName 姓名、EmpTel 電話、
EmpAdd 住址以及 EmpSalary 薪水屬性
- 希望建立物件同時可初始化員工物件0~5個
屬性內容，最少要定義下面六個 Employee
建構函式，且在初始化物件屬性內容會變得
非常麻煩。



```
public class Employee           //定義 Employee 類別
{
    public string EmpID { set; get; }
    public string EmpName { set; get; }
    public string EmpTel { set; get; }
    public string EmpAdd { set; get; }
    private int _Salary;         //_Salary 薪資欄位
    public int EmpSalary          //Salary 屬性必須大於 20000 以上
    {
        get{return _Salary ;}
        set{
            if(value <=20000) value=20000 ;
            _Salary = value ;
        }
    }
    public Employee(){}          //建構函式#1
    public Employee(string vID){EmpID=vID ;}    //建構函式#2
    public Employee(string vID, string vName){
        EmpID=vID ; EmpName=vName ;}          //建構函式#3
    .....
    public Employee(string vID, string vName,    //建構函式#6
        string vTel, string vAdd, int vSalary){
        EmpID=vID ; EmpName=vName ; EmpTel=vTel,
        EmpAdd=vAdd; EmpSalary=vSalary;}
    }
}
```



■ 簡例示範如何使用物件初始設定式來初始化

Employee 類別物件的

- **EmpID**編號
- **EmpName**姓名
- **EmpTel**電話
- **EmpAdd**住址
- **EmpSalary**薪資

的內容，完全不需用類別建構函式來初始化物件的屬性值。



```
public class Employee           //定義 Employee 類別
{
    public string EmpID { set; get; }
    public string EmpName { set; get; }
    public string EmpTel { set; get; }
    public string EmpAdd { set; get; }
    private int _Salary;         //_Salary 薪資欄位
    public int EmpSalary         //Salary 屬性必須大於 20000 以上
    {
        get{return _Salary;}
        set{
```

```
            if(value <=20000) value=20000 ;
            _Salary = value ;
        }
    }

    static void Main(string[] args)
    {
        //Mary 物件設定員工編號為 "A01"
        Employee Mary = new Employee {EmpID="A01"};

        //Jack 物件設定員工編號為 "B01", 姓名為 "傑克", 薪資為 25000
        Employee Jack = new Employee
        {EmpID="B01", EmpName="傑克",EmpSalary="25000"};

        //Tom 物件初始化員工編號, 姓名, 薪資, 住址, 電話
        Employee Tom = new Employee
        { EmpID="B01", EmpName="湯姆",EmpSalary="25000" ,
        EmpAdd="台中市忠明南路 1 號", EmpTel="04-1236587"};
    }
}
```



範例演練

- 將上面簡例寫成一個完整的範例，並練習使用「物件初始設定式」的宣告方式來初始化物件的屬性值。
- 定義Employee員工類別擁有EmpID編號、EmpName姓名、EmpTel電話、EmpAdd住址欄位以及EmpSalary薪水屬性，並定義EmpSalary屬性的值不可小於20000；定義ShowInfo()方法用來顯示員工的所有資訊。
- 最後在Main()方法使用物件初始設定式來初始化MoMo(莫莫)及Dora(朵拉)兩個員工物件，最後再將這兩位員工的資料顯示出來。

```
file:///C:/CSharp/chap06/ObjectSetValue/...  
編號 : A01  
姓名 : 莫莫  
電話 : 04-7895642  
住址 : 台中市中山路一段1號  
薪資 : 30000  
=====  
編號 : A02  
姓名 : 朵拉  
電話 : 02-1234567  
住址 : 台北市南港路一段2號  
薪資 : 20000  
=====
```



