Sun公司核心技术丛书

第6版

Java 2

核心技术 卷I：基础知识

涵盖Java J2SE 1.4版

Core Java 2 Volume I Fundamentals

Sun microsystems

(美) Cay S. Horstmann
Gary Cornell 著

程峰 黄若波 章恒翀 译

机械工业出版社
China Machine Press

Sun公司核心技术丛书

Java 2核心技术

（第6版）

卷Ⅰ：基础知识

Core Java 2, Volume Ⅰ: Fundamentals

（美） Cay S.Horstmann
Gary Cornell 著

程峰 黄若波 章恒翀 译

机械工业出版社
China Machine Press

本书覆盖了Java 2平台标准版1.4的基础知识，内容包括：面向对象的程序设计、反射和代理、接口和内部类、事件监听器模型、使用Swing GUI工具包的图形用户界面设计、异常处理、流输入/输出和对象的序列化。

本书内容全面、深入浅出，附有大量程序实例，极具实用价值。本书为Java 初学者和Java程序员提供了很好的指导。

本书版权登记号：图字：01-2003-0996

图书在版编目（CIP）数据

Java 2核心技术（第6版）. 卷Ⅰ，基础知识 /（美）霍斯特曼（Horstmann, C. S.）等著；程峰等译. -北京：机械工业出版社，2003.10

（Sun公司核心技术丛书）

书名原文：Core Java 2, 6e, Volume Ⅰ: Fundamentals

ISBN 7-111-12543-6

Ⅰ. J… Ⅱ. ①霍… ②程… Ⅲ. Java语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2003）第080754号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：李 英 贾 梅

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003年10月第1版第1次印刷

787mm × 1092mm 1/16 · 44.5印张

印数：0 001-5 000册

定价：75.00元

译 者 序

《Core Java》是Java编程方面一本享有盛誉的著作，从1996年第一版开始，这本书就畅销不衰，被评论家认为“对有经验的程序员来说，是当仁不让的最佳选择，虽然它讲述的是高深的技巧，却非常容易理解。”（K. N. King《Computing Reviews》）。因此，当我们被委托对《Core Java 2》第6版卷I进行翻译时，既感觉荣幸，更感觉是个难得的学习机会。

经过艰苦的翻译工作，我们感觉这本书的成功名至实归。它由浅入深地一步步介绍了Java基本知识，涵盖了Java语言的几乎全部的基本技术。对于没有什么编程经验的初学者，通过本书可以比较轻松地了解和掌握Java技术。对于有Java编程经验的程序员，本书既可以作为一个很好的语言参考手册，也可以帮助他们了解Java的最新版本（JDK1.4）提供的新特性。对于有C/C++编程经验的程序员，本书中对于Java和C/C++的对比可以使他们更加容易了解和掌握Java语言。作者于循循善诱中，把Java的要点讲解得清楚而透彻。这一版增加了正则表达式、新的I/O API、断言、环境设置选择配置、Swing的增强、日志等等JDK1.4新增的功能。总之，《Core Java》的确能够帮助我们解决那些极具挑战性的任务！

这本书由黄若波、程峰、章恒翀进行翻译。其中，程峰负责第1到第8章的翻译，章恒翀负责第9章，黄若波翻译第10到12章，并负责全书的统稿。俗话说：“三个臭皮匠，顶个诸葛亮”。译者不敢和诸葛亮相比，只求尽心尽力，不使原书因为翻译而失色过多。虽然诚惶诚恐，但由于时间和水平的限制，错讹之处，在所难免，恳请读者批评指正。

2003年6月于北京中关村

前　言

1995年底，Java程序设计语言闯入了Internet领域，并迅速占据了显著地位。Java的美好前景在于它将成为把用户和信息连接起来的“万能胶”而不管何种信息来源，如Web服务器、数据库、信息提供商、任何其他想像得出来的信息来源。实际上Java正处于实现这个前景的独特位置。它是一种非常可靠的设计语言，得到了除微软外的所有主要软件提供商的认可。Java内建的安全和保护特性打消了程序员和程序用户的顾虑，它甚至内建了对高级程序设计任务的支持，如网络编程、数据库连接和多线程等。

