国外经典教材

PEARSON Prentice Hall

# Practical Object-Oriented Development with UML and Java

# UML与Java面向对象开发实践

(美) Richard C. Lee
William M. Tepfenhart 著

王晨澈 译

清华大学出版社

国外经典教材

# UML 与 Java 面向对象开发实践

（美） Richard C. Lee
William M. Tepfenhart 著
王晨微 译

清华大学出版社
北 京

## 内 容 简 介

本书将软件开发过程看作一个建模过程，通过对 4 个模型的建模过程的说明，阐述了面向对象技术。这 4 个模型分别为规格说明模型、分析模型、设计模型和代码模型。同时，在描述建立这几个模型的时候，运用了 UML 技术，采用了 UML 中的各种图对建模过程进行说明，如用例图、顺序图等。而且，在代码模型中，以 Java 语言为例介绍了面向对象语言如何实现。


EISBN: 0-13-067238-6


北京市版权局著作权合同登记号　图字：01-2003-0565

**本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签，无标签者不得销售。**

**图书在版编目(CIP)数据**

UML 与 Java 面向对象开发实践/（美）李（Lee, R.C.），（美）泰芬哈特（Tepfenhart, W.M.）著；王晨激译.—北京：清华大学出版社，2003
（国外经典教材）
书名原文：Practical Object-Oriented Development with UML and Java
ISBN 7-302-07603-0
Ⅰ.U… Ⅱ.①李… ②泰… ③王… Ⅲ.①面向对象语言，UML－程序设计－教材 ②JAVA 语言－程序设计－教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 103448 号

**出 版 者**：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
**社 总 机**：010-62770175　　**客户服务**：010-62776969

**文稿编辑**：刘伟琴
**封面设计**：久久度企划
**印 刷 者**：北京牛山世兴印刷厂
**装 订 者**：三河市化甲屯小学装订二厂
**发 行 者**：新华书店总店北京发行所
**开　　本**：185×260　**印张**：25.25　**字数**：592 千字
**版　　次**：2003 年12月第 1 版　2003 年12 月第 1 次印刷
**书　　号**：ISBN 7-302-07603-0/TP・5595
**印　　数**：1～4000
**定　　价**：45.00 元

# 译 者 序

拿到本书，最令读者感到兴奋的可能就是本书涉及了当前软件领域最为重要的技术中的两项——面向对象技术和统一建模语言（UML）。早期的过程语言早已让位于面向对象语言，如 C++和 Java，这两种面向对象语言在当今各种应用开发中已经占据了重要地位，尤其是 Java，它作为一种纯面向对象的语言具有更多的优势。现在的软件开发不仅仅是编写代码，更多的工作量和重心体现在对问题域的分析和系统设计过程中，而这两个过程也就是将现实中的问题域建模的过程，UML 就是为这个建模过程而设计的，使用它能够更好地整理和表达我们对于现实的理解。本书恰恰抓住了这两个热点技术，这无疑是本书的一大优势。

本书是一本较实用的面向对象和 UML 方面的教材，适合自学使用。本书按照整个面向对象分析、设计、编码等阶段，对历史上曾使用的技术以及当今的最新技术都做了介绍，涉及内容比较广泛。本书顺序性较强，建议读者从前向后阅读。相信阅读本书之后，您对于面向对象技术会有一个比较全面的了解，而且在 UML 和 Java 方面具有一定基础，可以开始实践摸索。

本书的作者是面向对象技术方面的专家。Richard C. Lee 有 35 年以上的软件工程开发和管理经验。他曾工作（或管理）于电子成本、嵌入系统、大型 IMS 项目、多媒体、操作支持系统、处理控制、事务处理和切换方面的前沿开发。作为面向对象技术最早的使用者之一，他当前的兴趣是成功地完成更多的面向对象项目。William M. Tepfenhart 现在是 Monmouth 大学的软件工程系的程序主任。他作为程序员、开发人员和技术人员，具有 18 年开发制造业、军事和通信应用程序的工作经验。在过去的 17 年他一直都在开发面向对象系统。