上機

Step1 新增專案

新增主控台應用程式專案，名稱設為ObjectSetValue。

Step2 建立 Employee 類別

執行功能表的【專案(P)/加入類別(C)...】指令新增「Employee.cs」類別檔。

在 Employee.cs 撰寫下列程式碼：



```
// FileName : Employee.cs
01 namespace ObjectSetValue
02 {
03     class Employee
04     {
05         public string EmpID { set; get; }
06         public string EmpName { set; get; }
07         public string EmpTel { set; get; }
08         public string EmpAdd { set; get; }
09         private int _Salary;
10         public int EmpSalary
11         {
12             get
13             {
14                 return _Salary;
15             }
16         }
17     }
18 }
```



```
16      set
17      {
18          if (value <= 20000) value = 20000;
19          _Salary = value;
20      }
21  }
22  //顯示員工資訊
23  public void ShowInfo()
24  {
25      Console.WriteLine("編號 : {0}", EmpID);
26      Console.WriteLine("姓名 : {0}", EmpName);
27      Console.WriteLine("電話 : {0}", EmpTel);
28      Console.WriteLine("住址 : {0}", EmpAdd);
29      Console.WriteLine("薪資 : {0}", EmpSalary.ToString());
30      Console.WriteLine("=====");
31  }
32  }
33 }
```




Step3 撰寫Main()方法

```
// FileName : Program.cs
01 namespace ObjectSetValue
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             Employee MoMo = new Employee { EmpID = "A01", EmpName =
                "莫莫", EmpAdd = "台中市中山路一段1號", EmpTel = "04-7895642",
                EmpSalary = 30000 };
08             Employee Dora = new Employee { EmpID = "A02", EmpName = "朵拉",
                EmpAdd = "台北市南港路一段2號", EmpTel = "02-1234567",
                EmpSalary = 10000 };
09             MoMo.ShowInfo();
10             Dora.ShowInfo();
11             Console.Read();
12         }
13     }
14 }
```



6.3 靜態成員

- 使用 **static** 關鍵字宣告的成員稱為「**靜態成員**」
- 靜態成員可讓同類別建立的物件都可一起共用
- 靜態成員不需用 **new** 建立物件就可直接使用，
- 必須透過類別名稱再加上「**.**」運算子直接呼叫 **public** 的靜態成員即可。
- 呼叫靜態成員寫法：
 - 類別名稱.欄位
 - 類別名稱.屬性
 - 類別名稱.方法([引數串列])



範例演練

- 在 Product 類別新增一個 `private static _num` 靜態欄位，用來記錄目前共生產幾個產品；新增 `public static Num` 靜態唯讀屬性，用來讀取目前生產產品的個數；新增 `public static ShowNum()` 靜態方法，用來印出“目前共生產幾個產品”的訊息。

```
file:///C:/CSharp/chap06/StaticMember/...
編號 : B001
品名 : 變形金剛2
數量 : 20
=====
目前共生產 1 個產品!!

編號 : P001
品名 : 惠普HP iPAQ PDA行動手機
數量 : 10
=====
目前建立第 2 個產品!!
```



上機

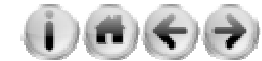
Step1 新增主控台應用程式專案

新增「主控台應用程式」專案，名稱為 StaticMember。

Step2 建立 Product 類別

執行功能表的【專案(P)/加入類別(C)...】指令新增
「Product.cs」類別檔。

在 Product.cs 撰寫下列程式碼：



```
// FileName : Product.cs
01 namespace StaticMember
02 {
03     class Product
04     {
05         public string PartNo { get; set; }
06         public string PartName { get; set; }
07         private int _Qty;
08         private static int _num;
09         public static void ShowNum()
10         {
11             Console.WriteLine("目前共生產 {0} 個產品!!\n", _num);
12         }
13     }
14 }
```



```
13     public Product()
14     {
15         _num += 1;
16         PartNo = "送審中";
17         PartName = "品名未定";
18         Qty = 0;
19     }
20     // 可設定編號,品名,數量的Product建構函式
21     public Product(string vNo, string vName, int vQty)
22     {
23         _num += 1;
24         PartNo = vNo;
25         PartName = vName;
26         Qty = vQty;
27     }
```



```
28     public static int Num
29     {
30         get
31         {
32             return _num;
33         }
34     }
35     // 設定庫存Qty屬性不可小於0, 若小於0則設定_Qty欄位為0
36     public int Qty
37     {
38         get
39         {
40             return _Qty;
41         }
42         set
43         {
44             if (value < 0) value = 0;
45             _Qty = value;
46         }
47     }
```



```
48     public void ShowInfo()
49     {
50         Console.WriteLine("編號 : {0}", PartNo);
51         Console.WriteLine("品名 : {0}", PartName);
52         Console.WriteLine("數量 : {0}", Qty);
53         Console.WriteLine("=====");
54     }
55     public void ShowInfo(string vNo, string vName, int vQty)
56     {
57         PartNo = vNo;
58         PartName = vName;
59         Qty = vQty;
60         ShowInfo(); //呼叫Product類別的ShowInfo()方法
61     }
62 }
63 }
```




