搞笑談軟工

敏捷開發，設計模式，精實開發，Scrum，軟體設計，軟體架構

標籤

2012年1月2日 星期一

亂談軟體設計（4）：Liskov Substitution Principle

January 02 09:34~11:30

這一集輪到了 The Liskov Substitution Principle，內容請參考 Agile Software Development 這本第 111-125 頁：

LSP: The Liskov Substitution Principle

Subtypes must be substitutable for their base types.

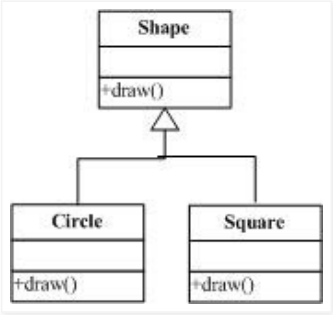
首先說明一下，Liskov 指的是 Barbara Liskov 女士，這個原則是她在 1988 年所提出來的，在 Design by Contract 中（請參考 Object-Oriented Software Construction, 2nd 這本書），有一個規則叫做 Subcontracting rule 所要表達的意義與 LSP 是一樣的。「*Subtypes must be substitutable for their base types*」翻成白話文的意思就是說：「如果你的程式有採用繼承，或是定義了類似 Java 的 interface 然後提供若干個該 interface 的實做（implementation），則在你的系統中，只要是 base types（父類別或是 interfaces）出現的地方，都可以用 subtypes （子類別）或是該 interface 的實做來取代，而不會破壞程式原有的行為」

這樣解釋可能還是不太容易理解，看一張圖先：

- 軟體設計 (147)
- 最新課程 (78)
- 創業 (80)
- 測試 (93)
- 需求 (15)
- 課程實錄 (22)
- 學習 (75)
- 貓 (15)
- 還少一本書 (34)
- agile (615)
- C. C. Agile (33)
- DDD (11)
- DevOps (3)
- Eclipse (6)
- exception handling (83)
- HCI (40)
- Implementation Patterns (5)
- Kanban (76)
- Mobile (2)
- Patterns (190)
- Programming (19)
- Refactoring (58)
- Scrum (448)
- TDD/BDD (37)

網誌存檔

- ► 2019 (33)
- ► 2018 (28)
- ► 2017 (142)
- ► 2016 (301)
- ► 2015 (366)
- ► 2014 (365)
- ► 2013 (365)
- ▼ 2012 (373)
  - ► 十二月 (31)
  - ► 十一月 (30)
  - ► 十月 (31)
  - ► 九月 (31)
  - ► 八月 (31)
  - ► 七月 (32)
  - ► 六月 (31)
  - ► 五月 (31)
  - ► 四月 (30)



這是一個很常見的繪圖系統設計，假設鄉民們要設計一個繪圖系統，可以畫圓（Circle）與正方形等形狀，於是鄉民們設計了一個父類別叫做 Shape（形狀），然後讓 Circle 與 Square 都繼承自 Shape。為了把這些圖形畫在螢幕上，鄉民們的程式中會有長成類似下圖的程式碼：

```
Shape myShape;
...

myShape = new Cirlce();
myShape.draw();
...

myShape = new Square();
myShape.draw();
```

程式中宣告 Shape 型態的 myShape 物件（instance），然後在 runtime（程式實行的時候）用這個 myShape 物件指到不同的子類別，然後再呼叫 draw method 就可以把各種不同的圖形畫在螢幕上。

上述作法其實是很基本的物件導向程式設計技術（polymorphism and dynamic binding）。這和 LSP 有什麼關係？因為上面這個設計符合 LSP，所以才能夠讓 Shape 物件（父類別）所出現的地方使用子類別（Circle 與 Square）取代。鄉民們看了 Robert C.

