Addison Wesley

Java设计：对象、UML和过程

Java Design Objects,UML,and Process

〔美〕Kirk Knoernschild 著
罗英伟 汪小林 译

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

Java 设计：对象、UML 和过程

[美] Kirk Knoernschild 著

罗英伟 汪小林 译

人民邮电出版社

图书在版编目(CIP)数据

Java 设计：对象、UML 与过程/（美）诺厄恩斯蔡尔德（Knoernschild,K.）著；
罗英伟，汪小林译. —北京：人民邮电出版社，2003.4
ISBN 7-115-10879-X

I. J… II. ①诺…②罗…③汪… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2003)第 010960 号

版 权 声 明

Java 设计：对象、UML 和过程

◆ 著 [美] Kirk Knoernschild
译 罗英伟 汪小林
责任编辑 陈冀康

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
读者热线 010-67132705
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：12.5
字数：300 千字 2003 年 4 月第 1 版
印数：1-5 000 册 2003 年 4 月北京第 1 次印刷

著作权合同登记 图字：01－2002－3739 号

ISBN 7-115-10879-X/TP・3198

定价：25.00 元

本书如有印装质量问题，请与本社联系 电话：(010)67129223

内 容 提 要

本书重点介绍如何把 Java、UML、OO 和软件过程等技术有机地结合起来，并成功地运用到软件开发中。

全书分为两部分，共 11 章。第一部分包括前 4 章，重点介绍了 OO 的原理和模式，UML 语言的基础，UML 和 Java 的结合以及软件开发过程。这部分清楚地阐释了每一种技术和概念的优点，强调了它们的共同原则。第二部分描述了如何将这些技术结合应用到软件项目的开发中，包括需求模型、问题分析、建模方式、子系统的设计等等。附录部分则简单介绍了统一开发过程和极限编程、J2EE 和 UML 结合以及书中一个典型 UML 练习的 Java 代码实现。

本书适合面向对象软件工程师、项目管理人员和 Java 程序员阅读。

译者的话

随着 Java 语言的不断发展，以及网络应用的快速普及，人们越来越多地使用 Java 语言作为应用系统的开发工具。但是，正是因为 Java 语言是一门新兴的程序设计语言（这只是相对于 C 语言而言的。尤其是在国内，Java 语言无论是在教学还是应用开发方面，都不是那么普及。很多高校都只是在大四简单地开设入门性课程，更多的高校甚至并不开设 Java 课程），在使用 Java 语言进行应用系统开发方面的经验还是非常欠缺，在整个开发过程中，开发者们都会遇到很多问题，尤其是软件设计问题。

Java 语言本身只是一种实现工具。要想开发出成功的系统，在整个开发过程中，必须综合利用各种技术。OO、UML 和软件过程等都是非常前沿而重要的技术，它们是软件成功开发必不可少的技术。将它们融合起来，应用到 Java 软件系统的开发中去，将为我们开发出符合业务需要的、高质量的、省时省力的软件系统提供强有力的保障。

融合的力量是无穷的。事实上，在任何一个领域，将一系列有利于完成目标任务的技术有机地融合起来形成一个整体，是研究和开发的强有力的手段之一。甚至可以说，在科学研究中，它与创新有着同等重要的地位。当然，这种融合是非常困难的。

本书的作者在应用 OO、UML 和软件过程等技术进行 Java 应用开发方面有着丰富的实践经验，本书正是作者这些经验的结晶。本书的全部目标就是要强调融合——Java、OO、UML 以及软件过程的融合，并且阐述每一项技术是如何融入一个有机的整体的，最终帮助开发者们更好地解决企业级 Java 应用开发过程中碰到的复杂问题。

可以说，在多技术融合方面，本书做了一个很好的尝试，并取得了很好的结果。对于从事多技术融合研究的人们来说，本书的方法也很值得借鉴。

本书可直接用作高校“Java 语言程序设计”课程的教学用书，因为在教授这门课程的同时，应该也必须讲授一些 Java 应用开发的软件工程方法。同时，本书也可作为“软件工程”课程的教学辅助用书。更重要的是，对于那些正在使用 Java 语言开发各种应用系统的软件开发人员来说，本书应该是一本必备的开发指南。阅读完本书后，你们就会发现本书对你的帮助有多大。