Step3 撰寫Main方法

```
// FileName : Program.cs
01 namespace StaticMember
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             Product DVD = new Product("B001", "變形金剛2", 20);
08             DVD.ShowInfo();
09             Product.ShowNum();
10             Product PDA = new Product("P001", "惠普HP iPAQ PDA行動手機", 10);
11             PDA.ShowInfo();
12             //呼叫Product類別的Num靜態屬性
13             Console.WriteLine("目前建立第 {0} 個產品!!", Product.Num);
14             Console.Read();
15         }
16     }
17 }
```



6.4 物件陣列

欲使用類別建立物件陣列，先建立屬於該類別的陣列元素，再逐一用 **new** 關鍵字對每個陣列元素做物件實體化。

寫法

- ① 先建立 **p[0]~p[4]** 的五個陣列元素是屬於 **Proudct** 類別的物件，此時 **p[0]~p[4]** 的值皆為 **null** 並未做物件實體化的動作
- ② 再用 **new** 關鍵字對 **p[0]~p[4]** 做物件實體化
- ③ 最後逐一設定 **p[0]~[4]** 每個陣列的屬性內容
設定方式是在 **[]** 中括號後加上「**.**」運算子。



```
Product[] p = new Product[5];  
p[0] = new Product();  
p[0].PartNo = "A01"; p[0].PartName = "火影忍者" ; p[0].Qty = 100  
p[1] = new Product();  
p[1].PartNo = "A02"; p[1].PartName = "哈利波特" ; p[1].Qty = 250  
.....  
.....  
p[4] = new Product();  
p[4].PartNo = "A05"; p[4].PartName = "網球王子" ; p[4].Qty = 50
```



範例演練

- 延續上例，使用 **Product** 類別來建立物件陣列。程式開始執行先詢問要產生多少個產品，接著再讓使用者逐一輸入產品的編號、品名及數量。如下圖先輸入3，接著產生p陣列物件且陣列元素為p[0]~p[2]，再讓使用者逐一輸入p[0]~p[2]的編號、品名、數量的資料，最後再印出”目前共生產幾個產品”的訊息。

```
file:///C:/CSharp/chap06/ObjectArray/...
請輸入欲產生幾個產品：3
請輸入第 1 筆產品
編號：A01
品名：金剛狼
數量：500
=====
請輸入第 2 筆產品
編號：A02
品名：變形金剛3
數量：600
=====
請輸入第 3 筆產品
編號：A03
品名：生人勿近
數量：1000
=====
目前共生產 3 個產品!!
```



```
// FileName : Product.cs
01 namespace ObjectArray
02{
03     class Product
04     {
05         public string PartNo { get; set; }
06         public string PartName { get; set; }
07         private int _Qty;
08         private static int _num;
09         //顯示共產生幾個產品
10         public static void ShowNum()
11         {
12             Console.WriteLine("目前共生產 {0} 個產品!!\n", _num);
13         }
14     }
15 }
```



```
14     public Product()
15     {
16         _num += 1;
17     }
18     // 設定數量Qty屬性不可小於0, 若小於0則設定_Qty欄位為0
19     public int Qty
20     {
21         get
22         {
23             return _Qty;
24         }
25         set
26         {
27             if (value < 0) value = 0;
28             _Qty = value;
29         }
30     }
31 }
32 }
```