在本书翻译过程中，本人得到了很多帮助，在此表示感谢——刘伟琴编辑给译者提出了很多建设性的建议；刘红艳、张恒、赵娟、于虹、卢颖提供了技术方面的支持；付亚伟、陈琳、仲志芬参与审稿；王以贵、付静林、高峰、吴丽琴协助录入。

由于时间仓促以及水平有限，错误之处在所难免，敬请读者批评指正。

# 前　言

本书适用于大型系统的专业软件分析者和开发者。您若无暇参加课程学习，但又需要掌握统一建模语言（unified modeling language，UML）和 Java，以跟上快速发展的面向对象技术，那么本书是很合适的自学指南。本书将帮助您理解面向对象分析、面向对象设计和面向对象编程之间的差别。其目标是：

- 用 Java 建立面向对象的应用，并根据业务需要进行折衷做出决定。
- 阐明与面向对象技术相关的基本概念。
- 为学生和该领域的从业者提供有足够深度的知识覆盖面，以保证他们能跟上快速发展的步伐。
- 集中阐述了面向对象技术作为一种软件工程工具的实用性，同时也揭示了围绕这一技术的一些神秘现象。
- 提供了一种使用面向对象技术进行分析、设计以及编程的实用方法。
- 展示了如何用 Java 实现面向对象技术。
- 在现有文献资料中的理论和应用实践之间建立平衡。

深入理解面向对象的重要概念和问题并不需要了解计算机科学或高等数学。即使是介绍编程的章节也不需要 Java 方面的背景知识，这些章节阐明了运行的 Java 代码是如何产生的。

## 1. 面向对象技术

我们是大型系统的软件开发者，已经交付了用多种编程语言编写的代码，这些代码代表了许多软件技术。我们几乎经历了过去 30 年中的所有软件革命，因此可以说面向对象技术是我们用过的最重要的软件技术。

为什么这么说呢？因为面向对象技术已经改变了软件的制作方式，也改变了应用通过互联网以及在不同厂家的电脑之间相互通信的方式。并且，对象模型正在改变我们设计业务过程的方式以及考虑企业的方式。

大部分的企业正在重新设计自己，以应对当前由互联网带来的业务竞争。面向对象在这一过程中起着重要作用，它提供了一种用于捕捉业务流程、规程、政策以及方便设计的规则的模型。使用工具将模型转化为可操作的系统能够加快重新设计的速度。随着市场或

业务条件的改变，应该通过更新模型及使用这些工具来重新生成这些系统，以反映这种变化。在过去的几十年中，扎实的软件工程实践相对于其他方法，带我们走得更远更快。

通常认为面向对象技术削弱了软件危机，面向对象技术这一机制对软件的意义，正像螺钉和横梁对于建筑设计，或者是芯片对于计算机硬件设计那样重要。这种信仰源于如下几个原因：

- 精通高层的面向对象模型为软件设计者提供了现实世界的、可编程的组件，因而降低了软件开发的成本。
- 面向对象技术的代码共享和重用能力缩短了应用的开发时间。
- 通过编程抽象机制可以使更改所造成的影响局部化和最小化，这可以加快增强开发并提供更可靠更健壮的软件。
- 面向对象技术管理复杂事务的能力允许开发者从事更有难度的应用开发。

面向对象的概念集合就是一个模型化现实世界的工具集。这个面向对象工具集为开发者管理复杂事务提供了一个最好的方法。面向对象概念确实能帮助开发者生产灵活且易于维护的软件。

## 2. 为什么选择统一建模语言

作为面向对象技术的实际应用者，我们知道如果应用恰当的话，所有的方法都会产生相同或相似的模型。但不同的建模语言表示法却阻碍了其发展。统一建模语言（UML）已经成为业界标准，它把不同的建模表示法溶合起来而形成一种建模语言表示法。这已足够成为选择 UML 的原因。