参加本书译校工作的还有刘昕鹏、熊国民、郑芳、喻丽姗。由于译者水平有限，错漏之处在所难免，敬请读者批评指正。

译者

2002 年 9 月

序

本书重点讲述如何将 Java、UML、OO 和软件过程等技术有机地融合起来使用。本书将帮助您：

- 理解如何使用 Java 语言，利用已经经过检验的 OOD（面向对象的设计，Object-Oriented Design）原理和模式来开发富于弹性的、强壮的、可扩展的软件系统。
- 在 Java 项目开发过程中，了解如何采用 UML 最有用的特性并从中受益，同时忽略那些不经常使用的特性。
- 为定义良好的、可重复的、可预知的软件开发过程做好充分的准备，确保所构建的软件产品是适用的。

本书记录了我在企业级软件应用系统开发方面的一些经验，包括多年来在教授和应用这些概念的过程中我所指导过的和与我一起工作过的几百名开发者的经验。本书以一种能被广泛理解的方式回答了这些开发者最经常问到的问题。用于讲述这些答案的方法既简洁又清晰，即详细阐述如何将不同的技术融合起来从而获得富于弹性的结果。希望本书所提供的信息能够帮助你在开发过程中既省时，又省力。

本书适合的读者

本书讨论了在软件开发生命周期的实现阶段如何使用 UML。由于强调了 OO、问题求解和通信的重要性，因此对于如何设计更加简洁的 Java 应用程序，本书将给予开发者更加深刻的理解。更多的讨论则集中在对现有的代码进行重构（Refactoring）和整理。利用这些概念，开发者将能够更快地发现富有弹性的答案。

对于如何在结构约束的建立和子系统的设计中使用 UML 来创建能经得起检验和比较的系统，设计者和结构师可以通过本书获得更加深刻的理解，进而从中获益。他们将充分认识我们的模型是如何作为一种有效的机制来验证我们的系统结构。本书所讨论的大量的原理和准则也将对构建更加富于弹性的系统有所帮助，同时也可以作为已有的 OO 设计的衡量准则。

项目管理者、IT 经理人以及项目承担者都可以在本书中获得对这些关键技术重要性的更深刻的理解，并进而从中获益。我们将不再把这些技术看作是一个个分离的实体，而将它们看作是一个互补的工具集，能够融合起来使用，从而降低软件开发的风险。

反馈

非常欢迎来自本书读者的反馈信息。如果您发现一些更为有用的信息，请给我发 E-mail。

更重要的是，我希望能从您那里获得有关如何改进本书的信息。所有这些反馈信息都将使得本书将来的读者能够更多地获得提高他们软件开发水平所必需的知识。我将把有关本书的附加信息发布在 www.kirkk.com/JOUP.html 上。

致谢

特别要感谢本书所有的审阅者，他们深邃的思想保证了本书中所有材料的实用性和精确性。同时，我要真诚地感谢 Adam Brace、John Brugge、Levi Cook 和 David Williams，他们富于思想的审阅对于完成本书起了非常重要的作用。

此外，我还要感谢本书的编辑 Paul Becher。毫无疑问，没有他不断的鼓励和耐心，我将很难完成本书。谢谢 Debbie Lafferty，没有他，我就不能成为 Addison-Wesley 家族的一员。我也要感谢 Tyrrell Albaugh，我的产品经理，在手稿的最终编辑过程中她给予了精心的指导。当然，没有 Nancy Crumpton 的耐心检查，书中的一些病句和语法错误不会被发现。我要感谢 Addison-Wesley 家族的其他成员，很遗憾他们中的大多数我都没能见面，他们做出了重要贡献使得本书能够得以和读者见面。

最后，感谢所有帮助过我的人们（不管帮助有多小），在这里无法一一列出他们的姓名，但我知道他们是谁！

Kirk Knoernschild
joup@krikk.com
www.kirkk.com

作 者 简 介

Kirk Knoernschild 是一位资深的软件顾问，致力于用最优的经验构建性能良好的软件。除了负责大规模的开发项目，他还通过设计课件、培训教学、编写书籍以及在学术论坛和会议发表演讲等方式，与人们分享自己在 UML、Java、OO、软件体系结构、RUP 和 XP 等方面的技术经验。你可以通过 kirk@kirkk.com 和他联系，还可以访问他的站点：www.kirkk.com。

前　言

将一系列技术融合起来形成一个有机的整体比单独使用这些技术本身，有着非常显著的优越性。Java、OO、UML 和软件过程等都是非常前沿而重要的技术，它们对于软件开发的成功具有突出的作用。但如果将它们单独使用，它们的真实威力就无法发挥出来。这四项技术的每一项，虽然是互补的，但它们各自都不同，并且可以单独地学习和应用。然而，如果将它们融合起来使用，开发出符合我们业务需求的、高质量的、省时省力的软件系统的可能性就大大提高了。

