# 物件導向原則

## 開放關閉原則

* 類別應該開放，以便擴充。並且應該關閉，禁止修改。
* 對程式容易變動且會重覆出現的程式碼採用開閉原則，到處使用開閉原則，是一種浪費也不需要。

## 顛覆依賴守則(dependency inversion principle)

* 依賴抽象類別，不要依賴具項類別。
* 例如PizzaStore知道(依賴)很多不同口味的pizza，是不好的。應該建立一抽象的pizza，讓各種口味的pizza繼承，而PizzaStore只需知道抽象的Pizza。

# 工廠模式(Factory Pattern)

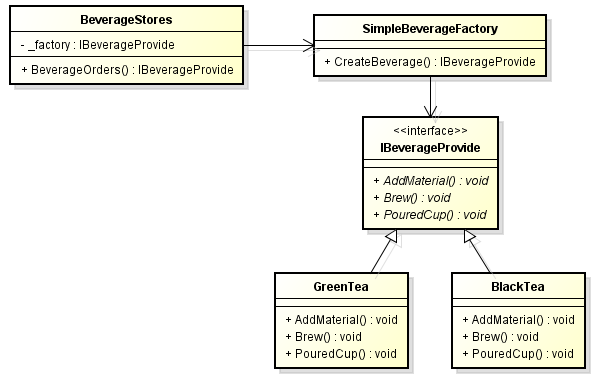
## 為何需要使用

1. 封裝建立物件的過程。
2. 將要被建立的物件，封裝到工廠中。主程式就只需要知道工廠，而不用知道有所有物件。(移植性較高)
3. 降低應用程式與具象類別之間的相依程度。
4. 讓開發者可以針對抽象類別寫程式，而非具象類別。

## 簡單工廠(Simple Factory Pattern)

包裝建立物件的程式碼，讓此程式碼可在不同的地方被應運，並減少重複的程式碼，以降低修改的難度。

如下圖，簡單工廠(SimpleBeverageFactory)，封裝了建立飲料的過程(CreateBeverage())，降低了BeverageStore與IBeverageProvide的耦合，而且簡單工廠可移到任意地方去建立飲料，不會只局限於BeverageStore。

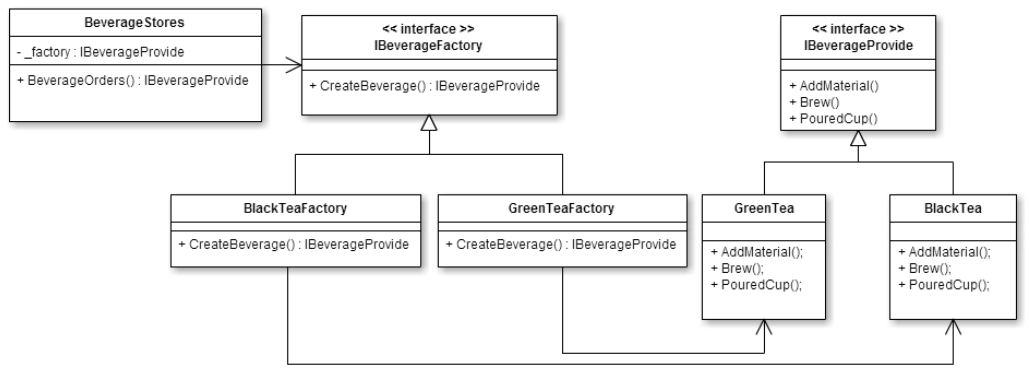


圖片來源: <https://dotblogs.com.tw/joysdw12/archive/2013/06/23/design-pattern-simple-factory-pattern.aspx>