UML 是用来编档分析和设计模型的一种语言。它提供了用来捕捉大部分解决实际业务问题时发现的有价值的概念和机制所必需的所有绘图图标，并提供了对编档模型来说很重要且必要的所有图表。最后，UML 是一种活的语言，用它可以扩充表示法，以表示 Rational 软件公司的著名的由 Grady Booch，James Rumbaugh 和 Ivor Jacobson 组成的团队中尚未定义的机制。

UML 不是本书的中心主题，它是作为编档分析和设计模型的方法而提出的，这种模型的开发方法才是本书的中心主题。所有的 UML 图形都是根据这些图中所捕捉的信息以及捕捉这些信息的方式来提出和讨论的。

## 3. 为什么选择 Java

只有 Java 是一种纯面向对象的编程语言，与多范式的编程语言 C++相比这反而限制了它的使用。然而 Java 的优势远远胜过任何局限性，尤其是 Java 运行在 Java 虚拟机（Java

Virtual Machine，JVM）上，这使 Java 程序可以在任何运行了 Java 虚拟机的机器上运行。这将开发者从必须为硬件和操作系统的不同组合而设计和实现相同功能的情况中解脱了出来。

使用虚拟机的好处乍看可能并不明显。一个好处就是计算机厂商现在知道了他们可以将开发经费只投入惟一的一个实现，而不用再开发 5 个或 6 个实现了，这样就可以将注意力集中于工具和产品的开发。这意味着他们可以着重实现更重要的功能（具有更大的业务价值），这也影响了重用工作。我们可以在运行任何操作系统的任何硬件上开发库，并且不用因为平台的不同而进行修改就能重用这些代码。编程错误不会在一个平台的代码版本上出现，而在另一个平台的代码版本中不出现。代码重用的更宽广的基础意味着可以用更短的时间和更低的成本实现组件更大的可靠性。分析者可以把注意力放在业务价值方面，设计者可以把注意力放在更大的灵活性和可维护性方面，实现者和测试者可以把注意力放在质量、可靠性以及性能方面。这种注意力的改变的最终结果是用更少的钱开发出了更好的代码。

当我们查看数量众多的 Java 库（框架）时就可以发现这些好处，这些库现在可以从 Sun 公司得到。目前可供开发者使用的库有：一套很好的通用工具库、一个高性能的图形用户界面库（比如 SWING）以及专用于业务的类库。和这些库的 C++版本相比，Java 版的库更复杂、功能更强大并更易于与最终产品相结合。因为这些库从所有的项目中都可以得到，并且其他书中已有很好的说明，所以这些库得到了广泛的使用。因此，使用这些库的专业知识也很容易得到。将它们与具有完全不同的应用程序接口（application programming interfaces，API）的 C++版的库相比较，在 C++中，开发者可能是一个平台上的某一产品的专家，但他对其他平台的产品却一无所知。几乎没有程序员在 PC 机、Mac 以及 Unix 这三个平台上同时编写过程序。

将 Java 虚拟机直接与网页浏览器结合使得从 Internet 上下载并在浏览器内运行 Java 程序成为可能。这有助于在浏览器环境中提供更大的功能，并已经产生了一种新的应用类。可以这样说，如果没有 Java 的话，在 Internet 给用户提供的功能方面我们将无法获得近期的收获。现在客户端应用和服务器端应用都可以是 Java 程序，Java 在现代 World Wide Web 中的广泛应用将不会减少，直至有新的技术（仍未发现）可以提供更强大的抽象集合和同样宽广的平台支持性。

## 4. 我们使用面向对象技术的方式

我们不是面向对象的纯化论者，也不是理论家，我们只是开发者，想利用任何好的思想来帮助实现两个很关键的业务目标：进一步降低开发成本，并缩短上市时间。我们相信，这些技术目标——可靠性、可维护性以及灵活性，对满足这些业务目标来说是很关键的。

