有意義的命名

- **讓名稱代表意圖**

好好選擇能夠代表其意圖的名稱，使得開發者或是閱讀者更容易了解和修改程式碼。

- **避免誤導**

拼寫與意圖相似的字詞就是好的命名，而使用與意圖不一致的拼寫就是誤導。

如 accountList 應使用 accountGroup。此處的 List 在程式設計師的理解內可能是某種資料型態，若該變數的資料型態不是 List，那就是一種誤導。

- **產生有意義的區別**

不要使用數列命名法（a1,a2,a3, … ,aN）

不要為了滿足編譯結果而故意使用錯誤的名稱 ex： clazz，klass

若無法一眼區分出 Product、ProductInfo、ProductData，你不應該感到羞恥~~，別難過~~。

因為 xxxInfo 以及 xxxData 已經泛濫到類似英文的冠詞 a、an 。

- **使用能唸出來的名稱**

命名的字詞應該是要能夠被唸出來的，人的腦袋被設計進化成有一部分負責處理字詞的語言區域，也有一定程度會關係到記憶。

如果沒能好好利用這一大塊的資源，著實是相當丟臉的事情。

- **使用可被搜尋的名字**

在這 IDE 盛行的年代，可不可以被搜尋關係到處理問題的速度。

單一字母的命名如 i、j、k 這種，在區域範圍無意義是可以被接受的，但如果是一個參考到比較大的範圍（scope），長名就不見得比短名差。

- **成員變數的字首**

類別以及函式已經盡可能的縮小，使得你應不需要加任何字首也該要知道成員或是方法隸屬的命名空間所代表的意義。

類別中揭露的資訊應該要是完整的詞彙，多餘且沒有意義的應被~~燒毀~~去除。

- **介面和實作**

介面（Interface）字首的 I 就免了吧，ex: IPizzaFactory …

看原始碼也知道這是一個介面， 使用的人也只需要知道這是一個披薩工廠，前綴什麼字母都是多餘的~~廢話~~。

但實作的類別可以在字尾加上 Imp、Impl 表示這是實作類別。

- **避免思維的轉換**

你的程式的讀者，不應該將你取的名稱在腦海中重新轉換成他們所熟悉的名稱，再重新理解你的程式碼。

一般的程式設計師跟專業的程式設計師的差別在於，專業的程式設計師了解

清楚明白才是正道~~→讀者怒吼不要跟我玩啞謎~~。

- **類別的命名**

正確的使用名詞，動詞~~走開~~應避免。

較無意義的詞彙 Data 、 Info 也應避免。

- **方法的命名**

正確的使用動詞、動詞片語命名。

建構子若有多載（Override），取得該物件時使用該物件的靜態方法（方法取名也需要符合該建構子的使用情境）回呼該建構方法，且該建構方法應使用 private 修飾。

ex: Customer clientCustomer = Customer.getRealAgeCustomer(26);

- **每個概念使用一種字詞**

專案中每個物件應該統一使用某個詞彙表達某個概念。

如 service 統一使用 get 當做方法前綴取得物件， dao 統一使用 query 查詢資料庫。

- **解決方案 / 問題 領域的命名**

非產業背景相關的程式工作清楚的使用電腦科學、演算法或是模式名稱表達概念，不要使用沒有意義的產業背景命名。

ex: WebServiceUrlProvider~~，看官你該不會不知道WebService是什麼吧？~~

若沒有可以使用的相關專業術語，請使用產業相關的術語。

ex: paymentDueDay，付款到期日。

- **添加有意義的上下文資訊**

方法中的區域變數若使用範圍過大，可以考慮新增一個類別（請清楚的命名）並將重複使用的區域變數加入至該物件的成員變數，清楚的給予命名空間。且透過該物件將原本參考到的方法切割成更小的函式，將使原演算法更加整潔。