切換導航 Farseerfc的小窩

- 繁體
- <u>简体</u>
- **English** 日本語

- <u>▲ About</u>
- **≜** Links
- About
- <u>Links</u>
- \Box
- □ Import
- □ Life
- <u>□ Tech</u>

- □ Import
- <u>Life</u>
- <u>Tech</u>
- Q

搜索

- 搜索
- #■ <u>歸檔</u>
- *y*

Program Development in Java Preface

從 farseerfc.wordpress.com 導入

程序開發原理

——抽象、規格與面向對象設計

Barbara Liskov、John Guttag 著

楊嘉晨 等譯

關於翻譯風格:

多年來閱讀計算機類的著作及譯作,感覺總體的困 難在於一大堆沒有標準譯名的技術術語。由於通行 於工業界和學術界的還是英文原名和術語,我決定 保留大量的英文術語。這樣的翻譯風格借鑑於臺灣 著名的譯者和作者侯捷先生。對於譯與不譯的權 衡,主要考慮閱讀的流暢,以及讀者的理解能力, 或許難免帶有一些主觀色彩。

前言 Preface

構建產品級質量的程序——可以在很長一段時間內使用的程序——衆所周知是極其困難的。本書的目標就是改善程序員解決這項任務的效率。我希望讀者在閱讀本書之後成爲一名好程序員。我相信本書的成功在於改善編程技巧,因爲我的學生告訴我這已經發生在他們身上。

怎麼纔算是一名好程序員?是產生整個程序產品的效率。關鍵是要在每一階段減少浪費掉的努力。解決的方法包括:在開始編寫代碼之前就仔細考慮你的實現方案,通過未雨綢繆的方法來編寫代碼,使用嚴格的測試在早期發現錯誤,以及仔細注意模塊化編程,這樣當錯誤出現時,只需要改動極少數代碼就可以修正整個程序。本書涉及所有這些領域的技術。

模塊化編程(Modularity)是編寫好程序的關鍵。把程序分解成許多小模塊,每一個模塊通過良好定義的狹窄接口和別的模塊交互作用(interact)。有了模塊化,可以修正一部分程序中的錯誤而不考慮程序的其他部分,而且可以僅僅理解一部分程序而不必理解整個程序。沒有模塊化,程序是一大堆有着錯綜複雜的相互關係的部分的拼湊。很難去領悟和修改

這樣一個程序,同樣也很難讓它正常工作。

因此本書的重點在於創建模塊化的程序:怎樣把程序組織成一系列精心挑選的模塊。本書認爲模塊化就是抽象(abstraction)。每一個模塊意味着一個抽象,比如說指引一系列文檔中的關鍵字的目錄,或者在文檔中使用目錄來查找匹配某個問題的文檔的過程。着重強調面向對象編程思想——在程序中使用數據抽象和對象的思想。

這本書使用Java作爲它的編程示例的語言。我們沒有假定讀者已經熟悉Java。儘管可能沒什麼價值,但是本書中的思想是語言無關的,並且可以在任何語言的編程中使用。

怎樣使用這本書? How Can the Book Be Used

本書《程序開發原理》有兩種使用方法。其一是作 爲課本教材,講述如何用面向對象的方法來設計和 實現複雜系統;其二是編程專家使用,幫助他們改 善編程技能,增進他們的關於模塊化和Object-Oriented(面向對象)設計的知識。 作爲教材使用時,本書一般作爲第二或第三門程序 設計課程。我們已經在MIT使用本書很多年,給大一 大二的本科生教授第二門編程課。在這一階段,學 生們已經知道怎樣編寫小程序。課程在兩方面利用 這一點:讓學生更仔細地思考小程序,以及教他們 如何利用小程序作爲組件構建大型程序。這本書也 可以在專業(如軟件工程)後期教學中使用。

建立在本書基礎上的課程適合於所有計算機科學專業。儘管許多學生可能永遠不會成爲真正的大型程序的設計師,他們可以在開發部門工作,在那兒他們負責設計和實現能與整個結構耦合的子系統。模塊化設計的子系統是這種任務中心,這對那些從事大型程序設計任務的人來說也同樣重要。

這本書講什麼?What Is This Book About

通觀全篇三分之二的書致力於討論在構建獨立的程 序模塊時產生的問題,剩下的部分討論怎樣運用這 些模塊構建大型程序。

程序模塊Program Modules

這一部分的書集中討論抽象機制(abstraction

mechanism)。它討論procedure(子程序)和 exception(異常),數據抽象,遍歷(iteration)抽 象,數據抽象系列(family)以及多態(polymorphic) 抽象。