我们使用面向对象技术的方式是管理软件开发的复杂性，以使软件可靠、易维护并且

灵活。管理复杂性是实现这些技术目标同时也是业务目标的关键。要在复杂问题领域内管理复杂性，就需要开发者懂得对象、类、关系和规则是如何适合对象范式的。为大部分复杂问题领域建模的时候，首先要找到对象、类以及诸多对象之间的关系，此外还需要捕捉该问题领域中的规则（策略）。因此，必须使用很丰富的静态建模技术来捕捉这些数据（对象）关系。

很多面向对象专家认为关系是“糟糕的”，因为它们破坏了面向对象的封装性。但是我们看来，关系有助于管理问题领域中的复杂性，并有助于实现业务目标，因此我们很乐意使用关系，并且期待会有更多的机制和语言对这方面的支持。第 9 章关于描述型语义学中提到，应该将规则和策略作为模型的一个组成部分，而不是作为一个特定的子系统扩展。

对复杂问题领域建模所用的各种机制跟选择 UML 作为建模语言及选择 Java 作为编程语言是一致的。用 UML 和 Java 可以定义任何需要的机制来帮助我们建立更容易管理的软件。

为了捕捉模型的过程方面，我们讨论行为（动态的和静态的）以及多态。用有限状态机或者其他状态模型可以帮助我们在处理计时、同步和中断时，管理过程的复杂性。我们也为管理错误恢复（错误恢复是一个重要的主题，因为错误恢复可以构成一个程序逻辑的一半）而提出了异常。通常，大部分的面向对象书籍对这些方面都有所忽略或者是没有注意到。

我们相信要想成功建立大型面向对象系统，关键是要求开发者和程序员了解比多数有关面向对象技术的书籍所教授的更多的内容。建立大型系统需要使用由一些面向对象专家们提出的机制，这些机制并不是所有人都能理解的，但专业的开发者在成为多产的团队成员之前最起码要理解如何处理问题领域的这些方面。本书并不能使我们成为一名专家，开发一个系统仍然离不开专家或顾问。本书运用了 80/20 规则：80%的内容是教大家怎样成为一名多产者，而 20%的内容是帮助大家理解专家是怎样解决困难的。

本书并没有介绍面向对象技术方面的最新趋势和时尚，包括对象设计模式、标准模板库以及分布式对象计算。虽然这些内容都很有意思，但对我们的目标并没什么意义，本书旨在提供一个实用框架，以帮助那些对面向对象编程不太熟悉的开发者尽快赶上来。

最后，很多专家都认为面向对象技术是一项成熟的技术，但我们并不认同这一点。我们认为它正在趋向成熟。面向对象技术具有以前的技术（过程的、函数的、基于规则的，等等）所不具备的巨大潜能来帮助我们管理复杂事务。我们在面向对象技术和 Java 中看到很多不同的抽象机制正在融合（结合）成为一种真正强大的技术，虽然这种融合尚未完成，但 Java 已经比其他任何面向对象技术完成的好得多。

## 5. 本书的组织结构

本书讲述如何合理运用面向对象技术和方法，这些规则并不是绝对的，目的是帮助读

者在开发软件和使用 Java 进行编程时，很好地理解面向对象的概念和设计规则。

本书是一本自学指南，读者应该按顺序阅读。本书采用 Richard 已用了多年的教学方法来讲解面向对象概念和基本技能，不过，我们并不提倡用这种方法来建立面向对象系统。每章讨论一个面向对象技术方法的主要步骤，大部分章节都以一步一步的形式给出了结论性的指导。希望读者只将这些步骤作为参考，要根据常识而不能盲目地遵循下面的步骤。