我们的目标是讨论一些基本概念，以帮助开发者们更好地使用 UML、OO 和软件过程等技术来解决企业级 Java 应用开发过程中碰到的复杂问题。我们必须确保在将最好的实践经验融合成一个有机整体时，能够充分发挥每一项技术的优势。此外，我们还必须忽略、或者至少要更加明智地使用这些技术中没有太大实际使用价值的特性。我们迫切需要那种能够确保系统的灵活性、可扩展性和可维护性的机制。

很不幸，单个技术的复杂性是无法避免的。然而，我们必须将一系列互补的技术融合起来以保证我们所构建的软件系统足够强壮。因此，本书的全部目标就是强调融合——Java、OO、UML 以及软件过程的融合，并且阐述每一项技术是如何融入一个有机整体的。

融合的力量

在作为一个合作开发者、专业指导人员、咨询者以及顾问的经历中，我发现所有的软件开发人员都在不断地陷入一些相同的基本问题。从本质上来看，这些问题都属于软件设计范畴。不管是调整软件以适应更大的范围，还是在开发一个新系统时管理对需求的持续评估，或者是试图往一个已有的系统中添加新特性，在任何情况下结构机制的使用对于软件开发的成败都是至关重要的。支持变化的系统将可以随着其支撑的业务一起变化，而那些一成不变的系统在它们自身的负担下将不可避免地要崩溃，并逐渐被新系统所代替。

从理论上看，使一个软件系统具有无限可扩展的体系结构，其可能性将是无休止的。简单地将新的构件插入现有系统中可以很容易地适应需求的变化和拓展系统的应用范围。不需要再支持的旧规则可以通过从系统中删除那些相应的构件而得到实现。当然，这仅仅是从理论的角度来看问题，从实践上来说，这几乎是不可能实现的。但这并不是说我们要停止追求，通过不断的努力，相信我们能够逐渐靠近这个目标。

往前走一步确实不容易，它涉及到整个软件开发周期过程中的每一个变化。通过使用当今最成熟的技术、方法和原理，我们能够创建一系列的互补工具，从而使我们的开发能够朝

前发展，进而创建能经得起检验和比较的系统。事实上，这些帮助确保我们的系统在面对变化时能够有更多的灵活性和可扩展性。

在 *The Timeless Way of Building* 一书中，Christopher Alexander 讨论了一个称为“The Quality Without a Name”的结构，从美学观点来看，其感觉非常美妙[ALEXANDER79]。他从各个方面描述了一个能给人以温暖感觉的花园、店面或房间。营造出这种感觉的因素并不是周边环境的某个方面，而是各种因素融合的结果，这种阐述同样适用于软件结构。在与学生和客户的讨论中，我经常问他们在软件开发过程中是否会有同样的感觉。大多数开发者都能清楚地记得他们对所完成的工作的自豪感，因为他们看到所开发的系统是灵活的，能正常工作，并且少有漏洞。他们达到了软件开发的“Quality Without a Name”。

本书是什么

在本书中，我们试图为读者们提供这样一个立体的视角，即在明确的软件过程的帮助下如何综合使用 UML 和 OO 技术来开发 Java 应用系统。根据我们研究和应用的进展，我们将集中讲述 UML 的那些最经常使用的特性，突出软件过程的最好的实践经验。因为本书是围绕整个开发周期来讲述的，因此我们将各种有助于进行全方位设计的最好实践经验的指南在书中都一一介绍了。

任何公司想在一夜之间就成功地应用一项技术都是不可能的。随着当前 Java（包括 Java 2 Enterprise Edition，J2EE）、UML、OO 以及各种各样的软件过程的迅猛发展，一个适当可用的策略是成败的关键。我们所讨论的就是那些能够帮助我们成功进行应用集成的实践经验。

然而，一旦系统需要集成，我们就迫切需要成功的实践经验。系统的面向对象的结构以及系统实现的方式都将对软件的成功产生巨大的影响。我们还讨论了在面向对象的结构设计时开发团队所要做的许多重要决策。

一般说来，我们所强调的是这些技术中比较重要的部分，同时忽略那些在实际中很少使用的部分。因此，在所有软件开发过程中，在确定最基本、最重要的决策时，本书都可以作为它们的指南。