从1995年开始，Sun公司共发布了五个Java软件开发包（Java Software Development Kit）的主要修订版本。在过去的七年当中，应用编程接口（Application Programming Interface, API）从大约200个类扩充到3000多个类。API跨越了多个领域，如用户界面构建、数据库管理、国际化、安全以及XML处理。

本书已更新到第6版。每个版本都尽可能快地跟上Java软件开发包的发布，并且每个版本都重新编写以使读者能从最新的Java特性中获益。

像这本书以前的版本一样，我们仍然着眼于希望将Java用于实际项目中的真正的程序员。我们仍然保证本书中不会出现模棱两可的文字和莫名其妙的字符。我们假定本书读者是具有坚实程序设计语言基础的程序员。但是你不必了解C++或面向对象程序设计。根据本书早期版本的反馈信息，我们相信有经验的Visual Basic、C或COBOL程序员阅读本书不存在困难。（读者甚至不需要有在Windows、UNIX或Macintosh下构建图形用户界面的经验。）

本书针对有以下想法的读者：

- 编写真正的代码来解决实际的问题。
- 不喜欢充斥着玩具例程的书（比如说厨房用具或果树）。

本书包含了大量的代码，几乎演示了我们讨论的所有语言和库的特性。我们有意使用简单的例程来突出要点，但是大多数例程既不是赝品，也没有偷工减料。它们能为你编写自己的代码开一个好头。

我们假定读者还渴望学习Java的所有高级特性。我们将提供如下内容的详细信息：

- 面向对象的程序设计
- 反射和代理
- 接口和内部类
- 事件监听器模型
- 使用Swing GUI工具包的图形用户界面设计
- 异常处理
- 流输入/输出和对象的序列化

我们没有在那些仅仅是为了丰富网页，虽然有趣但并不十分严肃的Java程序上花很多时间。这方面的书籍很多——我们推荐John Pew的《Instant Java》一书，它也是由Sun Microsystems出版社出版的。

最后，随着Java类库爆炸式地增长，单卷本的书已经无法涵盖真正的程序员必须掌握的Java所有特性。我们决定将这本书分为两卷。卷I，就是本书，集中讲述了Java语言的基础概念和用户界面编程的基础。卷II则更进一步讲述企业级特性和高级用户界面编程。它包括了如下内容的详细信息：

- 多线程
- 分布式对象
- 数据库
- 网络程序设计
- 集合类
- 高级图形编程
- 高级GUI组件
- 本地方法
- XML处理
- 国际化
- JavaBean

在写书时，错误和不准确难以避免，我们非常乐意知道这些错误。但是，我们当然也希望同一个问题只被告知一次。我们建立了一个网页，它包括常见问题（FAQ）、Bug修正和工作区，它的URL是：http://www. horstmann. com/corejava. html。FAQ的最后是用来报告bug和提出建议的表格（希望你能先浏览一遍FAQ）。如果我们没有回答每一个问题或者没有及时反馈，请不要失望。我们会认真地阅读所有来信，并感谢你的建议使得本书的将来版本更清晰且更有价值。

我们希望你觉得这本书生动有趣且能对你的Java程序设计带来帮助。

关于本书

第1章是对Java区别于其他程序设计语言的能力作一个总体描述。我们将解释这个语言设计者的设计初衷，以及在哪些方面实现了他们的设想。然后，我们简要叙述Java诞生和发展的历史。

在第2章中，我们将告诉你怎样从http://java. sun. com/j2se下载和安装Java SDK。我们也讲述了怎样从http://www. phptr. com/corejava下载和安装本书的程序例程。然后我们将向你示范如何编译和运行三种典型的Java程序：控制台应用程序、图形应用程序、applet。