**第 1 章**　将抽象作为控制复杂性的机制来介绍，并且将面向对象作为抽象机制的长继承链中的现代的形式进行了介绍。

**第 2 章**　提出面向对象的基本规则。

**第 3 章**　开发过程的开始阶段，使用用例的方法来开发规格说明模型。

**第 4 章**　开发分析模型过程的开始阶段，识别对象/类/接口。

**第 5 章**　描述如何通过识别与对象相关的属性（数据）和服务来区别“实际”对象和“虚假”对象。

**第 6 章**　阐明如何捕捉对象的行为。

**第 7 章**　描述如何识别并描述动态行为。

**第 8 章**　描述组织系统中的所有对象时可用的各种关系（泛化/特化、链接、对象聚合等等）。

**第 9 章**　描述如何将声明的事实同面向对象的模型相结合，模型是关于对象的知识以及实现它们的一个基于规则的机制。

**第 10 章**　回顾分析模型并将它重新构造以考虑助手类。

**第 11 章**　论述开发设计模型[①]的一些元素。

**第 12 章**　提出使用 Java 语言编程。

**第 13 章**　介绍类和接口是如何用 Java 实现的。

**第 14 章**　描述静态行为可以如何实现。

**第 15 章**　描述动态行为可以如何实现。

**第 16 章**　描述泛化/特化可以如何实现。

**第 17 章**　描述额外的关系可以如何实现。

---

① 设计是很复杂的，并且从系统的角度来看，设计是一个和面向对象技术无关的主题。然而，第 14～22 章从如何实现分析概念的角度提出了设计问题。所有的实现章节中都提到了习惯用语和设计模式。

**附录 A** 提出了统一建模语言的概括指南。

**附录 B** 提出 Java 的概括。

**附录 C** 提出 Java 和 C++的对比。

## 6. 怎样使用本书

本书主要用于有经验的软件开发者和高年级大学生。本书内容取自于教授合格程序员的行业课程所采用的授课材料。这些材料是在两个为期一周的课程中提出的。第一周包括前 11 章，第二周包括本书剩余的章节。这一课程采用让学生们完成一个实用项目的方法，学生们可以通过这个项目开发一个计算机游戏，而不采用让学生们完成家庭作业的方法。第一个教程结束时，学生们完成了游戏的设计工作，第二个教程结束时，学生们对他们的设计有了完全的功能实现。

我们采用基于项目的方式进行授课有以下几个原因。第一，家庭作业要么是太琐碎，不能有效地传达概念的重要性，要么是太复杂，不能在一个合理的时间内完成。第二，这个范式的价值最好从同概念一致的应用中学习，这只能通过项目来获得。第三，真实的项目有真实的成就感，因为这个项目不是简单的可以在一天内完成编码的小程序。第四，项目是在团队的环境中开发的，并经过讨论、决策以及决定的反复过程，这些都是开发程序时真实发生的。第五，对于大部分大学生来说，这将是他们要指定、分析并设计的第一个真实大小的项目。最后，选择一个规模合适的项目使学生能够掌握所有关键概念。

大学里使用的典型项目是一个大的历险游戏，在这个游戏里各个人物探索虚拟世界，拾得财宝、跟怪物或恶棍搏斗并获得最终的目标。这些游戏通常合并了上百个类以及同样多的关系。这些游戏包括很多不同种类的地形、武器、怪物、财宝和人物。随着网络游戏的广泛使用，很多项目小组已经开发了多人参与的游戏。大部分情况下，项目小组开发的游戏跟很多商业化的游戏产品具有相同的复杂程度。

适中的项目小组包括 3～4 个学生，太大的小组会花费太多的时间来达成一致，而更小的小组又很难完成任务。项目小组在课堂上进行项目开发，这样指导老师可以检查其进度，并回答与概念应用相关的一些问题，因此班级的规模应控制在一个可以管理的大小上。要按时完成项目，学生们必须整周都花在项目上，并且活动安排要和授课内容协调一致。

