

OO面相對象設計的五大原則

S.O.L.I.D

SRP(Single Responsibility Principle 單一職責原則)

規定每個類都應該有一個單一的功能，並且該功能應該由這個類完全封裝起來。

機車
↺
加速();↺ 減速();↺ 檢查汽油存量();↺ 加油();↺ 開大燈();↺ 洗車();↺

怎麼判斷到底這個責任屬不屬於這個class?

試著用以下的規則唸唸看：

可以使用「類別名稱」來「做某事」

可以使用「機車」來「加速」

可以使用「機車」來「減速」

可以使用「機車」來「檢查汽油存量」

可以使用「機車」來「加油」

可以使用「機車」來「開大燈」

可以使用「機車」來「洗車」

OCP(*Open-Closed Principle* 開閉原則)

- ▶ 為了擴充而開放
 - 模組的行為要能被擴充以增加新的功能
- ▶ 封閉修改
 - 增加新的程式碼，不要修改舊的已寫好的程式碼
- ▶ 達成方法
 - 抽象(Abstraction)、多型(Polymorphism)、繼承(Inheritance)
、介面(Interfaces)

LSP(Liskov Substitution Principle 里氏替換原則)

- ▶ 當子類別替換基礎類別，使軟體功能不受影響，此子類別才算真正被複用，子類別才能在基礎類別上增加新的行為。
- ▶ LSP清楚指引多型！基礎(base)類別適用的地方，子類別一定適用，故子類別須包含全部基礎類別介面。
- ▶ 違反LSP也違反OCP，因為要修改子類別的方法。
- ▶ 針對違反LSP設計時Refactoring方式，當classA錯誤繼承classB時
 - 建構新的抽象classC，作為2個具體classA, B的父類別
 - 重構為classB委派(Delegate) classA

ISP(Interface Segregation Principle 介面分割原則)

- ▶ 客戶端程式若依賴未使用到的介面，將造成介面污染/肥(FAT)介面，應被分解成幾群介面，每群服務一種客戶端
 - 客戶端程式間的依賴性應建立在最小的介面上
- ▶ ISP建議每個服務都有特定interface，依客戶型別分類，建立各種介面。
 - 維護程式時，現有類別或元件的介面往往會變動，迫使所有元件重新編譯及部署。
 - 應在現有物件加入新介面，而不是改變現有介面。

DIP(Dependence Inversion Principle 依賴反轉原則)