.NET 程式設計入門(使用 C#)

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

命名空間 (1)

- 使用命名空間的好處可以將功能類似的類別組織在一起
- 命名空間允許巢狀的結構，形成階層式的架構，更容易分類管理
- 若在同一個程式檔中宣告二個名稱一樣的類別時，編譯會發生錯誤，我們可以利用命名空間來解決此問題
- 利用 **using** 關鍵字來指定需要的命名空間
- 語法

```
namespace 命名空間名稱
{
    類別;
}
```

命名空間 (2)

- 用法

```
namespace IBM
{
   class Notebook
   {
   }
}
namespace Compaq
{
   class Notebook
   {
   }
}
```

類別

- 類別主要的功能是用來描述定義物件的模樣
- 類別主要包含下列成員
  - 欄位 (field)
  - 屬性 (property)
  - 方法 (method)
  - 事件 (event)

類別實例

- 我們可以建立一個用來描述『一台車子』的類別
- 在該類別中，可以建立有關車子的欄位資料，像是車牌號碼、排氣量…
- 接著設定該類別的屬性，讓使用者可以藉由屬性存取類別中的欄位資料
- 一台車子的功能包含前進、後退、換車牌、顯示目前油量等，可定義於類別方法中
- 該類別的可能事件有警報聲響起，發生碰撞，車門被開啟…

定義類別

- 語法

```
存取修飾詞 class 類別名稱
{
   類別成員;
}
```

- 用法

```
public class car
{
   string id;
}
```

- 類別存取修飾詞
  - public – 不同組件也可以被引用
  - internal – 用一個組件才可以被引用 (預設)

物件

- 我們可以把類別想像成是車子架構的設計圖，而物件就是根據車子架構設計圖所設計出來的車子
- 類別的使用需產生該類別的實體物件，我們可以透過 new 關鍵字來完成
- 語法
  - 類別名稱 物件名稱 = new 類別名稱();
- 用法
  - car myCar = new car();
  - car myCar;
    myCar = new car();

實例探討 sample4-a1 (1)

- 程式功能
  - 定義一個 car 類別
  - 在主程式中產生 car 實體物件

實例探討 sample4-a1 (2)

- 程式內容

```
namespace sample5_a2
{
   class Class1
   {
     static void Main(string[] args)
     {
              car c = new car();
     }
   }
   class car
   {
   }
}
```

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

成員存取修飾詞

- public
  - 任何外部類別都可不受限制存取此類別成員
- private
  - 此類別成員只能在此類別中使用
- protected
  - 此類別成員可在此類別及繼承此類別的子類別使用
- internal
  - 在同一個組件中都可存取此類別成員
- protected internal
  - 提供 protected 及 internal 二種存取方式

欄位成員

- 欄位為類別中所定義的各種資料型別變數
- 當我們產生實體物件後，可透過 "." 來存取欄位資料
  - ex：myCar.id
- 語法
  - 存取修飾詞 資料型別 欄位名稱;
- 用法
  - private string id;
  - public int num;

實例探討 sample4-a2 (1)

- 程式功能
  - 建立 car 類別
  - 在 car 類別中定義 id 欄位
  - 設定並列印出 id 欄位的值
- 程式內容

```
class car
{
   public string idField; //類別欄位 (field)

}
```

實例探討 sample4-a2 (2)

- 程式內容

```
static void Main(string[] args)
{
    //產生實體物件
    car c = new car();

    //設定並列印結果
    c.idField = "ABC-123";
    Console.WriteLine("車牌號碼：" + c.idField);
}
```

屬性成員 (1)

- 屬性用來存取類別的欄位值
- 實體物件一樣可透過 "." 來存取屬性資料
- 語法

```
存取修飾詞  資料型別  屬性名稱
{
    get
    {
      return 欄位名稱;
    }
    set
    {
      欄位名稱 = value;
    }
}
```

屬性成員 (2)

- 用法

```
public string id
{
    get
    {
      return idField;
    }
    set
    {
      idField = value;
    }
}
```

- 說明
  - 我們可以只設定 get 部份讓該屬性成唯讀屬性

欄位與屬性成員

- 利用將欄位成員設定成 public，讓實體物件可以直接存取的方法雖然簡單，但卻無法提供任何額外的控制
  - ex：speed 欄位只能介於 1 ~ 100 之間
- 較好的設計方式為將欄位設定成 private，讓實體物件透過屬性成員來存取欄位成員的值
- 我們可以在定義屬性成員時，加入存取方式的控制

實例探討 sample4-a3 (1)

- 程式功能
  - 建立 car 類別
  - 在 car 類別中定義 speed 欄位、speed 屬性
  - speed 屬性值介於 0 ~ 100 間
- 程式內容

```
class car
{
    private int speedField; //類別欄位 (field)
```

實例探討 sample4-a3 (2)

- 程式內容

```
public int speed
{
  get
  {
        return speedField;
  }
  set
  {
        if(value < 0) value = 0;
        else if(value >100) value = 100;
        speedField = value;
  }
}
}
```

實例探討 sample4-a3 (3)