Martin 對於 LSP 另一種解釋就比較容易理解：

LSP: The Liskov Substitution Principle

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

***

不知道鄉民們有沒有想過這樣一個問題：「為什麼寫程式的時候，可以宣告父類別的物件，然後在 runtime 的時候讓這個父類別的物件指到子類別的實例（instance）然後程式還是可以正確執行？」鄉民們想一下上面那段程式碼中， draw 這個 method 在父類別（Shape）中的「行為（behavior）」被定義為把圖形物件畫在畫面上。之所以可以做到「Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.」是因為子類別（Circle 與 Square）的 draw method 在實做的時候也遵循了原本父類別（Shape）對於 draw 的行為定義（把自己畫在畫面上）。如果子類別（Circle 與 Square）沒有遵循了原本父類別（Shape）對於 draw 的行為定義，那麼就不無法達到 LSP 的要求。

上面這段話需要思考一下，請鄉民們多看幾次。

那麼沒有達到 LSP 會怎樣？如果程式沒有達到 LSP 則程式的行為將變得「不可預測」，換句話說可能產生不可預知且不容易察覺的 bugs。舉個例子，如果 Circle 物件的 draw method 沒以把自己畫在畫面上，而是把自己存到檔案中，或是輸出到印表機中，或是輸出到畫面的時候使用的座標和 父類別（Shape）所定義的座標系統不同。這樣的話，當鄉民們從 source code 看起來程式都是正常的，但是在真正執行的時候程式的行為卻不是自己所預期的（一位 Circle 沒有被輸出到畫面上，或是有輸出到畫面上但是位置卻不正確）。

***

看到這邊假設鄉民們知道 LSP 的意義以及違反 LSP 的後果，最後還剩下一個重要的問題：

如何定義程式的行為？

要達到 LSP 就表示子類別或是實做的行為要和父類別或是介面所定義的行為一致，問題是，平常宣告父類別或是介面的時候，只宣告 method 的 signature 而已啊，一般的程式語言並沒有提供什麼方法讓開發人員可以宣告「行為」啊？

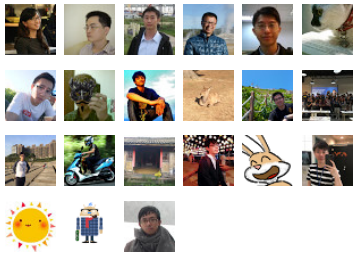
亂談軟體設計（4）：Liskov Substitution Principle

Scrum 是什麼（3）：三種補充文件

► 2011 (120)
► 2010 (104)
► 2009 (31)
► 2008 (6)
► 2007 (4)

追蹤者

追蹤者 **(328)** 下一頁



追蹤

最新回應

- 這文章的圖都看不到了 - 5/16/2019 - Pajace
- 重構系統永遠比砍掉重練好，遇過最大問題是:技術主管沒有能力/不想重構，亦沒有足夠經驗/知識分析問題所... - 4/28/2019 - Jason
- Martin Fowler 認為即使是 architect, 都應該寫程式，否則只是空談。http... - 4/28/2019 - Jason
- 我會解釋成 Hard Code 就是筆刷大小不能改 XD - 4/28/2019 - KK
- 身為一個初學程式的人而言...起頭總是痛苦的 XD 畢竟要從不會寫程式→能看懂程式→能改程式→自己可以... - 4/24/2019 - Unknown

熱門文章

羅技Logitech R500 雷射簡報筆開箱
June 22 12:51~14:16 需求分析 因為工作的關係Teddy經常需要使用簡報筆，雜牌的不算，光是羅技的簡報筆Teddy就買過以下四種（R500是最新上市產品，也是今天介紹的主角）： ▼畫面節錄自羅技官網 Teddy心中理想的簡報筆應該有以下幾個功能： 重量輕且...

不要只聽半套

沒錯，這就是為什麼 LSP 容易被忽略的地方。一般的物件導向程式語言（甚至是 C 語言）讓開發人員可以輕易做到「polymorphism and dynamic binding」但是卻沒有提供定義程式「行為」的機制。所以，Bertrand Meyer 才會提倡所謂的「Design by Contract（參考 Object-Oriented Software Construction, 2nd）」，建議開發人員幫每一個 method 定義 preconditions, postconditions 以及定義 class invariants 來規範程式的行為。有興趣深入了解的鄉民可以參考一下 Design by Contract 或是 Agile Software Development 這本書的 117-125。

***

還記得 GoF 在 design patterns 書中開宗明義所說的：

program to an interface，not an implementation

上面這個原則要能夠成立，interface 的實做就必須要遵守定義 interface 的人所期待的行為（因為 client 端只會透過 interface 看這個事件，並無法得知 runtime 時這個 interface 會被指派到那一個實做）。否則，就算是你的設計達到「program to an interface，not an implementation」也會因為沒有遵守 LSP 而破功。