第3章开始讨论Java语言。在这一章中，我们将涉及基础内容：变量、循环、简单函数。如果你是C或C++程序员，这会非常简单，因为Java语言的语法本质上和C相同。如果你没有C的编程背景但了解其他语言，如Visual Basic或COBOL，则应该仔细阅读这一章。

面向对象的程序设计（Object-Oriented Programming，OOP）是目前程序设计应用中的主流方法，而Java是完全面向对象的。第4章介绍了封装——面向对象的两个最基本的概念之一，以及Java语言对它的实现机制，也就是类和方法。除了Java语言规则外，我们还给出了正确运用OOP设计的建议。最后，我们介绍了javadoc这种非凡的工具，它能够把你的代码注释格式化成一组网页的超级链接。如果你熟悉C++，那么你可以快速浏览这一章。没有面向对象程序设计背

景的程序员在进一步学习Java前，应当花一些时间来掌握OOP的概念。

类和封装仅仅是OOP技术中的一部分，第5章介绍了另一个基本概念，也就是继承。继承可以利用一个已有的类并根据你的需要来修改它，这是Java程序设计中一项基本技术。Java中的继承机制与C++中的继承非常类似，C++程序员可以把重点放在两种语言的不同之处。

第6章教你怎样使用Java的接口（interface）概念。接口使你超越第5章的简单继承模型。掌握接口可以让你充分了解Java完全面向对象的方法的程序设计能力。这里我们也提到了Java的一项很有用的技术特征——内部类。内部类可以使你的代码更清楚、简洁。

在第7章中，我们开始细致讨论应用程序设计。我们将示范如何创建窗口，如何绘制它们，如何绘制几何图形，如何用多种字体格式化文本，以及如何显示图像。

第8章详细讨论了AWT（Abstract Window Toolkit，抽象窗口工具包）的事件模型（我们讨论的是Java1.1版中新加入的事件模型，不是1.0版中已经过时的和过于简单的事件模型）。你将看到如何编写响应事件（如按鼠标和击键）的代码。同时，你还将学会如何处理基本的GUI元素，如按钮和面板。

第9章非常详细地讨论了Swing GUI工具包。Swing工具包使你能够建立跨平台的图形用户界面。你将学习建立各种按钮、文本组件、边界、滑块、列表框、菜单以及对话框的所有技术。然而，一些更高级的组件将在卷Ⅱ中讨论。

完成了第9章后，你便具备了编写applet的所有基础，applet是可以嵌入到网页中的微型程序，第10章的主题就是applet。我们向你演示一些有用的也是很有趣的applet，但更重要的是我们把applet看作程序部署的一种方法。然后我们将描述如何把应用程序打包成JAR文件，如何用Java Web Start机制通过Internet发布应用程序。最后，我们解释部署Java程序以后，如何保存和取回它们的配置信息。

第11章讨论异常处理，即Java的健壮性机制，该机制用来处理这种情况：好的程序也会出现问题。例如，网络连接在文件下载中可能会变成不可用，磁盘可能会装满等等。异常处理是把正常运行的代码和错误处理分开的有效方法。当然，即使你在你的程序中加入了所有的异常处理代码，它仍然无法像预计的那样工作。这一章的第二部分我们将给出许多有用的调试技巧。最后，我们将引导你使用各种工具完成一个示例任务，这些工具包括JDB调试器、集成开发环境调试器、性能统计器、代码覆盖测试工具以及AWT机器人。

本书最后介绍输入/输出处理。在Java中，所有的I/O都通过所谓的流来处理。流让你用统一的方式来处理任何来源（如文件、网络连接或内存块）的数据。我们包括了详细的读取器和写入器类，它们使得Unicode的处理更为简单；而且我们还要介绍当你使用对象序列化机制时的底层实现，它使得存储和装入对象简单、方便。最后，我们探讨了几个新增到SDK 1.4的库："新的I/O"类，它支持更高级、更有效的文件操作，以及正则表达式库。

附录列出了Java语言的关键字。

约定

本书使用以下图标表示某些特殊内容：

本书中有许多C++注释用来解释Java和C++之间的区别。如果你没有C++背景，或者你觉得你的C++的编程经历对你来说像一场不愿提及的噩梦，那么可以略过这些内容。

