致读者：

我从 2002 年 7 月开始翻译这本书，当时还是第二版。但是翻完前言和介绍部分后，chinapub 就登出广告，说要出版侯捷的译本。于是我中止了翻译，等着侯先生的作品。

我是第一时间买的 这本书，但是我失望了。比起第一版，我终于能看懂这本书了，但是相比我的预期，它还是差一点。所以当 Bruce Eckel 在他的网站上公开本书的第三版的时候，我决定把它翻译出来。

说说容易，做做难。一本 1000 多页的书不是那么容易翻的。期间我也曾打过退堂鼓，但最终还是全部翻译出来了。从今年的两月初起，到 7 月底，我几乎放弃了所有的业余时间，全身心地投入本书的翻译之中。应该说，这项工作的难度超出了我的想像。

首先，读一本书和翻译一本书完全是两码事。英语与中文是两种不同的语言，用英语说得很畅的句子，翻成中文之后就完全破了相。有时我得花好几分钟，用中文重述一句我能用几秒钟读懂的句子。更何况作为读者，一两句话没搞懂，并不影响你理解整本书，但对译者来说，这就不一样了。

其次，这是一本讲英语的人写给讲英语的人的书，所以同很多要照顾非英语读者的技术文档不同，它在用词，句式方面非常随意。英语读者会很欣赏这一点，但是对外国读者来说，这就是负担了。

再有，Bruce Eckel 这样的大牛人，写了 1000 多页，如果都让你读懂，他岂不是太没面子？所以，书里还有一些很有“禅意”的句子。比如那句著名的“The genesis of the computer revolution was in a machine. The genesis of our programming languages thus tends to look like that machine.”我就一直没吃准该怎么翻译。我想大概没人能吃准，说不定 Bruce 要的就是这个效果。

这是一本公认的名著，作者在技术上的造诣无可挑剔。而作为译者，我的编程能力差了很多。再加上上面讲的这些原因，使得我不得不格外的谨慎。当我重读初稿的时候，我发现需要修改的地方实在太多了。因此，我不能现在就公开全部译稿，我只能公开已经修改过的部分。不过这不是最终的版本，我还会继续修订的。

本来，我准备到 10 月份，等我修改完前 7 章之后再公开。但是，我发现我又有点要放弃了，因此我决定给自己一点压力，现在就公开。以后，我将修改完一章就公开一章，请关注 www.wgqqh.com/shhgs/tij.html。

如果你觉得好，请给告诉我，你的鼓励是我工作的动力；如果你觉得不好，那就更应该告诉我了，我会参考你的意见作修改的。我希望能通过这种方法，译出一本配得上原著的书。

shhgs

2003 年 9 月 8 日

介绍

“他带来了语言，语言又创造了思维，而思维是衡量万物的标准” ——《解放了的普罗米修斯》，雪莱

人类...很大程度上是在受语言的支配，而语言也已经成为一种媒介，透过它我们可以了解社会的方方面面。你能想象，一个人能不借助语言而完全适应这个世界，或是仅仅把语言当作解决具体问题的交流工具和表述手段吗？实际上，“真实世界”在很大程度上是建立在人类语言的习惯之上的，而这又是人们没有意识到的。

《语言学，一门科学的地位》， Edward Sapir，1929

就像人类语言，Java 也提供了一种表述概念的方法。如果运用得当，随着问题大型化和复杂性程度的提高，它会比其他方法更加简单灵活。

你不能仅仅将 Java 看作是一组特性的集合——有些特性，如果孤立看待的话是毫无意义的。你不能只想着写代码，一定要树立了设计的观念，这样才能发挥这些特性的综合效能。而要能这样理解 Java，你必须先弄懂它的，以及编程所共有的问题。本书讨论编程问题，它们为什么会成为问题，以及 Java 所采取的解决这些问题的方法。由此，我是基于 Java 语言在解决这类问题上所采取的方法，来讲解每章所介绍的特性的。我希望能用这种方法，一步一步地将你领入 Java 的殿堂，最终让它成为你的母语。

在这本书里始终贯穿着这样一个观点，那就是，你应该在你的头脑中建立一个能让你深入理解解这个语言的模型；如果碰到什么问题，就能把它交给这个模型，然后由它来回答。

预备知识

本书假定你有一些编程的经验：你知道程序是语句的集合；知道子程序，函数，宏的概念；知道“if”是控制语句而“while”是循环语句，等等。你可能是在不同的场合学到这些的，可能是在用宏语言编程的时候，也可能是在用 Perl 之类的工具的时候。只要你已经写过一些程序，并且对这类基本的编程概念不感到陌生，你就能读完这本书。当然本书对于 C 程序员来说会比较简单，对 C++的程序员来说会更简单，所以如果你对这两种语言不熟悉的话，就得更用功一些(此外，随书附送的多媒体 CD 会帮助你快速掌握学习 Java 所需的基本知识的)。不过除了介绍面向对象的编程(object-oriented programming (OOP))的概念之外，我还是会介绍 Java 的基本控制机制。

虽然我会经常提到一些 C 和 C++的语言特性，但这么做并不是为了作更深入的解释。这么做只是为了帮助程序员能用从其它语言的角度来观察

Java，毕竟 Java 是由它们发展而来的。我会尽量说得简单一些，并且对那些我认为的，非 C/C++的程序员可能不熟悉的东西都作一些解释。

学习 Java

大约在我的第一部书 *Using C++* (Osborne/McGraw-Hill, 1989) 出版的时候，我就开始教 C++语言了。教授编程语言成了我的职业。算起来从 1987 年开始，我也走了不少地方。无论在哪里，我都能看到，听众中有的人在微微点头，有的人面无表情，有的人疑惑不解。等到我在课堂上为较小的人群开始讲课的时候，我发现做练习的时候，甚至是那些微笑点头的人都对很多问题不甚了了。我觉得，这么多年倡议和主持软件开发研讨会(Software Development Conference)的 C++专题(以及后来的 Java 专题)，使得我和其他一些发言人的形成一种普通听众无法接受的讲话风格。通常我们讲得太多，而且讲得也太快了。所以最终，由于听众水平的参差不齐，以及我本人的表述不当，我失去了部分听众。或许这个要求高了点，但由于我是那种不太喜欢传统授课方式的人(我相信很多人都是因为课堂教学太枯燥才不喜欢的)，因此我要让每个人都能跟上进度。