- 程式內容

```
static void Main(string[] args)
{
   car myCar = new car();
   myCar.speed = -200;
   Console.WriteLine("目前車速：" +

    myCar.speed);
}
```

課堂練習 sample4-b1 (1)

- 類別功能
  - 建立 empolyee 類別
  - 類別中含有 baseSalary、salary 及 benefit 屬性
  - baseSalary 屬性必須大於等於 0
  - salary 爲唯讀屬性，其值爲 baseSalary 加上 benefit
- 程式功能
  - 主程式中請使用者輸入底薪及獎金值
  - 列印出類別 salary 屬性值

課堂練習 sample4-b1 (2)

- 基本概念
  - 在 **baseSalary** 屬性定義中判斷使用者輸入的資料是否正確
  - 在 **salary** 屬性定義中，不設定 **set** ，並且 **get** 回傳值爲 **baseSalary** 加上 **benefit**

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

方法成員 (1)

- 方法是用來定義類別提供的特定功能
- 實體物件可透過 "." 來呼叫方法成員，但在方法成員名稱後需加上 () 來說明傳入的參數值
- 語法

```
存取修飾詞  回傳值  方法名稱 (傳入參數)
{
    方法內容程式區塊;
}
```

方法成員 (2)

- 用法

```
public void hello()
{
   Console.WriteLine("您好");
}
```

- 說明
  - **void** 表示沒有回傳值
  - **()** 內爲空白時，表示不需要傳作參數

實例探討 sample4-a4 (1)

- 程式功能
  - 建立 car 類別
  - 在 car 類別中定義 id 屬性及 showId 方法
- 程式內容

```
class car
{
   private string idField; //類別欄位 (field)


   public void showId() //類別方法 (method)
   {
    Console.WriteLine("車牌號碼：" + idField);
   }
```

實例探討 sample4-a4 (2)

- 程式內容

```
    public string id //類別屬性 (property)
    {
      get { return idField;  }
      set { idField = value; }
    }
  }
```

實例探討 sample4-a4 (3)

- 程式內容

```
static void Main(string[] args)
{
    car myCar = new car();
    myCar.id = "ABC-123";
    myCar.showId();

    Console.ReadLine();
}
```

參數傳遞與回傳值 (1)

- 類別的方法可以定義傳入的參數及回傳值，因此我們可以把需要運算的參數傳給方法，在方法中計算完畢後再回傳結果
- 語法

```
存取修飾詞 回傳值型別 方法名稱
    (參數1型別 參數1名稱, 參數2型別 參數2名稱, …)
{
    方法內容程式區塊;
}
```

參數傳遞與回傳值 (2)

- 用法

```
public int add(int a, int b)
{
   return a + b;
}
```

- 說明
  - 利用 **return** 關鍵字來傳遞要回傳的值
  - 注意型別須一致

實例探討 sample4-a5 (1)

- 程式功能
  - 建立 Caculator 類別
  - 定義二數相加 add 方法
- 程式內容

```
class Caculator
{
   public int add(int a, int b)
   {
     return a + b;
   }
}
```

實例探討 sample4-a5 (2)

- 程式內容

```
static void Main(string[] args)
{
    Caculator myCaculator = new Caculator();
    Console.WriteLine(myCaculator.add(10,20
    ));
}
```

課堂練習 sample4-b2

- 類別功能
  - 建立 graph 類別
  - 類別中含有 drawRectangle 方法，可傳入長及寬二整數值，並繪出該矩形
- 程式功能
  - 請使用者輸入長及寬
  - 呼叫該方法並將使用者輸入資料當爲傳入參數

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

靜態成員 (1)

- 類別中的靜態成員不需要產生實體物件即可直接存取引用
- 靜態成員必須使用 static 關鍵字進行宣告
- 引用靜態成員方式，直接使用類別名稱再利用 "." 連接靜態成員名稱即可
- Main 就是一個靜態方法成員

靜態成員 (2)

- 語法

  存取修飾詞 static 回傳值 方法名稱 (傳入參數)

  {

  方法內容程式區塊;

  }

- 用法
  - static void Main(string[] args)
- 說明
  - Main 方法爲靜態成員，傳入值爲一字串陣列，沒有回傳值、存取限制爲預設的 private

實例探討 sample4-a6 (1)

- 程式功能
  - 建立 Caculator 類別
  - 定義二數相加 add 靜態方法
- 程式內容

```
class Caculator
{
   public static int add(int a, int b)
   {
     return a + b;
   }
}
```

實例探討 sample4-a6 (2)

- 程式內容

```
static void Main(string[] args)
{
  //列印結果
  Console.WriteLine(Caculator.add(10,20));
}
```

課堂練習 sample4-b3

- 類別功能
  - 建立 console 類別
  - 類別中含有 query 靜態方法，可傳入一問句，並回傳使用者所鍵入的值
- 程式功能
  - 實際測試該類別
- 基本概念
  - query 靜態方法可傳入一個 string 參數，在方法中先列印該參數值至螢幕上，讀進使用者輸入 的資料後，回傳其結果