为正文提供注释和提示。

提示容易出错的地方。

警告有危险的地方。

通常Java有一个很大的编程库或应用编程接口（API）。当第一次使用一个API调用时，我们会在这一节的结尾处加上一段简要描述，用API图标来表示。这些描述不太正式，但是我们希望它们能比正式的API联机文档提供更多的信息。我们现在给每一个API注释标注版本号，该版本号标示哪个版本引入该特性，用以帮助不使用Java风险版本的用户。

源代码

本书的Web站点http://www. phptr. com/corejava以压缩的方式包含本书的所有代码。要展开这些文件可以使用熟悉的解压工具或者简单地使用jar功能，这是Java软件开发包的一部分。参见第2章了解安装Java软件开发包和例程代码的详细信息。

致谢

写书通常是一项巨大工程，修订也不像看起来那么容易，尤其是Java技术一直在不断更新。出版一本书可以说是耗尽了许多人的心血，在此我衷心地感谢本书写作小组。

Prentice-Hall PTR公司，Sun Microsystems出版公司，Navta公司的许多人都提供了非常有价值的帮助，但他们却甘愿隐藏在幕后。我想让他们都知道我是多么地感谢他们。和往常一样，我要真诚地感谢我的编辑，Prentice-Hall PTR公司的Greg Doench，他从本书的写作到出版过程一直给予我指导和帮助，同时还感谢所有与本书相关的幕后人物。我还要感谢早期版本中我的合作者，Gary Cornell，他已经转向其他事业。

感谢早期版本的许多读者，他们报告了许多错误并提出了很多有建设性的建议，这对我们的改进有着极大的帮助。我还要特别感谢本书优秀的修订小组，他们仔细地审阅我的手稿，使我少犯许多尴尬的错误。

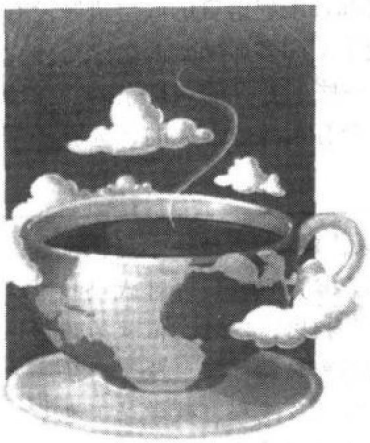
对本书及其早期版本进行评论的专家包括：Chuck Allison (Contributing Editor, C/C++ Users Journal), Alec Beaton (PointBase, Inc.), Joshua Bloch (Sun Microsystems), David Brown, Dr. Nicholas J. De Lillo (Manhattan College), Rakesh Dhoopar (Oracle), David Geary, Angela Gordon (Sun Microsystems), Dan Gordon (Sun Microsystems), Rob Gordon, Cameron Gregory (olabs.com),

Marty Hall (The Johns Hopkins University Applied Physics Lab), Vincent Hardy (Sun Microsystems), Vladimir Ivanovic (PointBase, Inc.), Jerry Jackson (ChannelPoint Software), Tim Kimmet (Preview Systems), Chris Laffra, Charlie Lai (Sun Microsystems), Doug Langston, Doug Lea (SUNY Oswego), Gregory Longshore, Bob Lynch, Mark Morrissey (The Oregon Graduate Institute), Mahesh Neelakanta (Florida Atlantic University), Paul Philion, Blake Ragsdell, Stuart Reges (University of Arizona), Peter Sander (ESSI University, Nice, France), Paul Sevinc (Teamup AG), Devang Shah (Sun Microsystems), Bradley A. Smith, Christopher Taylor, Luke Taylor (Valtech), George Thiruvathukal, Kim Topley, Janet Traub, Peter van der Linden (Sun Microsystems), Burt Walsh.