有一段时间，我写了很多小的程序。于是，通过实验与迭代(iteration，一种很适于 Java 程序设计的技术)，我学会了 Java。最后总结过去的教学经验，我开了一门课。这门课用一种不连续的，易于理解的步骤来解决教学中遇到的问题。此外，在培训班里(这是最理想的学习环境)，每节课之后还有练习。现在，我的公司，MindView, Inc. 提供公开的和在课堂教学的 *Thinking in Java* 的课程。这是一门很重要的入门课程，是为其它更高级的课程打基础的。你可以在 *www.MindView.net* 网站上找到更加详细的内容。(这门课程也以 *Hands-On* Java CD ROM 的形式发布。可以在同一个网站上找到相关信息。)

培训班上的反馈给了我很大帮助。据此我修改材料，重新设置重点，直到认为它是一本好教材。但是本书并不是课堂笔记；我尽量介绍更多的内容，经过编排之后一个专题一个专题地作介绍。最重要的是，本书是专为那些独自一人，与一门新的编程语言作斗争的读者们设计的。

目标

同我前面写的一本书，*Thinking in C++*一样，这本书的结构也是遵照教学的需要进行编排的。更何况我写这本书就是要给培训班准备教料。我是从一堂好的课应该教些什么内容，这个角度来安排本书的章节的。我的目标是，先整理出一些能在合理的时间内讲完的东西，再做一些在课堂里面可以完成的练习。

在本书中我的目标是：

1. 每次只介绍一小部分内容，这样在进入下一部分之前，你就能完全掌握这些概念。
2. 使用尽可能简单短小的例程。这点有时会妨碍我使用“真实世界”的例子。不过我发觉初学者们更乐意理解例程中的所有细节，而不是去感叹例程所解决的问题的难度。同时课堂环境有限制，代码不能太多。毫无疑问，我会因此受到使用“玩具例程”的批评，但是为了教学的需要我准备接受。
3. 仔细地安排特性的讲解顺序，这样你就能在看到其用法之前先对这个课题有一点认识。当然，有些情况下不可能做到这点；这时，我会先作一个简单的介绍。
4. 我不会把我知道的所有东西全都告诉你们，我只会教给你们一些我认为最重要的东西。我相信信息是分重要性的，有些东西，也就是那些只会把人搞昏，只会加深人们对语言复杂性成见的细枝末节，百分之九十五的程序员永远都不需要知道。以 C 为例，如果你记得操作符优先级的顺序(反正我从来不记)，你就能写出非常高效的代码。但如果你真的那么做的话，那么读代码和维护程序的人都会被你整死。所以还是别记了，真的搞不清楚的时候就用括号。
5. 每一节都只讲一个问题，这样讲课时间以及每次练习之间的间隔会很短。 这样不仅能让学员在课堂上保持精力集中，思维活跃，而且能给读者更高的成就感。
6. 帮你打下一个坚实的基础，为进入更为复杂的课程，阅读难度更大的书作准备。

JDK 的 HTML 文档

Sun Microsystems 的 Java 语言和类库(可从 *java.sun.com* 自由下载)会附带一份可以用 Web 浏览器阅读的电子文档。实际上，所有第三方的 Java 实现都带了它自己，或者相同的文档。出版的 Java 书籍也差不多都是在重复这些文档。所以，既然你可以下载，或者已经有了这份文档，那么除非必要，否则本书不准备再重复这些文档。况且用 Web 浏览器找一个类会比在书里查快得多(在线的文档可能还更新)。因此本书只会简单的告诉你，去查“JDK 文档”。只有在例程讲解的过程中需要用文档来做补充说明的时候，才会详细介绍某个类。

章节

设计本书时，我始终在想：大家都是怎样学 Java 的。课堂上的反馈给了我很大的帮助，使我认识到哪些是需要着重阐明的难点。有几次，我雄心勃勃地准备了很多内容，打算一次讲完，但是随着课程的发展，我发现：如果一次要讲很多新的特性，你就必须把它们全都讲清楚，但是这样一来，学员们容易就会被搞糊涂的。所以我费了很大功夫，尽量作到在介绍新特性的时候，每次都只讲尽可能少的内容。

因此，本书的目标就是，在不依赖尚未介绍过的特性的前提下，每一章都只讲一个或者一组相关的特性。这样你就可以在开始下一阶段的学习之前，先在现有的知识背景下掌握这部分内容。

下面简要介绍一下本书的各章。这些章节分别与 *Thinking in Java* 课程的授课和练习时间相对应。

第一章：对象简介

(CD ROM 上有相关课程)。这一章是一个关于面向对象的编程都讲了些什么的概述，并且它还回答了“对象是什么？”这一根本性的问题。本章涉及到接口与实现，抽象与封装，消息与方法，继承与合成，以及多态性这一微妙的概念。此外它还对对象创建方面的问题作了个概述，这些问题包括构造函数，如何持有对象，对象创建之后保存在哪里，以及神奇的，能清除无用对象的垃圾回收器。此外，这一章还会介绍一些别的内容，包括用异常来处理错误，用于创建反映敏捷的用户界面的多线程，网络与 Internet。你会知道是什么使 Java 如此的卓尔不群，而它又是凭什么能大获成功的。

