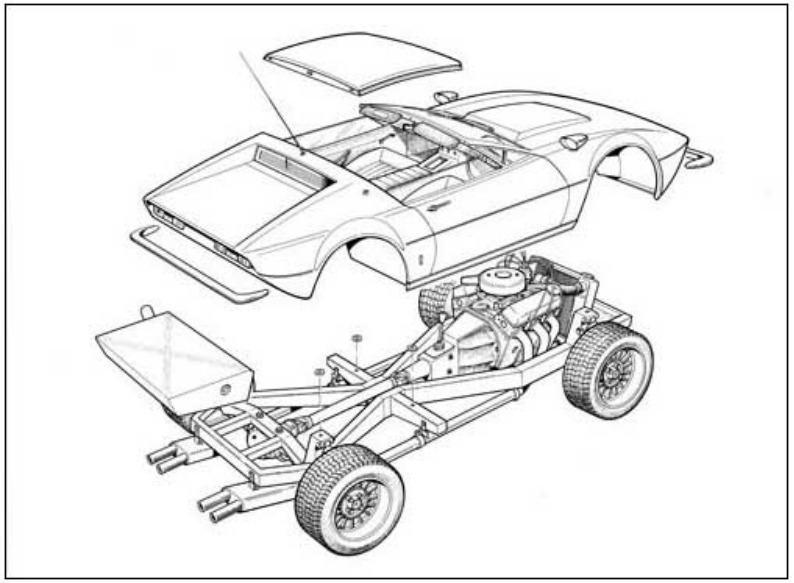
.NET 元件開發

Brian Lin

類別（Class）與物件（Object）

類別 Class



New

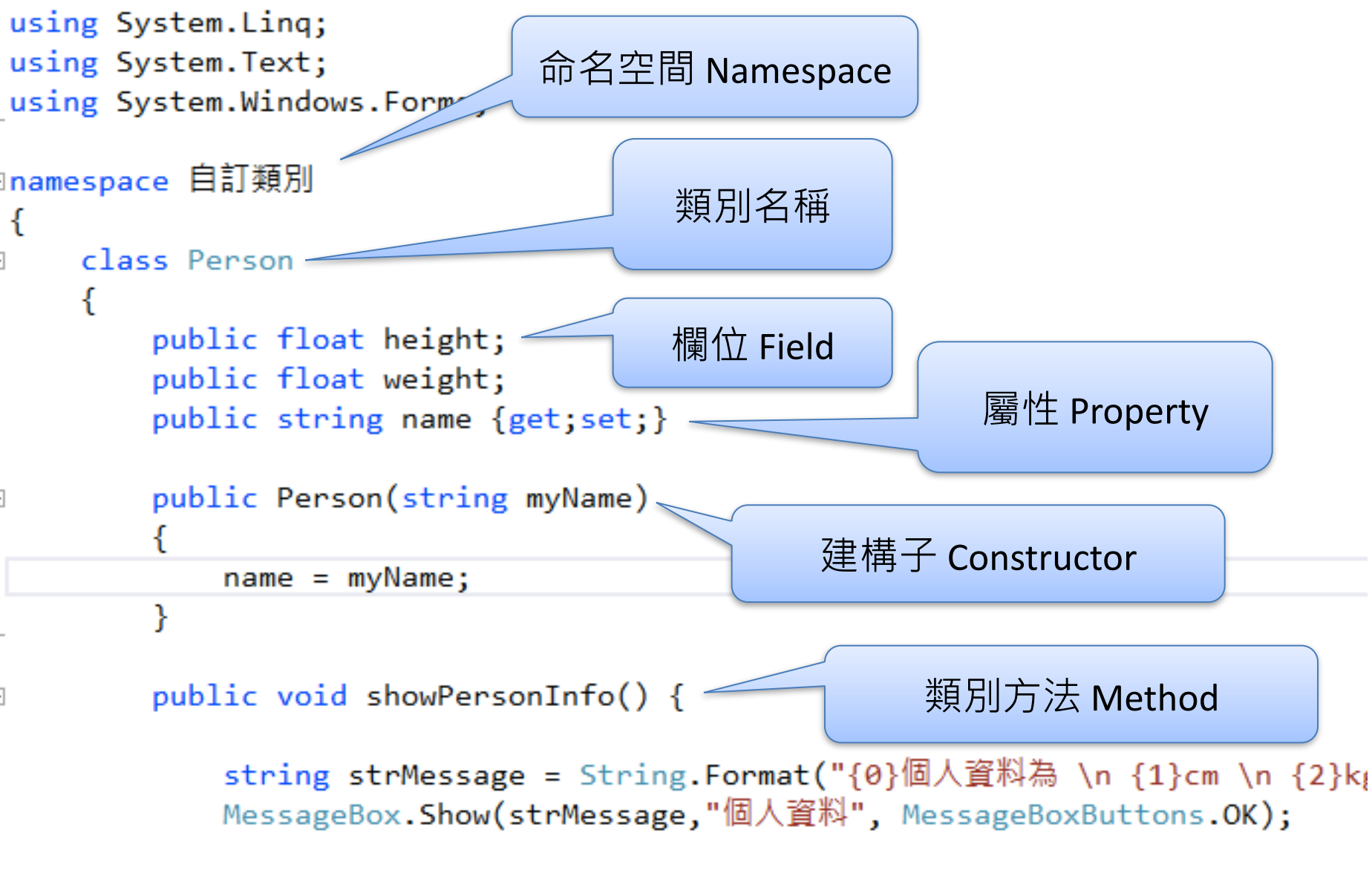
物件 object
Instance

物件，就是類別實體化之後的結果。

類別 Class

- 類別是物件導向開發（OOP）的重要基礎，而程式中最小的運作單元，就是物件（Object），也就是由類別（Class）產生的，提供並組成程式所需要的功能。
- 類別的構成：
  - ✓ C#的一切，都是基於類別。
  - ✓ 類別由方法（Method），欄位（Field），和屬性（Property）組成。
  - ✓ 類別俱有封裝的特性，利用公開（Public）或私有（Private），保護所屬成員的資料。
  - ✓ 利用建構子（Constructor），來初始化物件。但C#亦可不需要使用建構子來初始化。

自定類別 Class


using System.Linq;
using System.Text;
using System.Windows.Form
namespace 自訂類別
{
class Person
{
public float height;
public float weight;
public string name {get;set;}
public Person(string myName)
{
name = myName;
}
public void showPersonInfo() {
string strMessage = String.Format("{0}個人資料為 \n {1}cm \n {2}k
MessageBox.Show(strMessage,"個人資料", MessageBoxButtons.OK);
命名空間 Namespace
類別名稱
欄位 Field
屬性 Property
建構子 Constructor
類別方法 Method

命名空間 Namespace

- 命名空間提供一個有效的方法，去組織類別的程式碼，並且避免類別名稱重覆的問題，而且可以有效縮短執行方法的程式碼長度。

- 例如：
  - 在檔頭宣告
  - Using System.Windows.Forms;
  - 原來的程式碼
  - System.Windows.Forms.MessageBox.Show("Hello !!");
  - 可以改寫成
  - MessageBox.Show("Hello !!");

繼承 Inheritance

- 繼承是物件導向開發最主要的觀念，當需要創建建自己的類別時，可以繼承現有的類別，加以使用其功能，重覆利用已寫過的類別程式碼。
- 例如：

父類別：

```
Class Car { … }