Step3 撰寫 Main 方法

```
// FileName : Program.cs
01 namespace ObjectArray
02 {
03     class Program
04     {
05         static void Main(string[] args)
06         {
07             try
08             {
09                 Console.Write("請輸入欲產生幾個產品：");
10                 int n = int.Parse(Console.ReadLine());
11                 Product[] p = new Product[n];
12                 for (int i = 0; i <= p.GetUpperBound(0); i++)
13                 {
```



```
14         p[i] = new Product();
15         Console.WriteLine("請輸入第 " + (i + 1).ToString() + " 筆產品");
16         Console.Write(" 編號 : ");
17         p[i].PartNo = Console.ReadLine();
18         Console.Write(" 品名 : ");
19         p[i].PartName = Console.ReadLine();
20         Console.Write(" 數量 : ");
21         p[i].Qty = int.Parse(Console.ReadLine);
22         Console.WriteLine("=====");
23     }
24     Product.ShowNum();
25 }
26 catch (Exception ex)
27 {
28     Console.WriteLine(ex.Message);
29     Console.WriteLine("輸入資料有誤, 準備離開程式!");
30 }
31 Console.Read();
32 }
33 }
34 }
```




6.5 類別繼承

- 在物件導向程式設計中最為有力的機制就是繼承，透過類別繼承可增加程式的延展性。
- 下列寫法 `class B:A` 表示 B繼承A，此時可讓被繼承類別A所有功能延伸到類別B
- 被繼承的類別A可稱為基底類別(Base class)或父類別(Parent class)
- 繼承的類別B可稱為衍生類別(Derived class)或子類別(Child class)。

```
class A
{
    .....           // A類別的欄位, 屬性, 方法
}
class B : A           //使用「:」指定類別B要繼承類別A
{
    .....           //此時類別B會擁有類別A所有非private的成員
    .....           //然後加入新的欄位, 屬性, 方法
}
```



- 衍生類別無法使用基底類別的**private**成員
- 但可用基底類別的 **public** 和 **protected** 成員。
- 若衍生類別成員想要用基底類別的成員但不想公開
此時基底類別的成員必須宣告為 **protected** 保護層級。



