CH. 12

封裝與建構子 (ENCAPSULATION & CONSTRUCTOR)

HORAZON

C#程式設計

物件導向三大特性

1. **封裝** (Encapsulation)：保護資料，隱藏實作細節。
2. **繼承** (Inheritance)：程式碼共用與擴充。
3. **多型** (Polymorphism)：同一介面，多種實作。

本章將專注於 **封裝**。

為什麼要封裝？

在上一章的例子中，我們將變數設為 `public`：

```
class Player {
    public int hp;
}

Player p = new Player();
p.hp = -100; // ⚠️ 不合理的數值！
```

直接暴露資料 (Fields) 是危險的，可能導致：

- 資料被設為無效值 (如 HP < 0)。
- 外部程式過度依賴內部實作，導致日後難以修改。

存取修飾詞 (ACCESS MODIFIERS)

C# 透過修飾詞來控制成員的可見度：

- `public`：完全公開，任何人都能存取。
- `private`：**私有**，只有**類別內部**可以存取。(預設值)
- `protected`：只有繼承者可以存取 (下章詳談)。

```
class Player {
    private int hp; // 外部看不到了！

    public void SetHP(int value) {
        if (value < 0) value = 0; // 保護邏輯
        hp = value;
    }
}
```

屬性 (PROPERTY)

C# 提供了一種更優雅的語法來實現封裝，稱為 **屬性**。
它看起來像變數，但其實是**方法 (Getter & Setter)**。

```
class Player {
    private int _hp; // 私有變數 (backing field)

    public int HP // 公開屬性 (Property)
    {
        get { return _hp; }
        set {
            if (value < 0) value = 0;
            _hp = value;
        }
    }
}
```

使用時：`p.HP = -50;` (會自動變為 0)

自動實作屬性 (AUTO-IMPLEMENTED PROPERTY)

如果不需要特殊的保護邏輯，只是想寫成屬性：

```
class Player {
    // 編譯器會自動幫你建立一個看不見的 private 變數
    public string Name { get; set; }

    // 唯讀屬性 (只能在建構子中賦值)
    public int MaxHP { get; private set; }
}
```

這是 C# 開發中最常見的寫法！**請盡量使用屬性 (Property) 而非公開變數 (Public Field)**。

建構子 (CONSTRUCTOR)

當我們 `new` 一個物件時，常常需要**初始化**它的狀態。
建構子 是一個特殊的方法：

1. **名稱與類別相同**。
2. **沒有回傳型別** (連 void 都不寫)。
3. 在 `new` 的瞬間自動執行。

```
class Player {
    public string Name { get; set; }
    public int HP { get; set; }

    // 建構子
    public Player(string name, int hp) {
        Name = name;
        HP = hp;
    }
}
```

使用建構子

```
// 不需要一行一行設定屬性了
Player p1 = new Player("勇者", 100);
Player p2 = new Player("魔王", 999);
```

建構子多載 (Constructor Overloading)

你可以定義多個建構子，提供不同的初始化方式。

```
class Player {
    public Player() { // 無參數建構子（預設值）
        Name = "Unknown";
        HP = 10;
    }

    public Player(string name) {
        Name = name;
        HP = 10;
    }
}
```

物件初始化設定項 (OBJECT INITIALIZER)

C# 提供另一種初始化的語法糖，配合屬性使用非常方便：

```
class Item {
    public string Name { get; set; }
    public int Price { get; set; }
}

// 不需要定義建構子，也可以這樣寫：
Item potion = new Item
{
    Name = "紅藥水",
    Price = 50
};
```

總結

1. **封裝**：使用 `private` 隱藏細節，使用 `public` 開放介面。
2. **屬性 (Property)**：C# 特有的封裝語法，兼具安全與便利 (`{ get; set; }`)。
3. **建構子 (Constructor)**：用於物件初始化的特殊方法，名稱與類別相同。

掌握封裝，你的程式碼將更健壯、更安全！

下章預告：繼承與多型

如果我要做「戰士」、「法師」、「盜賊」，他們都有 HP 和 Name，我需要寫三個類別嗎？

下一章：**繼承 (Inheritance)**，讓我們不重複造輪子！