我们的讨论方法是注重实效的。只有在必不可少的情况下，或者确实是能够给读者更深刻的理解，我们才进行有关理论方面的讨论。

本书不是什么

本书并不是一本详细介绍 UML 语法的书籍。花大量时间去学习那些在应用开发中很少使用的技术，只是一种纯学术行为。因此，本书中我们仅仅在需要时才讨论 UML 的有关语法。

本书也不是一本深入学习所有 UML 图的书籍。我们主要讲述的是那些在开发周期中最经常使用到的 UML 图以及那些对应用开发最为有用的 UML 图。这些 UML 图通常是作为一个整体应用到第一次使用 UML 的开发环境中去。

本书也不是一本深入讲述 Java 语言的书籍。读者们只需对 Java 语法有一个基本的了解

即可。因为所有的例子都是用 Java 写的，并且一些讨论也是针对 Java 的，因此，对于那些对 Java 理解不是很深入的读者来说，如果已经掌握了另一门面向对象的语言将非常有助于他们阅读本书。

本书并没有提供任何正式的软件开发过程。相反，我们只是从一系列成功的软件开发过程中抽取出那些最好的实践经验。正因为如此，当我们不断地讨论过程时，我们并没有对某一个特定的软件过程发生太大的兴趣，而是关注那些体现在成功的软件过程中的经验。

本书的基本结构

本书从概念上可以分为两大部分。前四章分别讲述了 UML、OO 和软件过程，这有利于阐明每一项技术的目标和各自的作用。其余章节则以实践和示例的方式强调各技术的融合，所讨论的是前 4 章中各个概念的应用。

如何阅读本书

我们建议读者们按照本书章节的顺序来阅读，因为每一章的概念都是按照整个书的进展来安排的。如果不能按顺序阅读本书，建议如下：

- 如果对 Java 和 OO 非常感兴趣，那么阅读第 1、3 以及 7 到 11 章将是最佳选择。
- 如果希望了解软件过程及其与 UML 的关系，第 4 到 6 章将最有帮助。
- 如果想更多地了解 UML，第 2、3 和 6 章将最为适合。其余的可以只了解一下每章的大概内容。

章节纲要

第 1 章介绍对象以及面向对象的系统设计的目标。本章的一些内容可能会令您吃惊。我们并没有花太多的时间去介绍那些最基本的概念，相反，我们只是根据本书后面章节讨论的重点——原理（Principle）和模式（Pattern），介绍了诸如多态（Polymorphism）和继承（Inheritance）等概念。

第 2 章简要介绍 UML 的发展历史，我们认为这对于完整地理解 UML 是非常必要的。同时，本章还介绍了 UML 的主要目标和它能帮助我们解决什么样的问题。

第 3 章介绍 Java 程序设计语言及其与 UML 的映射。我们并不是试图讲授 Java 语言，而主要是讲述 UML 结构如何映射到 Java 中去。我们从概念层次上阐述了建模（Modeling），这是我们讨论的基础。

第 4 章讨论软件过程在软件开发过程中所起的重要作用。我们发现，以一种能够帮助我们更加容易地确定设法解决问题的方式来构造各种图是非常有益的。我们还介绍任何软件开发过程所能提供的最好的实践经验，并解释 UML 是如何与这些实践经验匹配的。

第 5 章的重点是为我们强调 UML、Java 和对象等技术而做铺垫，同时讨论那些将各种互补技术有机融合起来的关键因素。本章还考查了开发团队或组织在开发环境中集成各种新技

术时所应考虑的各种因素。

第 6 章开始对 Java、UML、OO 和软件过程等技术进行融合。首先讨论建立系统需求所要产生的基本产品。本章是后面各章讨论的基础。我们并没有详细论述获得或管理需求的方法和实践，而是给出了一个简单的需求示例及一个可选的方案。

第 7 章的工作就是要明确我们初步的分析结果。通过分析上一章所给出的需求，我们确定了初始的分析类。我们将它们分为边界类、实体类或控制类，这样一划分，就可以根据它们的行为来组织抽象类。这就是系统设计的初步成果。

第 8 章强调的是系统的动态特性。介绍了关于 UML 顺序图的更多细节，包括它的句法元素。同时还讨论了在为第 7 章所列的初始类分配行为时的一些重要决策。

第 9 章对系统的静态特性进行了讨论。基于在第 8 章中所讨论的那些为对象分配行为以及对象之间的合作，在这里，我们可以更好地为系统结构设计做准备。本章大量地使用了 UML 类图，同时还讨论了很多重要的设计决策。本章不仅阐述了类之间的关系，同时也给出了组成系统的包（Package）之间的关系的包图。