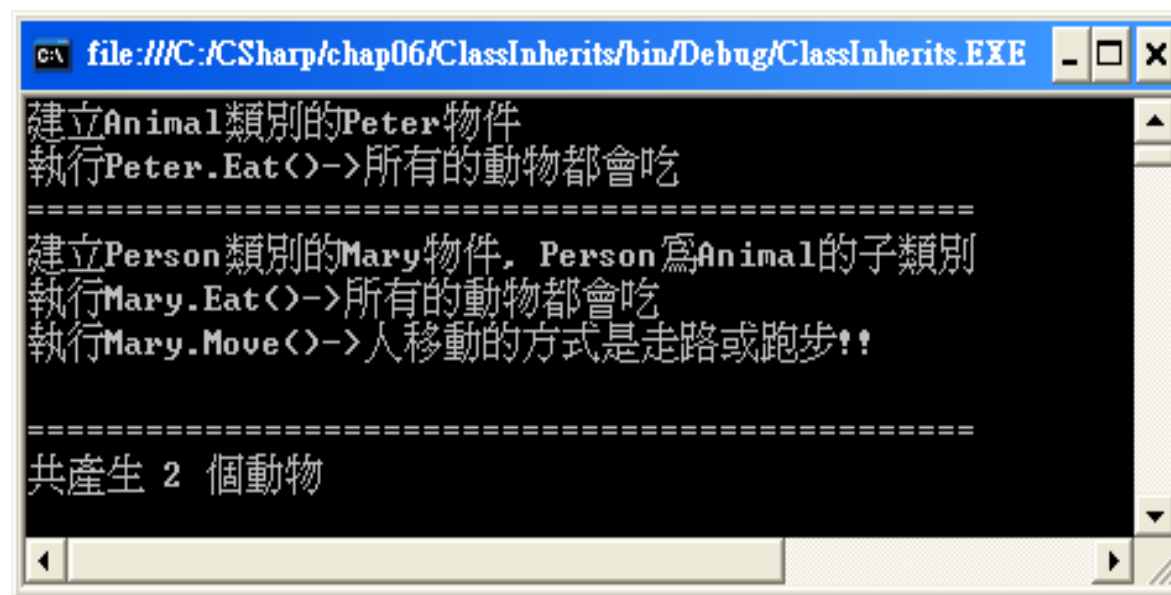
範例演練

繼承範例實作，說明如下：

- ① 定義 **Animal** 基底類別有 **protected** 保護層級的 **_Num** 靜態欄位用來計算目前共產生多少個動物；執行 **public** 建構式時 **_Num** 欄位會加1，並在該類別定義 **Eat()** 方法與 **ShowNum()** 靜態方法，**ShowNum** 靜態方法()用來顯示目前共產生多少個動物。
- ② 定義 **Person** 衍生類別繼承自 **Animal** 基底類別並加入 **public** 的 **Move()** 方法，此時 **Person** 類別會擁有 **_Num** 欄位、**ShowNum()** 靜態方法，**Eat()** 和 **Move()** 方法，且建立 **Person** 物件時也會執行 **Animal** 基底類別的建構式。
- ③ 在 **Main()** 方法使用 **Animal** 類別及 **Person** 類別建立物件並執行 **Eat()** 和 **Move()** 方法，最後再呼叫 **Animal.ShowNum()** 或 **Person.ShowNum()** 來顯示共產生多少個動物。



執行情形




```
file:///C:/CSharp/chap06/ClassInherits/bin/Debug/ClassInherits.EXE
建立Animal類別的Peter物件
執行Peter.Eat()->所有的動物都會吃
=====
建立Person類別的Mary物件, Person為Animal的子類別
執行Mary.Eat()->所有的動物都會吃
執行Mary.Move()->人移動的方式是走路或跑步!!
=====
共產生 2 個動物
```

- **Step1新增主控台應用程式專案**
新增「主控台應用程式」專案，專案名稱為ClassInherits。
- **Step2定義Animal類別**
執行功能表的【專案(P)/加入類別(C)...】指令
新增「Animal.cs」類別檔。接著請在
Animal.cs 撰寫下列程式碼：

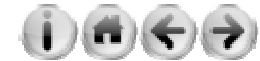


```
// FileName : Animal.cs
01 namespace ClassInherits
02 {
03     class Animal
04     {
06         protected static int _Num;
08         public Animal()
09         {
10             _Num++;
11         }
12         public void Eat()
13         {
14             Console.WriteLine("所有的動物都會吃");
15         }
16         public static void ShowNum()
17         {
18             Console.WriteLine("共產生 {0} 個動物", _Num);
19         }
20     }
21 }
```



 **Step3 定義Person類別**
執行功能表的【專案(P)/加入類別(C)...】指令新增
「Person.cs」類別檔。接著在Person.cs 撰寫如下程式碼：

```
// FileName : Person.cs
01 namespace ClassInherits
02 {
03     //Person類別繼承Animal類別, Person類別會擁有Animal類別的功能
04     class Person : Animal
05     {
06         //Person類別會移動，而移動的方式是走路或跑步
07         public void Move()
08         {
09             Console.WriteLine("人移動的方式是走路或跑步!!\n");
10         }
11     }
12 }
```



Step4 撰寫Main方法

// FileName : Program.cs

01 namespace ClassInherits

02 {

03 class Program

04 {

05 static void Main(string[] args)

06 {

07 Animal Peter = new Animal();

08 Console.WriteLine("建立Animal類別的Peter物件");

09 Console.Write("執行Peter.Eat()->");

10 Peter.Eat();

11 Console.WriteLine ("=====");

12 Person Mary = new Person();

13 Console.WriteLine ("建立Person類別的Mary物件, Person為Animal的子類別");

14 Console.Write("執行Mary.Eat()->");

15 Mary.Eat();

16 Console.Write("執行Mary.Move()->");

17 Mary.Move();

18 Console.WriteLine ("=====");

19 Animal.ShowNum();

20 Console.Read();

21 }

22 }

23 }





6.6 使用主控台程式建立視窗程式

- .NET Framework 的類別程式庫就是以類別繼承的架構而來，因此我們也可以自己撰寫一個類別然後繼承 .NET Framework 的類別並加以延伸。
- FirstForm.sln 範例使用主控台應用程式並繼承 .NET 的 System.Windows.Forms 命名空間的 Form 類別來實作視窗應用程式。



