深入探討 Object (C#)

Sam Xiao, Oct.8, 2017

Overview

OOP 以物件為基礎，本文探討什麼是 object？

Outline

What is Object ?

> 物件 Object
>
> 外界真實事物的抽象對應

物件導向的基本觀念，就是將電腦外界的事物加以對應成 `物件`，將該物件的 `資料` 和 `行為` 通通整合於該物件之內。如此，無論是從分析或寫程式的角度上看，都更容易方便。

抽象化解決問題

回想我們在國中/高中念物理、數學時，解決問題的方式：

- 如力學問題：我們會畫一個草圖描述事物之間的關係
- 如數學問題：我們會用 x, y 去描述事物之間的關係

我們並不是直接去解決問題，而是先對我們關心的部分抽象成草圖或 x, y，藉由其關係找出解決問題的蛛絲馬跡，進而解決問題。

> 將問題抽象化，本來就是人們習慣的解決問題方式

硬體早期因為運算能力較差，因此 programmer 必須配合電腦的思維去寫程式，但隨著硬體越來越進步，程式語言幾乎都往高度抽象化的方式演化，其目的就是讓電腦的程式語言與人類習慣的思考方式，能更緊密的結合。

> 雖然程式語言越來越高度抽象化，但由於 programmer 在學校就已經被訓練使用電腦的思維在寫程式，真的要改成人類的思維寫程式，反而改不過來

物件的五大特性

物件既然是 `外界真實事物的抽象對應`，就要能模擬世界中的真實事物，通常包含兩個部分：

1. `資料`：代表物件的性質
2. `行為`：代表物件能夠作用以及被作用的動作

> 為了 `模擬世界`，物件必須要有 `資料` 與 `行為` 兩大部分

此外，真實事物還有 `溝通`、`繁殖`、`出生` 與 `死亡` ...等能力，為了讓物件能夠模擬真實事物，必須有下 5 大特性：

物件必須有一個資料結構存放資料

真實事物根據不同 `需求` 角度來看，會有其不同的 `特性` 與 `內涵`，而且每個事物都不同，因此物件必須有個 `資料結構` 來存放不同物件的 `特性` 與 `內涵`。

使用 private 資料結構

```
class Zoo
{
    private List<Animal> animals = new List<Animals>();
}
```

> 不要以為在學校學的 `資料結構` 用不到，如 list 、dictionary、hashtable、tree .... 等，各種資料結構都有它的優點與缺點，要根據需求選擇適合的資料結構來存放資料

物件必須有狀態

真實事物會根據不同的 `狀態`，而有不同的行為，而且每個物件都不同，因此物件必須能儲存 `狀態`，並有不同的行為。

使用 public field

```
class Zoo
{
    public string Name;
}
```

使用 public property

```
class Zoo
{
    private string name;
    public string Name
    {
        get { return this.name }
        set { this.name = value }
    }
}
```

> 簡單狀態可以直接使用 public field，當需要進一步控制時再重構成 public property

物件必須要有行為

真實事物會根據不同的狀態，而有不同的 `行為`，而且每個物件都不同，因此物件必須有其各自的行為回應不同的狀態。

Lamp.cs

```
class Lamp
{
    private bool toggleStatus;

    public string showLight()
    {
        return (this.toggleStatus) ? "ON" : "OFF";
    }
}
```

`showLight()` 會因為 `toggleStatus` 的不同，有不同的回傳值。

> public method 或 private method 都可統稱為物件的 `行為`

物件必須可以被識別

真實事物雖然有相同種類的東西，但沒有完全一樣的東西，所以每個物件都不同，因此物件必須能被清楚地識別。

Student.cs

```
class Student
{
    public string Name;
}
```

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Student student1 = new Student();
        student1.Name = "John";

         Student student2 = new Student();
         student2.Name = "John";

         Console.WriteLine(object.Equals(student1, student2));
         // False
     }
}
```

儘管兩個物件的 `資料` 與 `行為` 都相同，仍然視為不同的物件。

> 若以電腦的角度思考，每個物件佔據了完全不同的記憶體空間，即使物件有相同的資料，仍然代表不同的物件。

物件必須可以被創造及消滅

真實事物有 `生` 也有 `死`，為了模擬真實事物，因此物件必須能被 `創造` 與 `消滅`。

Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Student student = new Student();
        student.Name = "John";
     }
}
```

當我們需要 `student` 物件時，可以使用 `new` 創造物件，如 C++，甚至可以手動用 `delete` 消滅物件，但現代程式語言多都有 garbage collection 機制，我們只要會 `new` 物件即可，`delete` 會由系統幫我們處理。

> 若以電腦的角度思考，存在於記憶體內的物件必須能被釋放，系統資源才能夠被再度使用。

> 物件 Object
>
> 外界真實事物的抽象對應。
>
> 必須有一個 `資料結構` 來儲存資料，並且有 `狀態` 與 `行為`。
>
> 可以被 `創造`、`消滅` 與 `識別`。

Conclusion

- 物件是對真實事物的抽象對應，讓我們可以將真實事物中，所關心的 `資料` 與 `行為` 部分，使用物件加以描述。