以下是推荐的课程活动安排。假设一学期有 15 周，第 15 周期终考试。这个安排的关键特点是在学期早期学生们有充足的时间来定义游戏并开发用例。有时讲课会比开发团队的活动早 3 周之多，但我们发现这种安排大有好处，因为它可以避免学生犯一些通常的错误，比如：混淆属性和相关性，或者混淆对象状态和对象属性。

# 致　谢

我们需要感激许多人。人们需要一本书来讲述用 Java 进行程序开发的更多工程方法，这成为我们写此书的力量源泉。尽管对我们的个人生活造成了影响，但还是应感谢他们的执着和鼓励。

因为我们主要是开发者（而不是研究者、学者或作家），所以我们省略了面向对象研究者的工作（他们最先提出所有的思想）以及面向对象作家的工作（他们在早期作品中向我们讲述了这些思想），而只是简单地把这些思想用于实际应用中。对于这些思想、概念、机制和技术的所有提出者和那些伟大的面向对象作家，在此我们一并表示感谢，没有他们就不可能有这本书。

理论和思想固然好，然而，对实践者来说，经验才是最好的老师。若没有在实际项目中应用面向对象技术和方法的经验，我们也写不出这本书。感谢我们的许多老板，无论是现在的还是以前的，他们有勇气让我们进行前沿的（常常是要付出代价的）软件开发。没有他们的支持，我们就无法对我们所写的内容进行测试。同时也感谢审阅者弗吉尼亚技术大学的 William McQuain 和南卡罗莱纳州大学的 Michael Huhns。

Richard Lee 在此感谢许多曾为他工作过的人们，也感谢那些把本书所写的思想运用到实际项目中的先驱们，他们分别在自己的公司中将 Java 技术首先应用到大型项目中，和 Richard 共同分享了作为“第一”的激动和痛苦。对所有这些人，Richard 一并表示感谢。

William Tepfenhart 感谢 Monmouth 大学的同事们，感谢他们协助整理本书。最重要的是要感谢他的家人，因为要优先考虑交稿日期，他们情愿忍受寂寞。

Richard C. Lee

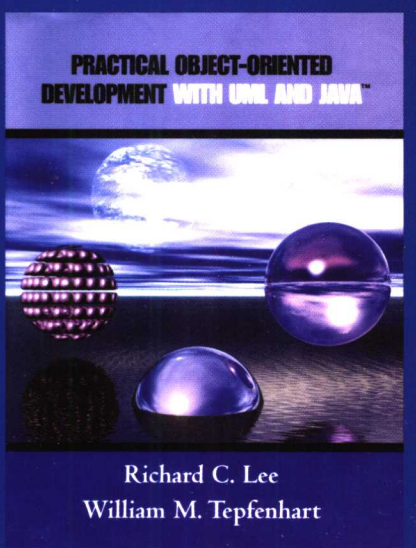
William M. Tepfenhart

(美) **Richard C. Lee**
**William M. Tepfenhart** 著

作者简介

理查德·C. 李（Richard C. Lee）有35年以上的软件工程开发和管理经验。他曾从事电子出版、嵌入系统、大型IMS项目、多媒体、操作支持系统、处理控制、事务处理和交换等前沿开发工作，并做过这些领域的管理工作。作为面向对象技术最早的使用者之一，他当前的兴趣是成功地完成更多的面向对象项目。

威廉姆·M. 泰芬哈特（William M. Tepfenhart）目前是Monmouth大学软件工程系的程序主任。他作为程序员、开发人员和技术人员，具有18年开发制造业、军事和电信应用程序的工作经验。在过去的17年中他一直从事面向对象系统的开发工作。



## 王晨溦 译

译者简介

王晨溦，1978年生于河南省郑州市，2000年以优异的成绩毕业于西安工业学院，获得计算机及应用专业学士学位。毕业之后，一直从事软件开发工作。曾从事于ERP系统和GIS系统的开发工作。现任北京数字证书认证中心软件工程师，从事安全方面的系统开发工作。

# 目　录