林瑋鴻 403530045 資管三

* List at least 5 ways in which refactoring can be performed.

1. Extract Method

將程式碼截取出放進一個獨立的函式中，讓原先的程式碼改呼叫此函式進行作業。

(如果沒有經過重構的動作將相同性質的程式碼抽出。除了本身在程式碼閱讀上相當的繁瑣外，在後期專案的維護上反而會花費更大的成本維護。)

2. Replace Temp with Query

把暫時變數轉換成一個查詢方法(Query method)，便可以讓你的程式碼更加的附有彈性及閱讀性。

定義：「將運算式提煉到一個獨立函式中，將這個暫時變數的所有被引用點轉換為對新函式的呼叫」。

1. Replace Parameter with Method

以函式取代過長的參數，目的在於讓函式內部對參數的使用替換為對新函式的呼叫。

4. 重複代碼的提煉

重複代碼是重構收效最大的手法之一，進行這項重構的原因不需要多說。它有很多很明顯的好處，比如總代碼量大大減少，維護方便，代碼條理更加清晰易讀。

它的重點就在於尋找代碼當中完成某項子功能的重複代碼。

5. 冗長方法的分割

分割一個大方法時，大部分都是針對其中的一些子功能分割，因此我們需要給每一個子功能起一個恰到好處的方法名，這很重要。可以說，能否給方法起一個好名字，有時候能體現出一個程序猿的大致水準。

往往在我們提煉重複代碼的過程中，就不知不覺的完成了對某一個超長方法的分割。

6. 去掉一次性的臨時變量

在程序當中，一次性的臨時變量不僅是對性能上小小的摧殘，更是對代碼可讀性的褻瀆。因此我們有必要對一些一次性的臨時變量進行手術。

7. 提取類或繼承體系中的常量

這項重構的目的是為了消除一些魔數或者是字符串常量等等，魔數所帶來的弊端自不用說，它會讓人對程序的意圖產生迷惑。而對於字符串等類型的常量的消除，更多的好處在於維護時的方便。因為我們只需要修改一個常量，就可以完成對程序中所有使用該常量的代碼的修改。

與此類情況類似並且最常見的，就是Action基類中，對於INPUT、LIST、SUCCESS等這些常量的提取。

* What are the advantages and potential difficulties of test-first development?

優點：

1. 測試範例是在對功能進行測試。在寫代碼之前先寫測試用例，可以對我們編寫代碼提供指導性的参考。防止我們漏掉一些功能。
2. Test First 可以確保測試的高涵蓋率，因為總是先有測試的程式，才有實際的成品
3. 你會更加站在用戶的角度去看你將要完成的產品，你要盡可能想到用戶所有進行的操作。而不是從程序員的角度想用戶應該會如何去使用我們的產品。

潛在困難：

1. 我們不清楚軟體元件最終設計出的完整品模樣，且不曉得是否能夠實做出需求的功能，所以有時候即使經過”測試先開發”的實驗及重新設計後，很有可能仍然無法做出我們真正要的完整品設計。
2. 代碼量可能暴增，測試代碼通常會是系統代碼的兩倍或更多。
3. 它需要我們有設計完備的測試用例的能力（這項能力是長期理論與實踐的結合體），否則你將會吃了虧編寫了一大堆測試用例，卻沒測到點子上。

Reference：

重構：

<https://dotblogs.com.tw/heckit/2011/06/06/27234> --1

<https://dotblogs.com.tw/heckit/2011/06/08/27310> --2

<https://dotblogs.com.tw/heckit/archive/2011/06/04/27123.aspx> --3

<https://read01.com/EBA2Ga.html> --4、5、6

<http://tutorials.jenkov.com/java-unit-testing/test-first-development.html#test-first-is-hard-during-experimental-programming>

<https://ingramchen.io/blog/2014/04/how-i-do-test-driven-development.html>

–test-first development