第 10 章讨论软件体系结构在开发更具灵活性、可维护性和可扩展性的系统时所起的重要作用。除了讨论软件体系结构的重要性之外，在示例中我们还介绍了通用的结构机制和模式，并讨论了它们的作用。此外，本章还提供了我们所给的包之间关系的详细介绍。

第 11 章介绍了系统及其本质。同时我们还介绍一个子系统应有的重要特性。

附录 A 介绍理性统一过程（RUP）和极限编程（Extreme Programming，XP）。同时还讨论那些流行的软件开发过程的相似性和差异。

附录 B 讨论 UML 如何与 J2EE 结合使用。此外，还将阐述 J2EE 是如何与本书内容相匹配的。

附录 C 给出了第 3.7 节中有关 UML 的第一次讨论的示例的源代码。

在当今技术发展变化极为迅猛的时代，任何开发团队在开发软件系统时都不可避免地要面临诸多挑战。因为有太多的新技术可用，在使用它们时应使我们能够受益是非常重要的。因此，我们必须掌握每一项技术中能使我们充分受益的各个环节，而忽略那些并不重要的方面。

此外，在充分发挥这些技术的优势时，我们必须使这些技术以一种统一的方式协同工作。这使得一项技术的长处可以弥补另一项技术的弱点，进而可以最大程度地为开发具有灵活性、健壮性、可维护性和可扩展性等特点的高质量软件系统提供保障。这样，在软件开发中，Alexander 所说的那种“Quality Without a Name”的境界，就可以通过融合的力量来实现！

目　录

第 1 章　面向对象的原理与模式

开发更多富于弹性的系统是我们行动的第一步。重用就是针对此目的的。

当设计面向对象的系统时，会面临着非常多的挑战，并且解决方案也是各式各样的。怎样才能确定一种方法，它将确保我们能够创建一个可扩展的、健壮的、易于维护的系统呢？一个可行的方法就是使用设计模式（Design Pattern）。设计模式被证明是能够精确剪裁以适应特殊设计要求的设计方案。从本质上看，设计模式是可重用的设计模板（Design Template）。当模式的思想自 1995 年由“Gang of Four”[GOF95]发表了基础性工作进入了主流发展以来，已有模式的数目已经变得难以控制。如今已有太多的模式，以至于当我们试图找到一个能令人信服地解决设计难题的模式时，所花费的时间甚至比设计一个新的解决方案还要长，而这个新的、有效设计的解决方案完全可能已经在某个地方被当作一个模式。当我们找不到一个能够解决难题的模式时，我们应当在设计过程中采用一个确保我们能正确解决难题的方法，而不管是否有容易得到的可用模式。这样的方法是基于面向对象思想的一些基本原理的。

当开发面向对象软件时，这些基本的原理能够提供有用的指导，但我们必须首先理解面向对象的思想。当我们没有完全理解一个原理的价值时，实际上我们就不可能应用这个原理。因此，我们不仅要理解这些原理，而且要理解面向对象思想的真正益处，以及能够帮助我们有效地和轻松地达到的目标。

1.1　原理、模式和 OO 范例

到现在，我们一直沉浸在对象方法带来的好处中。重用是面向对象方法的核心。不幸的是，许多现存的、讨论面向对象的著作都处在一个理论的高度，以至于它们很难被解释并在程序设计中加以应用，或者说这些著作处于一个如此细节化的层次上，以至于很难由它的全部内容得到一个简明版本的范例。理解诸如抽象、继承、封装和多态等概念是很奇妙的，但是它们只是概念，并没有在创建可重用的、高度可维护的系统方面提供更多的指导。实际上，本书中的讨论已经假定读者对这些概念有了一个基本的理解。

我们可以通过研究和应用模式来获得重用性，创造更加有弹性的设计，以及更彻底地理解面向对象范例。但是即使是模式也不能像指导性原理那样具有通用性，而且在过去几年中随着模式的广泛增加，仅仅找到最合适的模式也是件令人畏缩的事情。这带来了一些有趣的问题。面向对象范例的基本原理是什么？有没有这样一个指导性原理的集合，能够一贯地、忠实地应用于帮助我们创建更为健壮的系统？实际上是有的，而且我们将在这一节后的第 1.2