# 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。
* 對程式容易變動的地方採用開閉原則，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

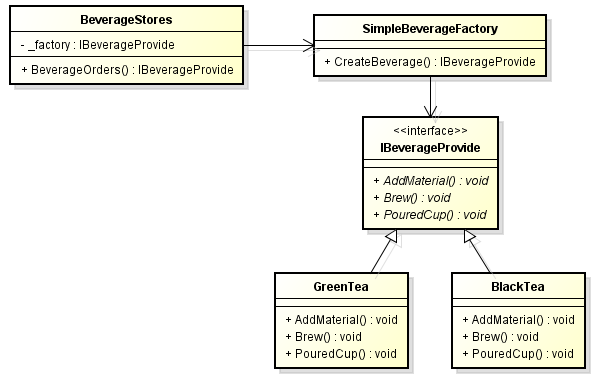
## 為何需要使用

1. 封裝建立物件的過程。
2. 降低應用程式與具象類別之間的相依程度。
3. 讓開發者可以珍對抽象類別寫程式，而非具象類別。

## 簡單工廠

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。

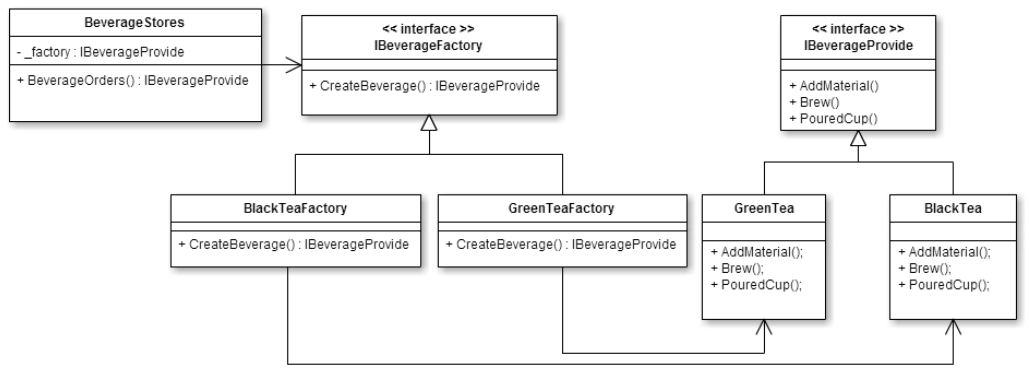


圖片來源: <https://dotblogs.com.tw/joysdw12/archive/2013/06/23/design-pattern-simple-factory-pattern.aspx>

## 工廠方法模式

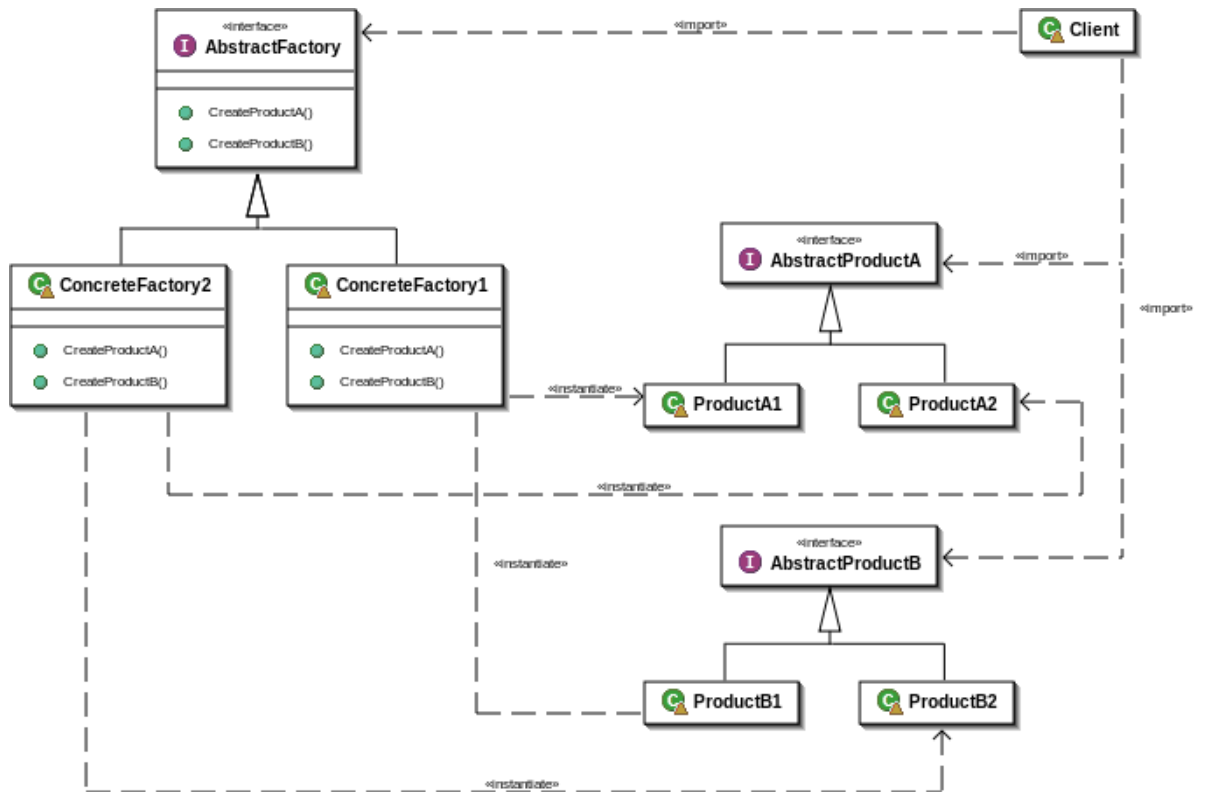
定義一個建立物件的介面，但由子類別實作如何建立物件(工廠方法)，也就是子類別只會覆寫父類別的工廠方法。

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。

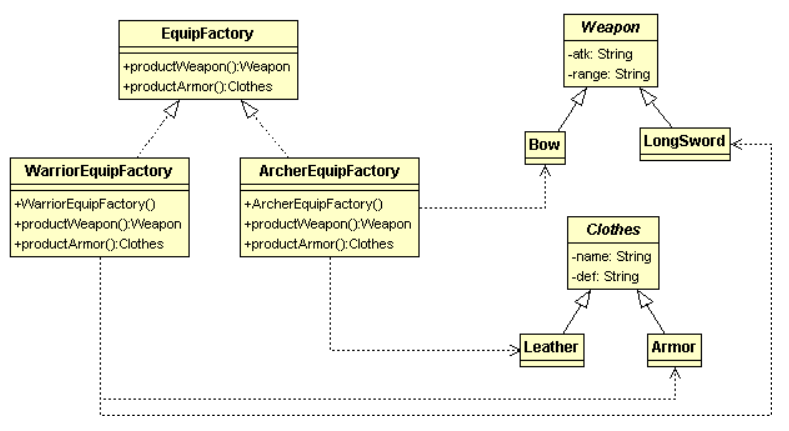


圖片來源: <https://dotblogs.com.tw/joysdw12/2013/09/12/design-pattern-factory-method-pattern>

## 抽象工廠



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



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