範例演練


- 建立一個 MyForm 類別繼承 System.Windows.Forms.Form 類別，在 MyForm 類別中加入  鈕，並建立該鈕的 Click 事件，使得按  鈕之後會顯示右圖的對話方塊訊息。





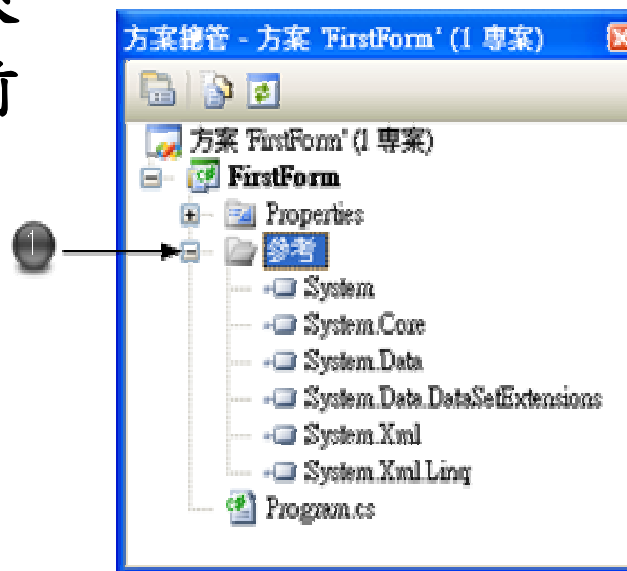
上機



 **Step1 新增主控台應用程式專案**
新增「主控台應用程式」專案，名稱為 FirstForm。

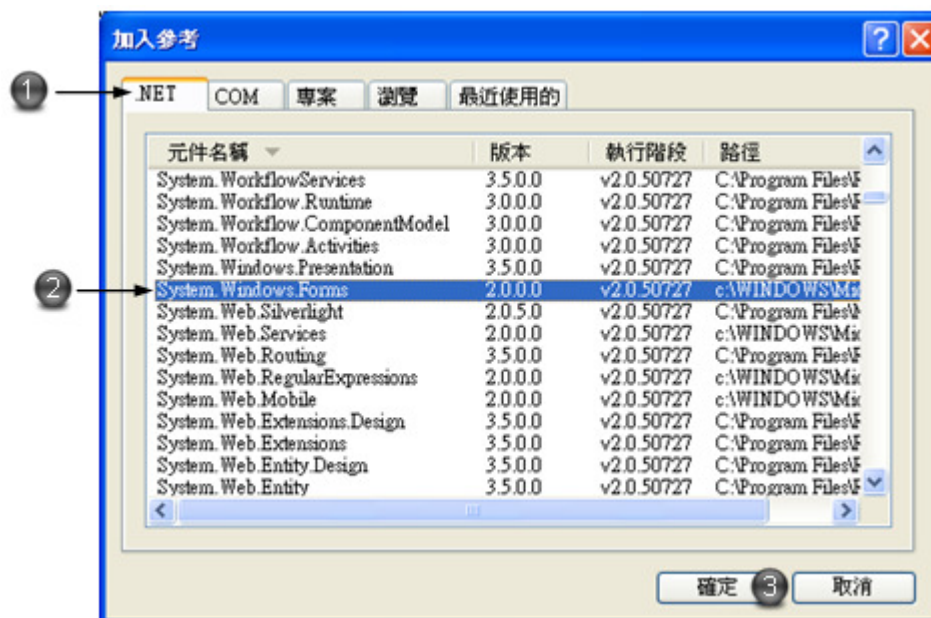
 **Step2 加入 System.Windows.Forms 命名空間**

1. 在方案總管按「參考」資料夾
此時參考資料夾下會列出目前
專案引用參考的命名空間。



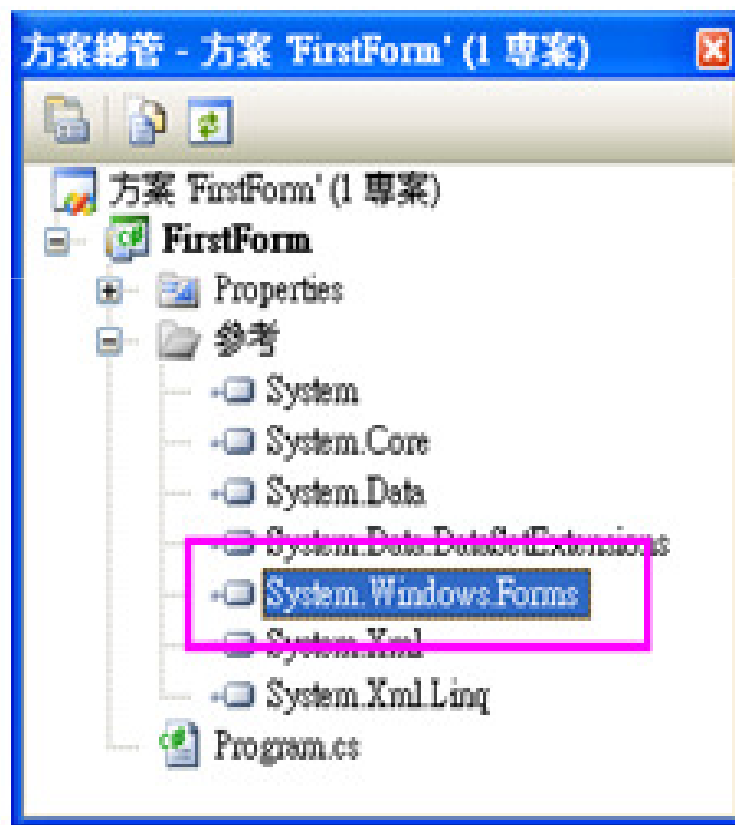


2. 要建立一個 MyForm 類別，繼承自 System.Windows.Forms.Form 類別並製作視窗程式，主控台應用程式預設沒加入 System.Windows.Forms 命名空間的參考執行功能表的【專案(P)/加入參考(R)】並依照圖示在「加入參考」視窗加入「System.Windows.Forms」※元件名稱(即命名空間)。





3. 完成上述步驟後，方案總管視窗的參考資料夾下會出現「System.Windows.Forms」命名空間。





■ Step3 撰寫程式碼

- 在 **Program.cs** 撰寫如下程式。
- 建立**MyForm** 類別並繼承 **System.Windows.Forms** 命名空間的**Form**類別
- 此時 **MyForm** 類別會擁有**Form**表單視窗的所有屬性及方法
- 接著在 **MyForm**類別的建構函式加入一個 **Button** 類別的**btnOk** 物件(或稱控制項)。



```
// FileName : Program.cs
01 using System;
02 using System.Collections.Generic;
03 using System.Linq;
04 using System.Text;
05 using System.Windows.Forms;
06
07 class MyForm : Form
08 {
09     Button btnOk;
10     public MyForm()
11     {
12         btnOk = new Button();
13         btnOk.Text = "確定";
14         btnOk.Width = 60;
15         btnOk.Height = 30;
16         btnOk.Visible = true;
17         btnOk.Left = 20;
18         btnOk.Top = 15;
```



```
19    this.Width = 200;
20    this.Height = 100;
21    this.Controls.Add(btnOk);
22    // 表單的標題顯示"第一個視窗程式"
23    this.Text = "第一個視窗應用程式";
24    // 指定btnOk按鈕的Click事件被觸發時會執行Click_Event事件處理函式
25    // 所以當按下btnOk鈕時會執行Click_Event事件處理函式
26    btnOk.Click += new EventHandler(Click_Event);
27 }
```