第二章：万物皆对象

(CD ROM 上有相关课程)。学完这一章之后你就能写第一个 Java 程序了。本章从本质问题入手，先作一个综述：对象的 *reference* 的概念；如何创建一个对象；primitive 类型和数组的简介；作用域以及用垃圾回收器是如何清理对象的；Java 里面的所有东西是怎样变成一种新的数据类型(类)的；创建类的基本方法；方法，参数，和返回值；名字的可见范围以及如何使用其它类库里面的组件 (component)；**static** 关键词；注释与内置的文档。

第三段：控制程序流程

(*Thinking in C* 的 CD ROM 上有相关课程)。这一章从 C 和 C++带到 Java 里的操作符入手。接下来会讲操作符的通病、传递、提升以及优先级。然后是实际上每种语言都有的基本的流程控制和选择操作：**if-else** 的判断结构，**for** 和 **while** 的循环结构，用 **break** 和 **continue** 以及 Java 的带标签的 **break** 和带标签的 **continue**(它们使得 Java 不再需要 **goto** 了)，以及用 **switch** 进行选择。尽管这部分内容同 C 和 C++的是一脉相承，不过还是有些区别。

第 4 章：初始化和清理

(CD ROM 上有相关课程)。这一章从介绍构造函数开始。构造函数能保证进行适当的初始化。定义构造函数又引出了方法的重载(overload，因为你可能需要好几个构造函数)。接下来讨论清理过程，这可不像它看上去那么简单。通常，不要用对象的时候，直接把它丢了就是了，垃圾回收

器会接手处理并释放内存的。这部分深入分析了垃圾回收器及其与众不同之处。最后，它对初始化作了一个详细的考察：成员数据的自动初始化，成员数据的赋值初始化，初始化的顺序，static (静态)初始化，以及数组的初始化。

第 5 章：隐藏实现

(CD ROM 上有相关课程)。本章介绍如何封装代码，并且解释了为什么类库中的某些部分会被公之于众，而有些部分却被隐藏起来了。这部分内容从讲解 **package** 和 **import** 这两个关键词入手。这两个关键词能进行文件级别的封装，主要用于类库的构建。接下来的课题是目录路径与文件名。本章剩余的部分探讨了 **public**，**private** 和 **protected** 这三个关键词和 package 访问权限 (package access) 的概念，以及在不同的上下文环境中各种访问权限所表示的意义。

第 6 章：复用类

(CD ROM 上有相关课程)。最简单的复用类的方法是合成(*compostion*)，也就是将对象嵌入新的类里。然而，合成并不是唯一的，使用已有的类来创建新类的方法。实际上所有的 OOP 语言都有继承的概念。这是一种能为已有的类添加新功能 (也可以修改它——这是第 7 章的主题) 的方法。通常用继承来复用代码的时候，不需要改动“基类”，而是通过对它进行修补来产生要用的东西。在本章，你会学到 Java 是如何用合成和继承来复用代码的，以及它们是如何运用的。

第 7 章：多态性

(CD ROM 上有相关课程)。多态性是 OOP 的基石，如果你只靠自学，可能要花九个月时间才能发现和理解这个概念。在本章中，通过许多小的、简单的例子，你会看到如何通过继承来创建一个类系(a family of types)，并且使用这个类系的基类来操控属于这个类系的对象。Java 的多态性能让你以整体的角度来看待这个类系的所有对象，也就是说绝大多数代码不依赖于某个具体的类型。这能使程序更为灵活，于是编程和维护代码也变得更加简单和便宜了。

第 8 章：接口 (*Interface*) 和内部类 (*Inner Class*)

Java 为设计和关系复用提供了一种特殊的工具：接口(*interface*)，这是一种纯粹的对象接口的抽象。**interface** 不仅仅是抽象类(abstract class)的极端形式，它还能是 C++里面的“多重继承”的一个变例。用它创建类之后，你就可以把它上传到多个基类。

乍看上去，内部类就像是一种简单的隐藏代码的机制；也就是把一些类放在另一些类里。但是，你会发现内部类并不是那么简单的；它知道包在它外面的那个类，并且还能同它通信。有了内部类，你就可以写出更为优

雅、清晰的代码。然而对于绝大多数人来说，内部类还是一种新的概念，要把它熟练地运用到设计中去还需要时间。

第 9 章：用异常 (Exception) 处理错误

Java 的基本哲学就是，要让有问题的代码根本没有机会运行。编译器会尽可能地捕捉错误，但有些问题——也许是程序员的错，也许程序正好好地运行着，但是其它什么地方出了问题——只能到程序运行的时候才能被发现并处理。Java 有一种能处理程序运行时所发生的所有问题的“异常处理(*exception handling*)”机制。本章会讲解 Java 的 **try**，**catch**，**throw**，**throws**，以及 **finally** 关键词的工作机理，以及何时抛出异常，捕获异常之后又应该做些什么。此外，你还会看到 Java 的标准异常，如何创建你自己的异常，构造函数里面碰到异常之后又该怎么做，以及异常处理程序是如何发现异常的。

第 10 章：检测类型

Java 的运行时类型识别(run-time type identification 缩写为 RTTI)能让你在只持有基类的 reference 的情况下，找出对象的准确类型。通常情况下，你会故意忽略对象的准确类型，然后让 Java 的动态绑定机制(就是多态性) 来实现这个类的正确行为。但是，有的时候，能够准确地知道这个对象是什么类型的是非常有用的。通常是因为，你要利用这种信息来更为有效地进行一些特定情况的操作。本章还会介绍 Java 的 *reflection* 机制。此外还包括，为什么要有 RTTI 和 reflection，如何使用，以及什么情况下应该不使用 RTTI。