Cay Horstmann

2002年于圣弗朗西斯科（旧金山）

目　录



第1章 Java简介

- 作为编程工具的Java
- Java的优点
- Java“白皮书”中的关键词汇
- Java和Internet
- Java简史
- 关于Java的常见误解

很长一段时间以来，随便打开一本计算机杂志，都会看到关于Java的内容。即便是像《纽约时报》、《华盛顿邮报》、《商业周刊》这样的主流报纸和杂志上也发表了大量有关Java的文章。CNN、CNBC这样家喻户晓的大众媒体也都在谈论着Java能做这个，Java能做那个。

不过，我们写这本书面向的是专业程序员，而且由于Java是一种严肃的编程语言，能够讨论的内容很丰富。所以，我们不会专注于对Java宣传的分析以试图找出那些天花乱坠的宣传后面的有限的事实，而是想详细介绍Java这种编程语言本身的知识（当然，其中包括那些使Java获得广泛关注的特性，即它在Internet上的应用）。毕竟，我们最终将会通过解释Java能做什么和不能做什么来区分那些狂热和现实。

在Java语言发展的早期，它的实际能力和广告宣传有着巨大的差距。随着Java的成熟，技术变得更加稳定和可靠，人们对Java的期待也变得更为合理。在本书写作之时，Java正被日益广泛地用于客户端和服务器资源（例如数据库）间通信的“中间件”。虽然还不是十分引人注目，但这是具有可移植性、多线程机制和网络能力的Java重要用武之地。Java还广泛地用于嵌入式系统开发，正逐渐成为开发手持设备、互联网信息站、车载计算机等应用的标准。不过，早期试图用Java重写常规PC程序的努力并不成功，开发出来的应用程序功能低下且速度缓慢。虽然在Java现在的版本中，一些这样的问题已得到解决，但这种努力仍然不值得鼓励，毕竟用户关心的是程序的功能和性能，而不是它用什么语言编写。我们认为Java的优势将来自于它在新设备和新应用领域中的应用，而不是对已有程序进行重写。

1.1 作为编程工具的Java

作为一种计算机语言，Java的宣传有些过火。不过，它确实是一种好的编程语言。对于真正的程序员来说，Java毫无疑问是一种好的选择。我们认为Java本来有可能成为一种伟大的编程语言，只是现在看来时机已经错过。当一种语言应用于某个领域时，它同现存代码兼容的严峻问题也随之而来。而且，即使变化可能没有破坏现存的代码，语言（包括Java）的设计者不可能向后一坐，轻松地说：“好了，看来我们在X上犯了错误，用Y会更好一些”。总之，虽然我们期望随着时间会有一些改进，但基本上Java语言的结构是不会有什么改变了。

既然如此，那么Java的巨大改进又来自何处呢？答案是虽然Java编程语言的底层机制没有改变，但Java库发生了巨大变化。随着时间的进展，Sun公司改变了从许多库函数的名字（使它们变得更一致）到图形工作机制（改变了事件处理模型并重写了一些部分）的一切东西，并增加打印等Java1.0不具备的重要功能。这么做的结果就是产生了一个有用的、强大的、远胜于早期版本的Java编程平台。

微软发布的J++产品和Java具有家族关系。和Java类似，J++也通过一个虚拟机解释执行，该虚拟机在执行Java字节码时同Java虚拟机兼容，但在同外部代码接口时两者有明显的区别。J++的基本语法和Java几乎完全相同，不过，微软为该语言增加的语言结构除了和Windows API接口外，其他功能值得怀疑。除了语法一样之外，J++和Java的基础库（字符串、工具类、网络、多线程、数学等）也基本相同。不过，在图形库、用户界面、远程对象访问上两者完全不同。目前，微软不再支持J++，而是推进了另外一种语言C#。C#在很多方面和Java类似，但它使用了不同的虚拟机，本书将不讨论有关J++和C#的内容。

1.2 Java的优点

Java的一个显著优点就是运行时环境提供了平台无关性：你可以在Windows、Solaris、Linux或其他操作系统上使用完全一样的代码。这点对于在各种不同平台上运行从Internet上下载的程序来说很有必要。