```

繼承父類別：

```
Class Porsche : Car { … }
```

使用：

```
Car myCar = new Car();
Porsche myPorsche = new Porsche();
```

多型 Polymorphism

- 多型又可稱同名異式，在類別中建立同名的方法或屬性，來提供不同的功能或傳遞不同的參數，多型主要有兩種型態，多載和覆寫。

- 覆寫 (Override)：宣告相同方法名稱，也使用相同參數，取代原來方法功能。
- 多載：宣告相同方法名稱，但使用不同參數。

列舉 Enumeration

- 列舉(Enumeration)是一系列的整數常數的定義，方法是利用變數的命名，與所需的整數常數對應起來，提供程式碼來存取使用，宣告的關鍵字為enum。

  形式如下：

  enum Days { Sun = 7, Mon = 1, Tue = 2,Wed = 3, Thu = 4, Fri = 5, Sat =6 }

介面 Interface

- 介面(Interface)提供了一種程式設計的模式，利用相同的介面，在不同類別繼承並實作出不同功能的方法，但仍舊可以維持相同的方法或屬性的名稱。

- 介面僅能宣告成員，不能實體化(new)，亦無實作程式碼，必須由類別繼承來實作。
- 介面的成員可以是屬性，方法，事件。
- 類別雖只能繼承單一父類別，但可繼承多個介面。
- 類別繼承了某介面，則該介面的所有成員都必須實作。

泛型 Generics

- 泛型(Generics)的概念是，當我們在定義類別，介面或方法的時候，可以不用先指定其資料型態，之後實體化來使用時，再決定其型別。

```
// 泛型，未指定的資料型態，由 T 關鍵字代替;
public class MyGenericClass<T>
{
    public T myGenericProperty;
    public void MyMethod(T input) { }
}
```

結構

- 結構(Struct)是一種值型態(Value Type)的資料結構，定義方式跟類別有點類似，但不完全一樣，以下是結構的特點。

- 結構是值型態(Value Type)，類別是參考型態(Reference Type)。
- 結構不支援繼承，也就是不能繼承，也不能被繼承。
- 結構的建構子，一定要有參數。
- 結構的欄位，屬性，不能在宣告的時候賦值，可以在建構式的階段賦值。

委派 Delegate

- 委派（Delegate），就是將方法（Method）物件化，於是可以將物件化後的方法，代入方法的參數，然後傳遞呼叫使用。

- Delegate 實作的流程：
  1. 定義委派（Delegate），目標（Target）方法相關參數，需對應。
  2. 指定委派的目標方法（Target Method）。
  3. 調用委派執行動作。