第 11 章：对象的集合

如果程序只处理数量有限，且寿命可知的对象，那么这个程序是相当简单的。总之，程序总是在不断地创建对象，但是具体是在什么时候，只有到程序运行的时候才能知道。此外，不到运行时你是不会知道到底需要创建多少对象，甚至是什么类型的对象。要解决这一普遍的编程难题，你必须能够随时随地创建任意数量的对象。本章深入探讨了 Java 所提供的各种容器类库。这些类库就是用来保存待处理对象的，它包括简单的数组，以及更为复杂的像 **ArrayList** 和 **HashMap** 之类的容器(数据结构)。

第 12 章：Java I/O 系统

从理论上讲，所有程序都可以分成三个部分：输入，处理，和输出。也就是说 I/O(输入/输出)是程序的重要组成部分。在本章中，你会学到 Java 所提供的各种用以读写文件，内存块，以及控制台的类。然后是 Java I/O 架构的演进以及 JDK 1.4 中的“新”I/O (**nio**)。 此外，本章还演示了如何使用 Java 的对象序列化 (*object serialization*) 将一个对象“流”化 (这样就把它能放进磁盘或是通过网络进行传输)，然后再重建。接下来会讲解 Java ARchive (JAR) 文件格式所使用的 Java 的压缩

类。最后是新引入的 preference API 和正则表达式(regular expression)。

第 13 章：并发

Java 提供了内置的，能让你在单个程序里面并发运行多个被称为线程的子任务的支持。(除非你的机器上有多个处理器，否则多个子任务只是表象。)虽然，这种多线程技术能被运用于任何地方，但是最常见的还是用来创建反映敏捷的用户界面，这样程序在进行处理的时候，就不会妨碍用户去揿按钮或输入数据了。本章会帮你在多线程编程领域打下一个坚实的基础。

第 14 章：创建窗口与 Applet

Java 有一个 Swing GUI 类库，这是一组能创建跨平台的用户界面的类。它所创建的窗口程序既可以是运行在 World Wide Web 上的 applet，也可以是独立的应用程序。本章只是 Swing 编程的一个介绍。此外还示范了 Applet 的签发和 *Java Web Start*。当然，它还介绍了 *JavaBeans* 这项重要的技术。这是创建 RAD(Rapid Application Development)编程工具的基础。

第 15 章：发现问题

对于开发能正常运行的程序，语言的检查机制就只能做到这里了。这一章介绍了一些能帮助我们解决编译器所不能解决的问题的工具。自动化的单元测试 (*automated unit testing*) 是这个领域的一项重大进展。本书定制了一个保证程序能产生正确输出的测试系统，但是我们也介绍了 **JUnit**，这个事实上的标准。自动编译是由开放源代码的标准工具，**Ant** 来实施的，此外还有用于团队编程的 CVS。本章还介绍了能在程序运行时报告问题的 Java 断言机制(*assertion*，这里同“订单设计 (*Design by Contract*)”一起讲)，logging API，debugger，profiler，甚至 doclet (它能帮助我们发现源代码里面的问题)。

第 16 章：分析与设计

面向对象是一种新的，不同的思考编程的方式，而且很多人在第一次接触 OOP 项目的时候，确实碰到了麻烦。一旦你理解了对象的概念，并且越来越多地以面向对象的方式来思考问题的时候，你就能利用 OOP 的优势来构思“好”的设计了。本章介绍了分析和设计的概念，以及一些具体的方法，以帮助你在合理的时间内开发一个好的 OOP 程序。这些内容包括 UML (Unified Modeling Language) 图表及其相关的方法，使用情况 (use case)，Class-Responsibility-Collaboration (CRC) 卡片，迭代式的开发，极限编程 (*Extreme Programmng* 缩写是 XP)，用来开发与演进(evolve)可复用的程序的方法，以及向面向对象的开发方式迁移的策略。

附录 A：传递和返回对象

由于 Java 里面只有通过 reference 这一种办法跟对象打交道，因此把对象传给方法以及从方法里面返回对象，就有了一些很有趣的结果。这个附录会讲，向方法传递对象和从方法返回对象的时候，应该怎样管理对象。此外本章还谈到了 **String** 类。在这个问题上，它用了一种不同的方法。

附录B：Java编程准则

这个附录囊括了这么多年来我所发现和整理的，能对你设计和编写底层程序起到帮助和指导作用的经验。

附录 C：补充资料

MindView 所提供的其它学习资料的介绍：

1. 本书背后的 CD ROM，其中包括了本书的预备教材——CD 版的 *Foundations for Java*。
2. *www.MindView.net* 还提供 *Hands-On Java* 的第三版的 CD ROM。这是基于本书的 CD 版的课程。
3. Thinking in Java 的课程。这是 MindView 公司的主要的入门级课程，其内容也是基于这本书的。*www.MindView.net* 上有课程的安排以及报名表。
4. *Thinking in Enterprise Java* 这本书是讲适合企业编程的，更高级的 Java 课题的。可以从 *www.MindView.net* 下载。
5. *The J2EE Seminar*。这个课程会介绍如何用 Java 编写真正能用的，可以在 Web 上发布的，分布式应用程序。参见 *www.MindView.net*。
6. 对象与系统设计(Designing Objects & Systems)的课程。面向对象的分析，设计以及实现的技巧。参见 *www.MindView.net*。
7. *Thinking in Patterns (*用 *Java* 语言描述*)*，这本书探讨的是一些比较高级的 Java 课题，包括设计模式，以及解决问题的技巧。 可以从 *www.MindView.net* 下载。
8. *Thinking in Patterns Seminar*. 这门课就是在讲上面提到的那本书。*www.MindView.net* 上提供课程的安排以及报名表。
9. *Design Consulting and Reviews*。主要针对项目管理。

附录 D：资源

我认为特别有用的 Java 书籍的清单。

练习

讲课的时候我发觉，做一些简单的练习会对学生完全掌握所学的内容产生无法替代的作用，因此我会在每一章的最后都加一些练习。

绝大多数的练习都非常简单，因此它们都能在课堂里面，在有限的时间以及教师的监督下完成，这样就能保证所有的学生都已经理解这些内容了。有些难度大一点，但是都算不上很难。(我敢肯定你会自己去找一些真正的挑战的——但是更可能的情况是，它们会自己找上门来。)