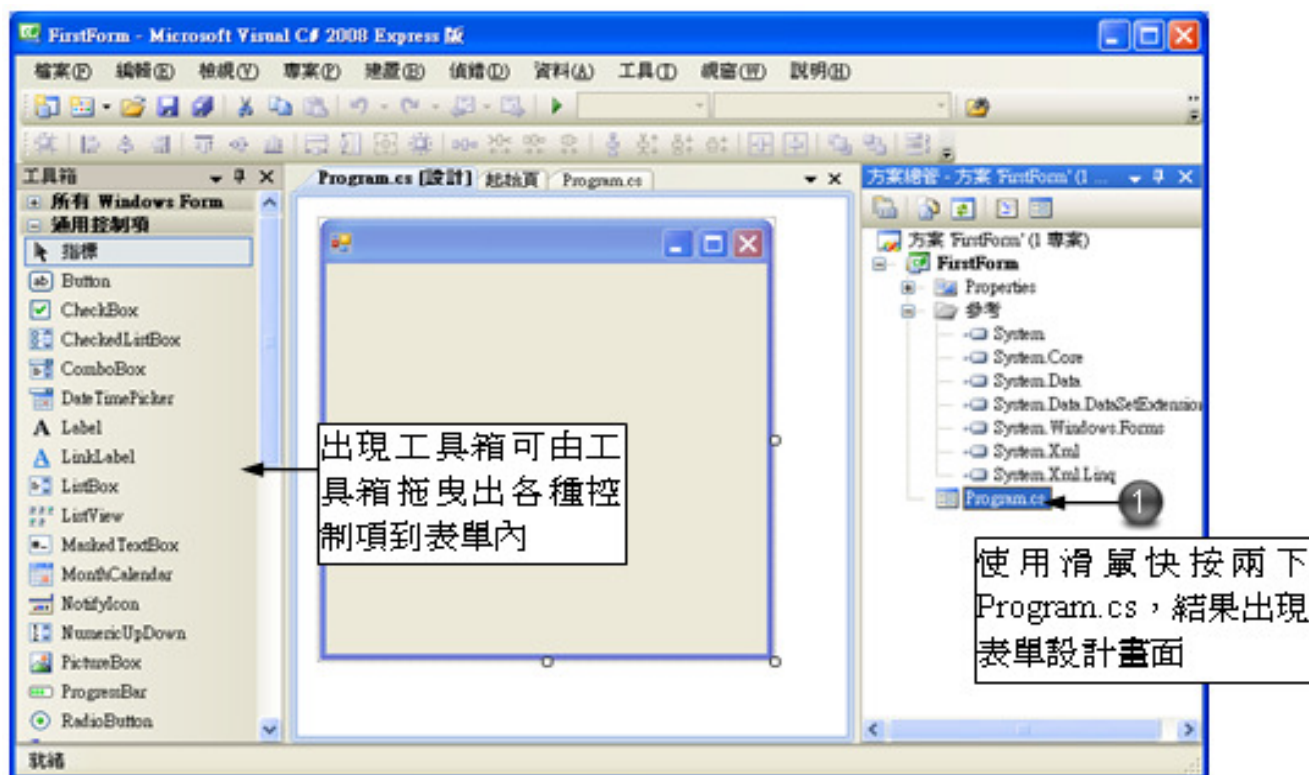


```
28 //Click_Event事件處理函式用來處理btnOk.Click事件
29 private void Click_Event(System.Object sender, System.EventArgs e)
30 {
31     MessageBox.Show("Hello World...\n歡迎光臨Windows Form視窗程式");
32 }
33 }
34
35 namespace FirstForm
36 {
37     class Program
38     {
39         static void Main(string[] args)
40         {
41             MyForm f = new MyForm();
42             f.ShowDialog();
43         }
44     }
45 }
```




■ Step4 顯示表單設計畫面

接著快按 **Program.cs** 檔兩下，結果會顯示表單的設計畫面，且右方會顯示工具箱。





Step5 執行程式

執行功能表的【偵錯(D)/開始偵錯(S)】指令
測試結果。

