物件導向程式設計(Object-Oriented Programming)

[壹、 物件導向三大特性 1](#_Toc90242750)

[1. 封裝(Encapsulation) 1](#_Toc90242751)

[2. 繼承(Inheritance) 1](#_Toc90242752)

[3. 多型(Polymorphism) 2](#_Toc90242753)

[貳、 物件導向五大原則(SOLID) 2](#_Toc90242754)

[1. Single Responsibility Principle (SRP) 2](#_Toc90242755)

[2. Open-Closed Principle (OCP) 2](#_Toc90242756)

[3. Liskov Substitution Principle (LSP) 2](#_Toc90242757)

[4. Interface Segregation Principle (ISP) 2](#_Toc90242758)

[5. Dependency Inversion Principle (DIP) 3](#_Toc90242759)

1. 物件導向三大特性
2. 封裝(Encapsulation)

將資料和處理函數包裹成一個黑盒子，將實做細節隱藏起來封裝成物件，使用者只要知道物件如何使用，不需要知道原理。封裝有助於提高類和系統的安全性。

簡單講：**封裝使物件的設計者與物件的使用者分開。**

1. 繼承(Inheritance)

延伸現存類別之屬性或方法，已達成新的目的。

* 繼承的類別會包含原本類別全部的屬性或方法。
* 繼承的類別可新增額外的屬性或方法。
* 繼承的類別可覆寫或隱藏原本類別的屬性或方法。

父類別(Superclass)/基礎類別(Base Class): 被繼承的類別。

子類別(Subclass)/延伸類別(Derived Class): 繼承後新產生的類別。

兄弟類別(Sibling Classes):繼承同一個父類別的子類別。

1. 多型(Polymorphism)

在同一個操作介面下或方法，可以有很多不同的實作型態。

需要使用抽象方法。

1. 物件導向五大原則(SOLID)
2. Single Responsibility Principle (SRP)

單一職責原則，一個類只做一件事，不要為實現過多的功能，避免相同的職責分散到不同的類中，如果一個類功能過多，可能引起變化的原因也就越多，因為程式碼更加相依或耦合，導致難以維護，「牽一髮而動全身」。

1. Open-Closed Principle (OCP)

開放封閉原則，開放新增、封閉修改

1. Liskov Substitution Principle (LSP)

里氏替換原則，

1. Interface Segregation Principle (ISP)

介面隔離原則，

1. Dependency Inversion Principle (DIP)

依賴導致原則，