The Thinking in Java Annotated Solution Guide 电子文档会提供某些练习的答案，只要付很小一笔钱就能从 *www.BruceEckel.com* 下载。

CD ROM

这一版还附送了包装在书背面的 CD ROM。过去我是反对在我的书里加 CD 的，因为我觉得为了区区几个 k 的源代码而支付一张 CD 的价格，是很划不来的，因此我更喜欢让读者到我的网站上去下载。但是，你很快就会看到，这张 CD 有点不一样。

这张 CD 里面没有本书的源代码，相反只是加了一个连到 *www.MindView.net* 上的源代码的链接(你不一定要用 CD 上的链接。你可以直接上这个网站去找)。这么做有两个原因：第一，把 CD 送给出版商的时候，源代码还没有完全写完；第二，这个方法能让我对源代码作一些改进，而且出问题的时候，也可以及时更正。

由于本书在这三个版本里作了许多修改，因此 CD 还包括了 HTML 格式的本书的第一和第二版。其中包括了，由于前面所提到的原因，在后面的版本中被删掉的章节。有可能你会要用这几章。此外，你可以从 *www.MindView.net* 下载本书的 HTML 格式的最新版(第三版)，以及勘误表。HTML 有一个好处，那就是索引都是超链接，因此查起来比较简单。

这张 400 多兆的 CD 的绝大部分内容都是一个名为 *Foundations for Java* 的多媒体课程。这其中包括了 *Thinking in C* 的课程。这门课是介绍 C 的语法，操作符和函数的，而 Java 的这部分东西正是来源于 C。此外，它还包括了由我亲自制作和讲解的，CD 版的 Hands-On Java 课程的第二版的前七章。虽然 *Hands-On Java* 的 CD 一直是单独卖的(第三版的 *Hands-On Java* CD 也一样，读到本书的时候，已经可以买到这张 CD 了——见 *www.MindView.net*)，但是从第二版起，我就决定在 CD 里面放上前面七章。因为相比第三版，这部分内容不会有很大的变化，因此它们(再加上 *Thinking in C*)不仅能为你学习本书和上 *Thinking in Java* 的课程打下更好的基础，而且还能让你了解 *Hands-On Java CD*第三版的质量和价值。

最初，我要求 Chuck Allison 把这张 CD 课程里面的 *Thinking in C* 部分做成一个单独的产品，不过最后还是决定把它放到第二版的 *Thinking in C++*和 *Thinking in Java* 里面，因为老是会有一些根本没有任何 C 基础的人来上我的课。他们想法很明显“我是一个聪明的程序员，我不想

学 C，我要学的是 C++或者 Java，所以我应该跳过 C，直接从 C++/Java 开始”。等到开课之后，他们才发现，预先掌握 C 的语法是多么的重要。在这本书里加了 CD ROM 之后就能确保来上我的课的人都已经做好准备了。

这张 CD 还能吸引更多的听众。尽管第 3 章(控制程序流程)已经涵盖了 Java 从 C 那里得来的基础性的东西，但是 CD 上的介绍还是更亲切一些，而且它所要求的编程基础比书上的更低一些。此外用 *Hands-On Java* CD 上的内容来学习前面七章，能为你学习 Java 打下更好的基础。我的愿望是，这张 CD 能吸引更多的人加入 Java 编程的行列。Hands-On Java CD ROM 的第三版只能直接从 *www.BruceEckel.com* 定购。

源代码

本书的所有源代码都是以 freeware 形式发布的，你可以到 *www.Bruceeckel.com* 网站去下载打包的源程序。为了确保你们都能得到最新的版本，我将它定为发布本书的源代码和电子版的官方网站。你可以在其它网站上找到本书电子版和源代码的镜像 (*www.BruceEckel.com* 列出了一些)，但是你应该检查一下官方网站，以确保镜像网站上刊载的确实是最新的版本。你可以在课堂，以及其它教学环境里分发这些代码。

声明版权的目的，主要是想确保源代码会被适当的引用，并且要禁止人任何未经授权的情况下就将它刊登到印刷物上。(只要是引用本书的源代码，那么在绝大多数媒介里使用本书的例程，通常都不会有问题。)

