CH. 13

繼承與多型

(INHERITANCE & POLYMORPHISM)

為什麼需要繼承？

假設遊戲中有 `Warrior` (戰士) 與 `Mage` (法師)：

```
class Warrior {
    public string Name { get; set; }
    public int HP { get; set; }
    public void Attack() { ... }
    public void Slash() { ... } // 戰士特有
}

class Mage {
    public string Name { get; set; } // 重複！
    public int HP { get; set; }      // 重複！
    public void Attack() { ... }     // 重複！
    public void Fireball() { ... }   // 法師特有
}
```

問題： 大量重複的程式碼（`Name`、`HP`、`Attack`）。

繼承 (INHERITANCE)

我們可以提取出共同的部分，變成一個 **父類別** (Base Class)，例如 `Character`。
然後讓 `Warrior` 與 `Mage` 去 **繼承** (Inherit) 它。

```
// 父類別
class Character {
    public string Name { get; set; }
    public int HP { get; set; }
    public void Attack() { Console.WriteLine("攻擊！"); }
}

// 子類別 (Derived Class)
class Warrior : Character  // 使用冒號 : 代表繼承
{
    public void Slash() { Console.WriteLine("揮砍！"); }
}
```

`Warrior` 現在自動擁有了 `Name`, `HP` 和 `Attack()` ！

繼承的特性

1. **程式碼共用**：父類別寫一次，所有子類別都能用。
2. **層級關係**：建立 "Is-A" 的關係 (Warrior **is a** Character)。
3. **單一繼承**：在 C# 中，一個類別只能有一個父類別 (但可以實作多個介面)。

protected 修飾詞

- `public`：大家都能用。
- `private`：只有自己能用。
- `protected`：自己和 **子類別** 能用。

多型 (POLYMORPHISM)

多型的意思是「同一種方法，有不同的實作方式」。

父類別的 `Attack()` 可能只是普通的攻擊，但我們希望：

- 戰士的 `Attack()` 是用劍砍。
- 法師的 `Attack()` 是丟火球。

這時候我們需要 **覆寫** (Override)。

VIRTUAL 與 OVERRIDE

1. 父類別將方法標記為 `virtual` (虛擬的)，表示允許被修改。
2. 子類別使用 `override` (覆寫) 來重新定義該方法。

```
class Character {
    public virtual void Attack() {
        Console.WriteLine("普通攻擊");
    }
}

class Mage : Character {
    public override void Attack() {
        Console.WriteLine("發射火球！");
    }
}
```

多型的威力

多型最強大的地方在於：**使用父類別型別來操作子類別物件**。

```
// 雖然陣列型別是 Character
Character[] party = new Character[2];
party[0] = new Warrior();
party[1] = new Mage();

foreach (Character c in party)
{
    c.Attack();
    // 神奇的事情發生了！
    // c 是 Warrior 時，執行 Warrior 的 Attack
    // c 是 Mage 時，執行 Mage 的 Attack
}
```

這在遊戲開發中極度重要！(例如：管理所有敵人、所有道具)。

BASE 關鍵字

在子類別中，可以使用 `base` 來呼叫父類別的成員。

```
class Mage : Character {
    public override void Attack() {
        base.Attack(); // 先執行父類別的普通攻擊
        Console.WriteLine("附加燃燒傷害！");
    }

    // 建構子繼承
    public Mage(string name) : base(name) {
        // ...
    }
}
```

總結

- **繼承 (`:`)**：讓子類別擁有父類別的屬性與方法，減少重複。
- protected：開放給子類別使用的權限。
- **多型**：
  - virtual：父類別允許覆寫。
  - override：子類別進行覆寫。
- 利用多型，我們可以寫出更通用、更具擴充性的系統。

下章預告：集合與泛型

陣列（`Array`）長度固定很難用？
如果是 RPG 遊戲的背包，道具數量一直變來變去怎麼辦？

下一章：**List 與 Dictionary**，C# 最強大的資料結構！