## 工廠方法模式(Factory Method Pattern)

定義一個建立物件的介面，但由子類別實作如何建立物件(工廠方法)，也就是子類別只會覆寫父類別的工廠方法。

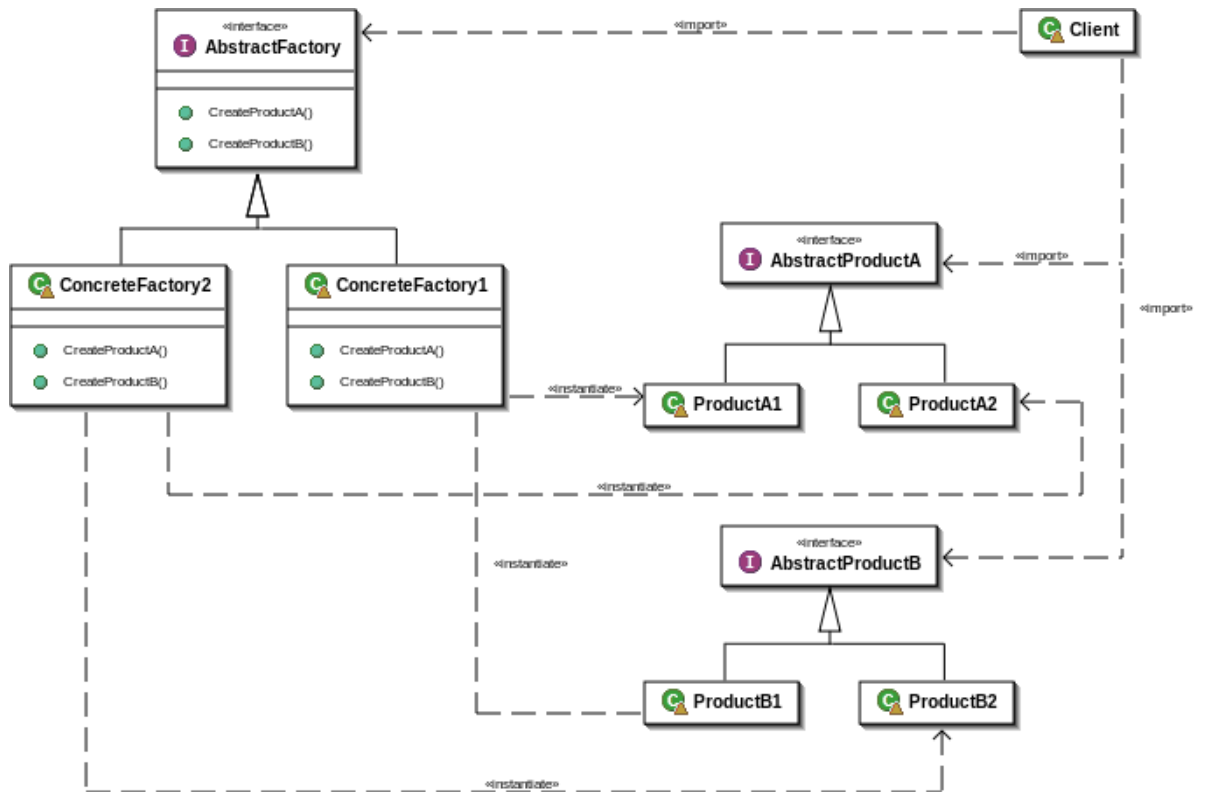
以下圖為例，BlackTeaFactory、GreenTeaFactory都繼承IBeverageFactory，並覆寫CreateBeverage()，兩者分別又知道BlackTea與GreenTea，如此可以達到分類的效果。紅茶不一定只有BlackTea這一種，可以有各式各樣的紅茶，而BlackTeaFactory可以知道所以類行的紅茶，將他們歸為一類，同理，各種綠茶也可被歸類到GreenTeaFactory。



圖片來源: <https://dotblogs.com.tw/joysdw12/2013/09/12/design-pattern-factory-method-pattern>

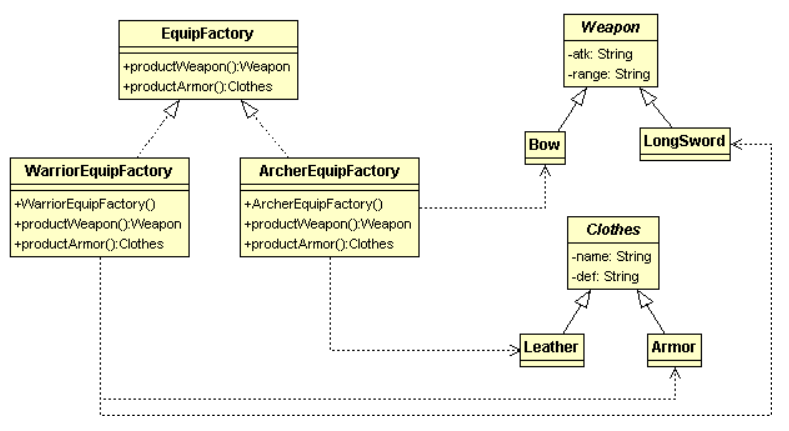
## 抽象工廠(Abstract Factory Pattern)