在每段源代码文件里，你都会找到指向如下的版权声明的内容：

```
This computer source code is Copyright ©2003
MindView, Inc.
All Rights Reserved.

Permission to use, copy, modify, and distribute this
computer source code (Source Code) and its
documentation
without fee and without a written agreement for the
purposes set forth below is hereby granted, provided
that
the above copyright notice, this paragraph and the
following five numbered paragraphs appear in all
copies.

1. Permission is granted to compile the Source Code
and to
include the compiled code, in executable format only,
in
personal and commercial software programs.

2. Permission is granted to use the Source Code
without
modification in classroom situations, including in
```

```
presentation materials, provided that the book
"Thinking in
Java" is cited as the origin.

3. Permission to incorporate the Source Code into
printed
media may be obtained by contacting

MindView, Inc. 5343 Valle Vista La Mesa, California
91941
Wayne@MindView.net

4. The Source Code and documentation are copyrighted
by
MindView, Inc. The Source code is provided without
express
or implied warranty of any kind, including any
implied
warranty of merchantability, fitness for a
particular
purpose or non-infringement. MindView, Inc. does not
warrant that the operation of any program that
includes the
Source Code will be uninterrupted or error- free.
MindView,
Inc. makes no representation about the suitability
of the
Source Code or of any software that includes the
Source
Code for any purpose. The entire risk as to the
quality
and performance of any program that includes the
Source
code is with the user of the Source Code. The user
understands that the Source Code was developed for
research
and instructional purposes and is advised not to
rely
exclusively for any reason on the Source Code or any
program that includes the Source Code. Should the
Source
Code or any resulting software prove defective, the
user
assumes the cost of all necessary servicing, repair,
or
correction.

5. IN NO EVENT SHALL MINDVIEW, INC., OR ITS
PUBLISHER BE
LIABLE TO ANY PARTY UNDER ANY LEGAL THEORY FOR
DIRECT,
INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
DAMAGES,
INCLUDING LOST PROFITS, BUSINESS INTERRUPTION, LOSS
OF
BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS,
OR FOR
PERSONAL INJURIES, ARISING OUT OF THE USE OF THIS
SOURCE
CODE AND ITS DOCUMENTATION, OR ARISING OUT OF THE
INABILITY
```

```
TO USE ANY RESULTING PROGRAM, EVEN IF MINDVIEW, INC.,
OR
ITS PUBLISHER HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH
DAMAGE. MINDVIEW, INC. SPECIFICALLY DISCLAIMS ANY
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR
PURPOSE. THE SOURCE CODE AND DOCUMENTATION PROVIDED
HEREUNDER IS ON AN "AS IS" BASIS, WITHOUT ANY
ACCOMPANYING
SERVICES FROM MINDVIEW, INC., AND MINDVIEW, INC. HAS
NO
OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES,
ENHANCEMENTS, OR MODIFICATIONS.

Please note that MindView, Inc. maintains a web site
which
is the sole distribution point for electronic copies
of the
Source Code, http://www.BruceEckel.com (and official
mirror
sites), where it is freely available under the terms
stated
above.

If you think you've found an error in the Source
Code,
please submit a correction using the URL marked ""
in the electronic version of the book, nearest the
error
you've found.
```

只要保留源文件中的版权提示，你就可以在项目或课堂上(包括演示材料中)使用这些代码。

编码标准

在本书的文字中，标识符(方法，变量以及类的名字)都用**粗体字**来表示。绝大多数的关键字也用粗体来表示。但是像“class”这样用得实在太多，因而不宜设成粗体的，不在此列。

在本书中，我用了一种特定的编码风格。这种风格就是 Sun 的风格，实际上它自己的网站上的所有代码都遵循这种风格(见 *java.sun.com/docs/codeconv/index.html*)，而且看上去绝大多数的 Java 开发环境也都支持这种风格。如果你读过我写的其它书，你会发现 Sun 的编码风格同我的正好相同——虽然这不是我的功劳，但我还是很高兴。程序格式(formatting style)的问题可以争论上好几个小时，所以我不不打算用我的程序来衡量那种才是对的；我用这种风格自有我自己的道理。由于 Java 是一种形式自由的(free-form)编程语言，因此你尽可以使用你自己觉得舒服的风格。

本书的程序都是编译之后直接放进写这本书的字处理程序的。因此书里的程序应该都没有编译错误。那些“应该”在编译时产生错误信息的代码都已经用 **//!** 注释掉了，所以你可以很容易地发现它们，并且进行自动测试。错误一旦被发现，并且报告给了作者，会首先在发布的源文件里得到更正，然后在书再版的时候作更正(这些错误也会出现在 *www.BruceEckel.com* 网站上)。

Java 版本

通常我要依靠 Sun 的 Java 实现来判断行为是否正确。

本书关注的是 Java 2，JDK 1.4。而且也是用这个版本测试的。如果你想学习本书没有涉及的较早版本的 Java，那么可以到 *www.BruceEckel.com* 上去下载本书的第一和第二版。此外，本书附带的 CD 中也有这两本书。

错误

无论作者花了多少心思去检查错误，总会有一些错误能躲过检查潜伏下来。然后每当有新的读者翻到这一页的时候，它们又会从纸面上跳出来。

由于聪明的读者们所提供的建议实在是太有价值了，因此我开发了好几个被称为 *BackTalk* 系统的版本 (由 Bill Vernners 帮助构思，使用多种技术，有许多人帮助实现)。在可自由下载的本书的电子版中，每一段文字的最后都有一个唯一的 URL，它会向 BackTalk 系统发送一个邮件，并且记录下你对这段文字所作的评论。这种方法能很容易的跟踪和更新更正。如果你觉得你发现了什么错误，请使用 BackTalk 系统来报告错误，并提交更正建议。谢谢你的帮助。

封面设计的注解

Thinking in Java 的封面的设计受到了 19 世纪末 20 世纪初兴起，到 1900 至 1920 年发展到高峰的美国艺术和工艺运动(American Arts & Crafts Movement)的影响。这个流派兴起于英格兰，它既是工业革命的机器大生产的一种产物，也是对维多利亚时代那种高度装饰性的风格的回应。艺术与工艺流派崇尚简洁的设计，强调这一运动对于艺术的本质表现形式的看法，它要求手工制作，重视个人创作的重要性，但是它并不排斥使用现代工具。这一流派的影响直到今天依然存在：一个世纪过去了，计算机革命也从萌芽发展为影响到我们每个人的洪流，而软件设计技巧也已经不再是编写代码了。

我是这样看待 Java 的：这是一种尝试，它要帮程序员脱离操作系统维修工的身份，而要让他们成为软件艺术家。

本书的作者和封面的设计者(我们还很小的时候就已经成了朋友了)都从这一运动中获得启示，我们都有自己设计的，或有那段时间风格的家具，台灯，以及其它东西。

这个封面还有一个主题，那就是很多盒子。这是自然学家用来展示昆虫的盒子。这些昆虫都是对象，而它们都被放进了盒子这个对象。盒子这个对象又被放在“封面这个对象”里面，因此这张封面是在用图讲解面向对象的编程里面的“聚合”概念。当然程序员会情不自禁的想起“bug”，但是这里的 bug 都已经捉住了，而且作成了标本，最后还放在陈列用的小盒子里，因此它体现了 Java 在寻找，展示和征服 bug 方面的能力(实际上这也是它最强大的功能)。