***

友藏內心獨白：怎麼有種愈來越不好寫的預感。

讚 25 人說這個讚。趕快註冊來看看朋友對哪些內容按讚。

張貼者： Teddy Chen 於 上午11:31

標籤： 軟體設計

5 則留言:

匿名 2014年1月30日 下午6:39

老師你好, 我是剛學OOP的香港學生, 我想問一下這個情況是不是就違反了LSP的原則

```
public class Rectangle{
protected int width;
protected int height;
```

May 28 23:11~23:54 ▲這句怎麼都沒人聽進去 XDD 故事一 有一個公司派了整個團隊來泰迪軟體上 Scrum敏捷方法實作班，回公司之後團隊跑了一陣子Scrum。有一天因為產品出了包，非常緊急，Product Owner希望團隊加班趕緊把這個問題解決。沒想到，團隊...

Java的try、catch、finally（1）：Java SE 7之前

Nov. 06 16:22~17:47 Java SE 7之前 寫了這麼多篇Java例外處理的文章，只談過checked與unchecked exception的差別，還沒有正式介紹過Java的例外處理機制。鄉民們可能會想：「Java的例外處理不就是try、catch...

什麼是物件導向（2）：Object, Class, Instance

January 21 22:58~January 22 00:10 鄉民們看到這篇的日子應該是大年初一，先跟大家說聲新年快樂。 這次要介紹在物件導向技術中三個經常容易搞混的名詞，分別是object、class、instance。首先看一下Object-Oriented S...

一個Scrum各自表述

May 24 00:40~01:27 ▲每個component team都在跑Scrum，啊不就好棒棒…Orz 沒事不要找Teddy聊天XD 去年在某個演講場合，演講結束後某位鄉民找Teddy聊天…. 鄉民：Teddy你好，我是你部落格的忠實讀者。 Teddy：你好、你好，謝謝...

Top-down和Bottom-up設計方法

Mar. 12 08:40~10:10 Top-down（由上而下）和Bottom-up（由下而上）是兩種設計與解決問題的技巧。前者對問題先有一個整體的概念，然後再逐步加上設計細節，最後讓整體的輪廓越來越清楚。後者則是先將解決問題可能所需的基本元件、方案給準備好，然後再...

```
public Rectangle(){width = 0;height=0;}
public void setHeight(int height){this.height = height;}
public void setWidth(int width){this.width = width;}
public int getArea(){return width*height;}
}

public class Square extends Rectangle{
public Square = {super()};
public void setHeight(int height){this.height = height;this.width=height}
public void setWidth(int width){this.width = width;this.height=width}
public int getArea(){return width*height;}
}
```

回覆

回覆

next 2014年2月4日 下午8:56

如果我用你的Square class, setHeight()和setWidth()對我來說很不make sense.
所以明顯違反了LSP。（not well-designed inheritance)。

Teddy Chen 2014年2月4日 下午9:14

我覺得 next 說的很對，是違反了LSP。Joseph Chong你這個例子我在某本書看過，可能是 Object-Oriented Software Construction, 2nd 還是 Agile Software Development, Principles, Patterns, and Practices。LSP 又叫作 subcontracting，Square違反了Rectangle的contract (沒有遵守 Rectangle 的 postconditions)，所以違反了LSP。

Sakuya Izayoi 2014年3月3日 上午4:22

我也是學生(嗯)。請教一下，如果我硬要有Square的話，那麼是不是唯有delegate了？至於不能用Rectangle s = new Square();這問題就用一個共通的interface解決？

```
public interface Shape { public int getArea(); }

public Rectangle implements Shape { /* 不變，略 */ }

public Square implements Shape {
   private Rectangle rectangle = new Rectangle();
   public void setSide(int side){
      rectangle.setHeight(side);
      rectangle.setWidth(side);
   }
   public int getArea(){
      return rectangle.getArea();
   }
}
```

回覆

GJ J 2017年10月24日 下午11:35

感謝分享，這樣之後在寫虛擬函式的時候就有sense了。

回覆

輸入您的留言...

發表留言的身分： tlyau62@gmail ▼ 登出

發佈 預覽 通知我

這篇文章的連結

建立連結

較新的文章 首頁 較舊的文章

訂閱： 張貼留言 (Atom)

分析