建立一個(工廠、原件家族)介面，將建立物件所需的多個原件結合在此介面中。實際建立物件的方式則交由子類別去完成。此模式之類別圖如下。



圖片來源: <https://zh.wikipedia.org/wiki/%E6%8A%BD%E8%B1%A1%E5%B7%A5%E5%8E%82>

下圖為抽象工廠的範例



圖片來源: <https://skyyen999.gitbooks.io/-study-design-pattern-in-java/content/abstractFactory1.html>

## 實際開發經驗

1. 誤用經驗: 在開發OrteryViewer中的Button時想套用此抽象工廠模式，因為程式中有多種不同的按鈕，而它們的”圖示”, ”路徑”接不相同。但其實”圖示”, ”路徑”都只是字串，並沒有多型的必要。如果套用抽象工廠模式，每新創一個按鈕就要為它創一個class，而只為了不同的”圖示”, ”路徑”，似乎太多餘，耗費開發時間。應該使需要將”圖示”, ”路徑”當作參數傳入建構值中即可。

或許真的需要用抽象工廠模式的時機是如上圖的情況(物件中的子物件有需要多型的函式)，equipment中的不同weapon，會有不同的攻擊力，所以需要覆寫attack()，這確實需要利用多型來完成，因此可以套用此模式。

# 命令模式(Command Pattern)

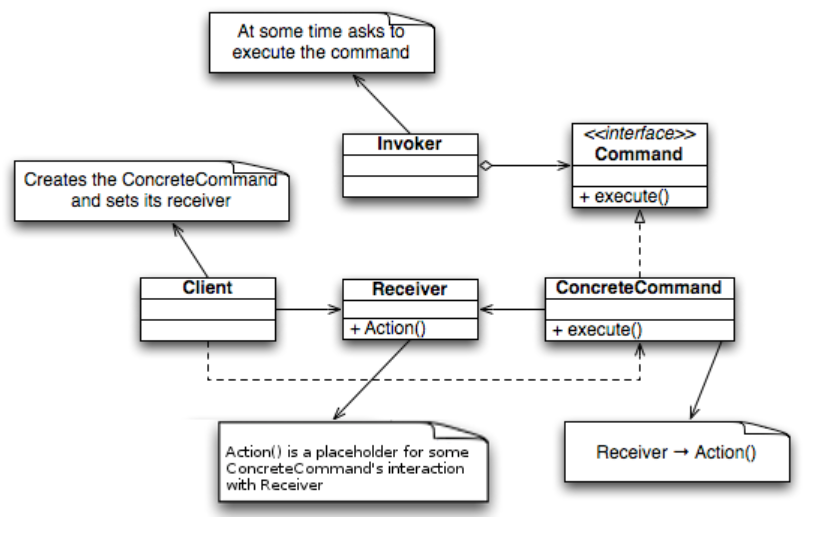
## 為何需要使用

1. 分離執行命令與觸發命令的物件。

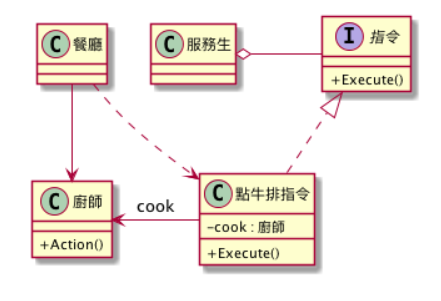
例: 若按鈕(觸發命令的物件)要加入主頁面(執行命令的物件)的某個命令，如果不使用命令模式，可能需要將整份主頁面傳到按鈕中，這樣按鈕就可以執行所有主頁面可以做的事，這樣就與當初設計的原意不同。所以可以利用命令模式，如此就只需將主頁面的某個指令包成命令物件，傳入按鈕中即可，亦可避免主頁面的其他功能被濫用。

1. 將物件實際的行動封裝成命令，即可(1)重覆使用、(2)還原這些命令。

## UML圖



圖片來源: <https://zh.wikipedia.org/wiki/%E5%91%BD%E4%BB%A4%E6%A8%A1%E5%BC%8F>



圖片來源: <https://notfalse.net/4/command-pattern>

## 實際開發經驗

1. 可套用之實際範例: 在OrteryViewer中，有兩種方式可以呼叫命令，有兩種方法可以達到此功能: (1)按畫面上的按鈕、(2)在瀏覽器鍵入API名稱。下列以放大鏡為例進行說明，開啟放大鏡時，還需要傳入類別Animation，也就是說要執行開啟放大鏡這個命令需要Magnifier跟Animation這兩個物件。若沒有將這指令以命令模式封裝，在任何需要開起放大鏡的類別中，都必須將這兩個物件傳入，這樣很麻煩，之後也不好修改，包成命令後就可以解決上述問題。