致谢

首先我要感谢那些同我一起讲课，提供咨询服务，以及开发课件的同事们：Andrea Provaglio，Dave Bartlett，Bill Venners，Chuck Allison，Jeremy Meyer，以及 Larry O’Brien。我要感谢你们的耐心等待，使我能不断尝试，最终为同我们一样的独立民间人士开发出最好的教学模式。

最近，我同许多帮助我写这本书的人建立了联系，他们通常都是在自己家里工作，这一切当然要归功于 Internet。如果是在过去，我恐怕得花一大笔钱来为他们租办公场地，但是有了网络和 Fedex，此外偶尔还有电话，我就能在不花很多钱的情况下得到他们的帮助了。在我学习更好地“与别人合作”的过程中，他们都给了我很大的帮助。我希望我能继续得到别人的帮助，以把自己的工作做得更好。在帮助我处理那些临时决定的商务活动方面，Paula Steuer 作出了无法估量的贡献(感谢你 Paula，在我不想干什么事的时候督促我去干)。Jonathan Wilcox 先生帮我审查了公司的治理结构，仔细检查了所有可能会有漏洞的地方，并且手把手的教我们怎样才是诚实合法的经营。感谢你长久以来的关心。感谢你在面对那些极难处理的计算机问题的时候所表现出的耐心。Sharlynn Cobaugh(她第一个发现 Paula)已经成为语音处理方面的专家了，她是制作多媒体教学 CD 的重要成员，并且还担负着别的一些任务。Evan Cofsky (Evan@TheUnixMan.com)已经成为我的开发团队的重要成员了，他是 Python 的行家，帮我解决了很多难题，包括重新对 BackTalk 进行架构上的(最终的？)设计，使之成为一个由 email 驱动的 XML 数据库。还有那些在 Prague(布拉格)的 Amaio 的人也帮我解决了一些问题。Daniel Will-Harris 是第一个鼓励我在 Internet 上工作的人，当然他也是设计方案的关键人物。

为了写这本书，我做了一件在心里酝酿了有一段时间的事情。2002 年夏天，在 Colorado 的 Crested Butte，我办了一个实习计划。最初想找两个人，最后来了五个(有两个是志愿者)。他们不仅对我写书有帮助，而且他们还让我成为项目的焦点。感谢 JJ Badri，Ben Hindman，Mihajlo

Jovanovic，Mark Welsh。Chintan Thakker 是第二个实习者，他一直待到书完成还没走，由于我们总得在 Mount Crested Butte 租间房间，因此我们等广告找了一些志愿者，他们是 Mike Levin，Mike Shea，以及 Ian Phillips。他们都对我有所帮助。或许过段时候我又会开一个实习计划；到 www.MindView.net 来看消息吧。

感谢 Doyle Street Cohousing Community，我写本书第一版的两年间，他们对我非常宽容(而且他们对我一直这样宽容)。感谢 Kevin 和 Sonda Donovan，把 Colorado 州 Crested Butte 的这么好的房子转租给我，使我能集中精力撰写本书的第一版(还要感谢 Kevin 帮我重新装修了在 CB 的房子)。还要感谢 Crested Butte 和 Rocky Mountain Biological Laboratory 的居民们，他们让我找到了宾至如归的感觉。此外，还有我在 CB 的瑜伽老师，Maria 和 Brenda，在写本书第三版的过程种，她们帮我保持良好的状态。

Colorado 州 Crested Butte 的 Camp4 Coffee 已经成为来上课的老师的标准聚会地了，而且还是课间休息的时候我最喜欢，也最便宜的去处。要感谢 Al Smith 开了这么好的一个咖啡店，而且还经营得如此成功，这是我在 Crested Butte 最有趣也最愉快的经历。

感谢 Moore Literary Agency 的 Claudette Moore，她用非凡的耐心与恒心帮助我写出满意的作品。感谢 Prentice Hall 的 Paul Petralia，他一直在为我创造条件，为了能让我顺利的写下去，他全力以赴的帮我作了很多事(而且还容忍了我的一些比较特殊的要求)。

我的前两本书是由 Osborne/McGraw-Hill 出版的，编辑是 Jeff Pepper。Jeff 在正确的时间出现在了正确的地点。他跳槽到了 Prentice Hall。在由 Paul 接手之前，他承担了这几本书的大量的准备工作。谢谢你，Jeff。

感谢 Rolf André Klaedtke (瑞士); Martin Vlcek，Vlada & Pavel Lahoda，(布拉格)；以及 Marco Cantu (意大利)。我第一次到欧洲组织培训课程的时候，他们接待了我。

感谢 Gen Kiyooka 以及他的 Digigami 公司，慷慨地提供给我 Web 服务器。前几年，我的网页是建在他的服务器上的。这种帮助是无法估量的。

特别要感谢 Larry 和 Tina O'Brein，他们帮我把课程改造成 *Hands-ON Java* CD ROM 的第一版。(可以在 *www.BruceEckel.com* 找到更多的关于这张 CD 的信息。)

在我写这本书的时候，下面这些开源工具帮了我很大的忙，每次用到它们的时候，我都要感谢那些制作者们。Cygwin (*http://www.cygwin.com*)帮我解决了大量 Windows 不能，或者不愿