Java的另一个优点在于它具有和C++类似的语法。这使得C和C++程序员可以很容易地学习Java。当然，Visual Basic程序员可能会为语法头疼一下了。

如果你熟悉其他语言，而不熟悉C++，那么本节的一些术语可能会让你觉得比较陌生，那么就跳过本节好了。等到第6章结束，你会熟悉所有这些术语。

Java还是完全面向对象的——甚至比C++还要面向对象。除了数字之类的几个基本类型，Java中的一切都是对象（面向对象编程已经取代了早期的结构化技术，因为它在处理复杂项目时具有很多优势。如果你不熟悉面向对象编程，也不用担心，第3章到第6章提供了你需要了解的内容）。

然而，Java绝不仅仅是某种经过改善的C++方言。很关键的一点在于：用Java开发没有bug的代码比用C++要简单得多。

原因何在呢？Java的设计者仔细研究了是什么使得C++代码这么容易出现bug。他们为Java增加了一些特性，来消除出现常见bug的可能性：

- Java设计者取消了手工内存分配和回收。在Java中，内存是自动进行垃圾收集的。你永远不必担心会出现内存崩溃现象。
- 他们引入了真正的数组而且取消了指针算法。你永远不必担心由于指针操作时出现的偏移错误而重写内存区域。

- 他们消除了在条件性语句中弄混赋值和相等测试的可能性。比如，你根本不能编译通过if (ntries = 3)....这样的语句。（Visual Basic程序员可能注意不到这种问题，但是请相信，在C/C++代码中，这种语句是导致常见的错误根源。）
- 他们消除了多重继承，替换为从Objective C中借鉴而来的新概念“接口”（interface）。接口能够实现多重继承的大部分功能，却没有管理多重继承层次关系带来的复杂性。（如果你不了解继承，第5章将对它进行讲解。）

Java语言规范是公开的。你可以在下面站点找到它：http://java.sun.com/docs/books/jls/html/index.html。

1.3 Java“白皮书”中的关键词汇

Java作者写了一本很有影响的白皮书，来解释他们的设计目标和完成情况。该书用如下11个关键词汇进行组织：

1) 简单	2) 可移植性	3) 面向对象	4) 解释型
5) 分布式	6) 高性能	7) 健壮性	8) 多线程
9) 安全	10) 动态	11) 体系结构中立	

在上一节中，我们已经涉及其中一些特点。本节，我们将：

- 通过引用白皮书中Java设计者的描述来总结每个关键词汇。
- 基于我们对Java当前版本的经验，告诉读者我们对每个关键词汇的理解。

写作本书时，白皮书的网址为http://java.sun.com/doc/language_environment。

1.3.1 简单

我们希望构建一个无需经过深奥专业训练就可以进行编程的系统，并且它要符合当前的标准惯例。因此，尽管我们发现C++不合适，但我们仍把Java设计成同C++尽可能的接近，以便使系统更容易理解。Java剔除了C++语言中一些很少使用、难以理解、易于混淆的特性，在我们看来，这些特性引发的麻烦远远多于带来的好处。

Java的语法实际上是C++语法的一个“纯净”版本。其中没有头文件、指针算法（甚至是指针语法）、结构、联合、操作符重载、虚基类等等（本书各处的“C++注意”段落介绍了Java和C++的更多差异）。然而，设计者没有试图纠正C++中所有的拙劣特性，比如对switch语句就没有做任何改动。如果你了解C++，那么你会发现转到Java语法很容易。

如果你习惯于可视化编程环境（比如Visual Basic），你会发现Java并不简单。其中有很多奇怪的语法（尽管可以很快熟悉它们）。更重要的是，在Java中，你必须自己做大量的编程工作。Visual Basic的魅力在于它的可视化设计环境几乎全自动地为应用程序提供了许多基础结构，而在Java中，同样的功能却要手工编写代码来实现，而且代码量通常还很大。不过，现在已经有了一些提供拖放风格程序开发的第三方开发环境。