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

方法多載

- 方法多載允許我們將傳入參數不同的方法，定義成相同的名稱
- 當我們在定義二數相加的方法時，藉由方法多載可讓我們不需定義二個不同名稱的方法，以區隔為二個整數相加或二個浮點數相加

實例探討 sample4-a7 (1)

- 程式功能
  - 建立 Caculator 類別
  - 定義二數相加 add 靜態方法
  - 分別傳入整數及浮點數進行計算
- 程式內容

```
class Caculator
{
   //add for 整數相加..
   public static int add(int a, int b)
   {
    return a + b;
   }
```

實例探討 sample4-a7 (2)

- 程式內容

```
    //add for 浮點數相加..
    public static double add(double a, double b)
    {
      return a + b;
    }
  }
  static void Main(string[] args)
  {
    Console.WriteLine(Caculator.add(10,20));
    Console.WriteLine(Caculator.add(10.5,20));
  }
```

課堂練習 sample4-b4

- 類別功能
  - 定義 **math** 類別
  - 類別中含有 **max** 方法，能傳入二或三個整數，並回傳其中最大的一個
- 程式功能
  - 分別以二個參數及三個參數引用該方法
  - 列印回傳值
- 基本概念
  - 利用方法多載

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

建構式 (1)

- 建構式 (Constructor)
  - 建構式在類別實體物件建立前即會執行，用來初始化物件
  - 建構式的名稱一定要和類別名稱一樣
  - 建構式與方法相同允許多載
- 語法

```
存取修飾詞 類別名稱()
{
}
```

建構式 (2)

- 用法

```
public car()
{
   idField = "ABC-123";
}
public car(string id)
{
   idField = id;
}
```

實例探討 sample4-a8 (1)

- 程式功能
  - 建立 Caculator 類別
  - 定義多載建構式
  - 利用建構式初始化物件
- 程式內容

```
class car
{
   private string idField;
```

實例探討 sample4-a8 (2)

- 程式內容

```
    public car()
    {
      idField = "ABC-123";
    }
    public car(string id)
    {
      idField = id;
    }
    public void showId()
    {
      Console.WriteLine("車牌號碼：" +
    idField);
    }
}
```

實例探討 sample4-a8 (3)

- 程式內容

```
static void Main(string[] args)
{
    car myCar1 = new car();
    Console.Write("Car1 ");
    myCar1.showId();

    car myCar2 = new car("NTU-123");
    Console.Write("Car2 ");
    myCar2.showId();
}
```

課堂練習 sample4-b5

- 類別功能
  - 建立 **math** 類別
  - 類別中含有欄位值 **x** 及方法 **square**，該方法沒有傳入，回傳值爲欄位 **x** 的平方值
  - **x** 的值在建構式中預設爲 10
- 程式功能
  - 請輸用者輸入一整數
  - 列印該整數的平方值
  - 當輸入 的數小於 0 時，列印預設值 10 的平方值
- 基本概念
  - 在建構式中先初始化 **x** 的值

Outline

- 類別與物件
- 欄位與屬性
- 方法
- 靜態成員
- 方法多載
- 建構式
- 遞迴

遞迴

- 遞迴是一種本身呼叫自己的方法
- 利用遞迴可以提供較為簡潔的方法進行數學運算
- 遞迴程式中，必須撰寫令遞迴結束執行的程式碼

實例探討 sample4-a9 (1)

- 程式功能
  - 利用遞迴進行整數和運算
- 程式內容

```
private static int sum(int x)
{
   if(x==1) return 1;
   else return x + sum(x-1);
}
```

實例探討 sample4-a9 (2)

- 程式內容

```
static void Main(string[] args)
{
   Console.Write("請輸入一整數：");
   int x = int.Parse(Console.ReadLine());

  Console.WriteLine("1 加至 {0} 的和爲 {1}", x,
   sum(x));
}
```

課堂練習 sample4-b6

- 類別功能
  - 在預設 Class1 類別中定義 factorial 靜態方法
  - factorial 靜態方法，可傳入一整數參數 x，並回傳 x 的階層值
- 程式功能
  - 請使用者輸入 x 的值
  - 呼叫 factorial 靜態方法
  - 列印出 x 的階層值
- 基本概念
  - 利用遞迴來完成

課後練習 hw3

- 類別功能
  - 在預設 **Class1** 類別中定義 **fib** 靜態方法
  - **fib** 靜態方法，可傳入一整數參數 **x**，並回傳 **x** 的費氏數列值
- 程式功能
  - 請使用者輸入 **x** 的值
  - 呼叫 **fib** 靜態方法
  - 列印出 **x** 的費氏數列值
- 基本概念
  - $Fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-1) + fib(n-2) & \text{if } n >= 2 \end{cases}$
  - 費氏數列：0 1 1 2 3 5 8 13 21 34 …

進階練習 sample4-dl (1)

- 程式功能
  - 輸入年份及月份
  - 列印出該年該月份的月曆
- 紅利
  - 萬年曆 (自己判斷閏年)：20 分
  - 萬年曆 (使用類別)：15 分
- 公式
  - 逢四年閏一年
  - 逢一百年不閏
  - 逢四百年閏

進階練習 sample4-d1 (2)