在對抽象的討論中,三個步驟是重要的。首先是決定被抽象的東西到底是什麼:它提供給它的用戶哪些行為。創造抽象是設計的關鍵,因此本書討論如何在衆多選擇中挑選,以及怎樣才能創造出好的抽象。

第二步是通過爲一個抽象制定一個規格 (specification)來獲取它的意義。如果沒有一些描述,一個抽象就會含糊不清,而變得沒有使用價值。specification則提供了需要的描述。本書定義了一種specification的格式,討論了一份好的specification應有的屬性,並且提供了許多示例。

第三步是實現抽象。本書討論怎樣設計一份實現, 以及在簡潔性和執行性能之間怎樣權衡利弊。書中 強調封裝(encapsulation)的重要性以及在一份實現 中履行規格中定義的行為的重要性。書中同樣提供 一些技術——尤其是不變式斷言(representation invariant)和抽象函數(abstraction function)——來 幫助讀者理解代碼和它的原因。不變式斷言和抽象 函數都實現到儘可能的程度,這對於除錯和調試很 有用。

關於類型層次(type hierarchy)的材料注重討論使用它作為抽象的技術——一種把相關聯的一組數據抽象歸入同一系列的技術。這裏很重要的一點是,是否應當將一個類型作爲另一個類型的子類。本書定義了替換原則——通過比較子類和父類的specification,來決定是否建立子類關係的方法[1]。

本書同樣涉及除錯和調試。書中討論怎樣得到足夠數量的測試情況,來準備通過黑箱和白箱測試,它同樣強調了複查(regression)測試的重要性。

編寫大型程序 Programming in the Large

本書的其後部分講解怎樣用模塊化的方法設計和實現大型程序。它建立在前文有關abstraction和 specification的材料的基礎之上。

編寫大型程序涵蓋四個主要議題。首先講解需求分析——怎樣才能領悟程序中需要什麼。本書討論怎樣

實施需求分析,也討論書寫產生的需求規格的方式,通過使用一種描述程序的抽象階段的數據模型。使用這種模型將產生一份更爲正式的specification,同時它也使需求檢查更加嚴格,這樣可以更好的領悟需求。

編寫大型程序的第二項議題是程序設計,這通常是一個循序漸進的過程。設計過程圍繞構建有用的抽象來組織,這些抽象作爲整個程序之中理想的構建組建。這些抽象在設計時被仔細的編寫規格,這樣當程序實現時,那些實現抽象的模塊可以獨立地開發。這種設計使用設計筆記編寫文檔,包括描述整個程序結構的模塊間依賴性的圖示。

第三項議題是實現和測試。本書討論了前置設計分析對於實現的必要性,以及怎樣進行設計複審。它同樣討論了設計和實現的順序。這一部分比較了自頂而下與自底而上的組織方式,討論如何使用驅動程序和佔位程序[2](stub),並且強調了制定一個事先的順序策略的必要性,以滿足開發組織和客戶的需求。

本書以一章設計模式(design pattern)結束。一些模式在前面的章節介紹過,比如遍歷抽象是算法的主

要組建。最後的章節討論前文中沒有涉及到的模式。希望它作爲這一教材的介紹。有興趣的讀者可以繼續閱讀其它書中更完善的討論[3]。

- [1] 譯註:如果子類的specification包括了所有父類的specification,就是說父類的要求也是子類的要求,或者子類的要求更爲嚴格,那麼可以建立父子關係。而替換原則的說法是,對於具有父子關係的類,任何需要一個父類對象的地方,都可以替換爲一個子類對象。
- [2] 譯註:在測試某一組建時,由於其餘組建還未實現,這一組建與其餘組建的接口銜接部分無法工作。此時可以針對這一組建編寫其餘組建的佔位程序(stub),預留出接口的銜接代碼。佔位代碼通常不做任何有價值的事情,只報告組建的銜接部位工作正常。
- [3] 譯註:作者指的是設計模式的開山之作——《Design Patterns—Elements of Reusable Object-Oriented Software》,作者爲設計模式界著名的"四人幫"GoF(Gang of Four)。此書詳盡討論了三大類共23個廣泛使用的設計模式的適用範圍、依

存關係、實現細節以及已有的應用領域等問題。書中以C++和Smalltalk爲示例語言,不過書中所涉及的模式適用於所有面向對象的語言。