意解决的问题，现在我一天也离不开它了(真希望 15 年前就有它，那时我满脑子都是 GNU Emacs)。在我的 Java 开发步骤里面，CVS 和 Ant 起着非常重要的作用，现在我已经会不去了。我甚至喜欢上了 JUnit (*http://www.junit.org*)，实际上它是要让你去做“你能做的最简单的事情”。Eclipse (*http://www.eclipse.org*)是 IBM 奉献给开发社区的一件精彩的礼物。它还在不断完善之中，因此我还期待着能从它那里看到更多的了不起的东西。(IBM 是怎么变成嬉皮士的？我肯定是漏看了哪份备忘录)。开发过程中每天都会用到 Linux，那些实习生用得更多。当然，还有我一直在说的 Python (www.Python.org)，这是我的朋友 Guido Van Rossum，以及 PythonLabs 的一群白痴天才们的杰作。我和他们一起呆了好几天，我们一道以 XP 的方式开发 Zope 3，这真是很了不起的几天 (Tim 现在我已经把从你那里借来的鼠标用框子给框了起来，我还给它起了一个正式的名字叫“TimMouse”)。你们这些家伙应该找一个对健康有益一些的地方吃午饭。(此外，还要感谢整个 Python 社区。他们有很多人)。

很多人都曾经给我发过勘误信，在此我表示感谢。我要特别感谢(针对第一版)：Kevin Raulerson (找到了成吨的 bug)，Bob Resendes (简直不可思议)，John Pinto，Joe Dante，Joe Sharp (他们三个都非常了不起)，David Combs (很多语法和表述方面的改进意见)， Dr. Robert Setphenson，John Cook，Franklin Chen，Zev Griner，David Karr，Leander A. Stroschein，Steve Clark，Charles A. Lee，Austin Maher，Dennis P. Roth，Roque Oliveira，Douglas Dunn，Dejan Ristic，Neil Galarneau，David B. Malkovsky，Steve Wilkinson，以及很多别的人。Prof. Ir. Marc Meurrens 花了很多时间帮我整理本书第一版的电子版，并且负责在欧洲发布。

感谢那些帮助我(在第二版当中)用 Swing 类库重写例程，并且提供其它帮助的人：Jon Shvarts，Thomas Kirsch，Rahim Adatia，Rajesh Jain，Ravi Manthena，Banu Rajamani，Jens Brandt，Nitin Shivaram，Malcolm Davis，以及所有表示支持的人。

我有很多技术上很牛的朋友，他们不但在技术上影响了我，而且还教我瑜珈和其它一些有益身心的锻炼，我发现这种锻炼这能带给我灵感，给我启示。他们是 Kraig Brockschmidt，Gen Kiyooka，以及 Andrea Provaglio (在意大利，他帮我理顺了 Java 和编程的总体思路，现在他已经移居美国，并且成为 MindView 团队的一员。)

对我来说，理解 Delphi 有助于理解 Java 并不是什么值得大惊小怪的事，因为这两种语言之间有许多相通的概念和设计。我那些 Delphi 朋友给了我许多 Delphi 语言的真知灼见。他们是 Marco Cantu (也是意大利人，是不是讲拉丁语的人在编程方面特别有天赋？)，Neil Rubenking (在接触计算机之前，他常醉心于瑜珈、素食、以及禅宗)，当然还有 Zack Urlocker (Delphi 的产品经理)，我们是老朋友了，曾经一同环游过世界。

我的朋友 Richard Hale Shaw 的见解和支持是很有帮助的 (当然还有 Kim)。Richard 和我合作开了好几个月的课，我们竭尽所能，以期为学员们提供一期尽善尽美的课程。

这本书的设计，封面，以及封面照片都是由我的朋友 Daniel Will-Harrisd 设计的，他是一位知名的作者和设计家(*www.Will-Harris.com*)。上初中的时候他就喜欢摆弄胶字，那时计算机和桌面排版系统还没有发明，他会一边玩一边数落我老是在那里嘀咕一些代数问题。但是，后来我做了一个照相排版软件(camera-ready pages)，因此我就得对排版错误负责了。我用 Windows 版的 Microsoft Word XP 写这本书，并且直接用 Adobe Acrobat 创建印刷用的胶片；这本书是直接用 Acrobat PDF 文件印刷的。本书第一和第二版出版的时候我正巧在国外，这一切要归功于我们所处的电子时代。第一版是在南非的开普敦(Capetown)完成的，第二版是在布拉格(Prague)写就的。第三版是在 Colorado 的 Crested Butte 写的。正文字体是 *Geogia*，抬头用了 *Verdana*。封面所有的字体是 *ITC Rennie Mackintosh*。

我要特别感谢所有教过我和我教过的人(他们也是我的老师)。我遇到的最有趣的写作老师是 Gabrielle Rico (她是 *Writing the Natural Way* 的作者，这部书在 1983 年由 Putnam 出版的)。我会永远记得在 Esalen 渡过的那极不寻常的一周。

我女朋友 Dawn McGee 为我排了封底那张照片，是她叫我这么笑的。

支持我的朋友里面还包括，但只不限于：Andrew Binstock，Steve Sinofsky，JD Hildebrandt，Tom Keffer，Brian McElhinney，Brinkley Barr，*Midnight Engineering Magazine* 的 Bill Gates，Larry Constantine 和 Lucy Lockwood，Greg Perry，Dan Putterman，Christi Westphal，Gene Wang，Dave Mayer，David Intersimone，Andrea Rosenfield，Claire Sawyers，还有一些意大利人 (Laura Fallai，Corrado，Ilsa，以及 Cristina Giustozzi)，Chris 和 Laura Strand 夫妇，Almquist 一家, Brad Jerbic，Marilyn Cvitanic，Mabry 一家，Haflinger 一家，Pollock 一家，Peter Vinci，Robbins 家族，Moelter 家族 (以及 McMillan 一家)，Michael Wilk，Dave Stoner，Laurie Adams，Cranston 一家，Larry Fogg，Mike 和 Karen Sequeira 夫妇，Gary Entsminger 和 Allison Brody，Kevin Donovan 和 Sonda Eastlack，Chester 和 Shannon Andersen 夫妇，Joe Lordi，Dave 和 Brenda Bartlett 夫妇，Patti Gast，Rentschler 一家，Sudek 一家，Dick，Patty，和 Lee Eckel，Lynn 和 Todd，以及他们家庭。当然还有老爸老妈。