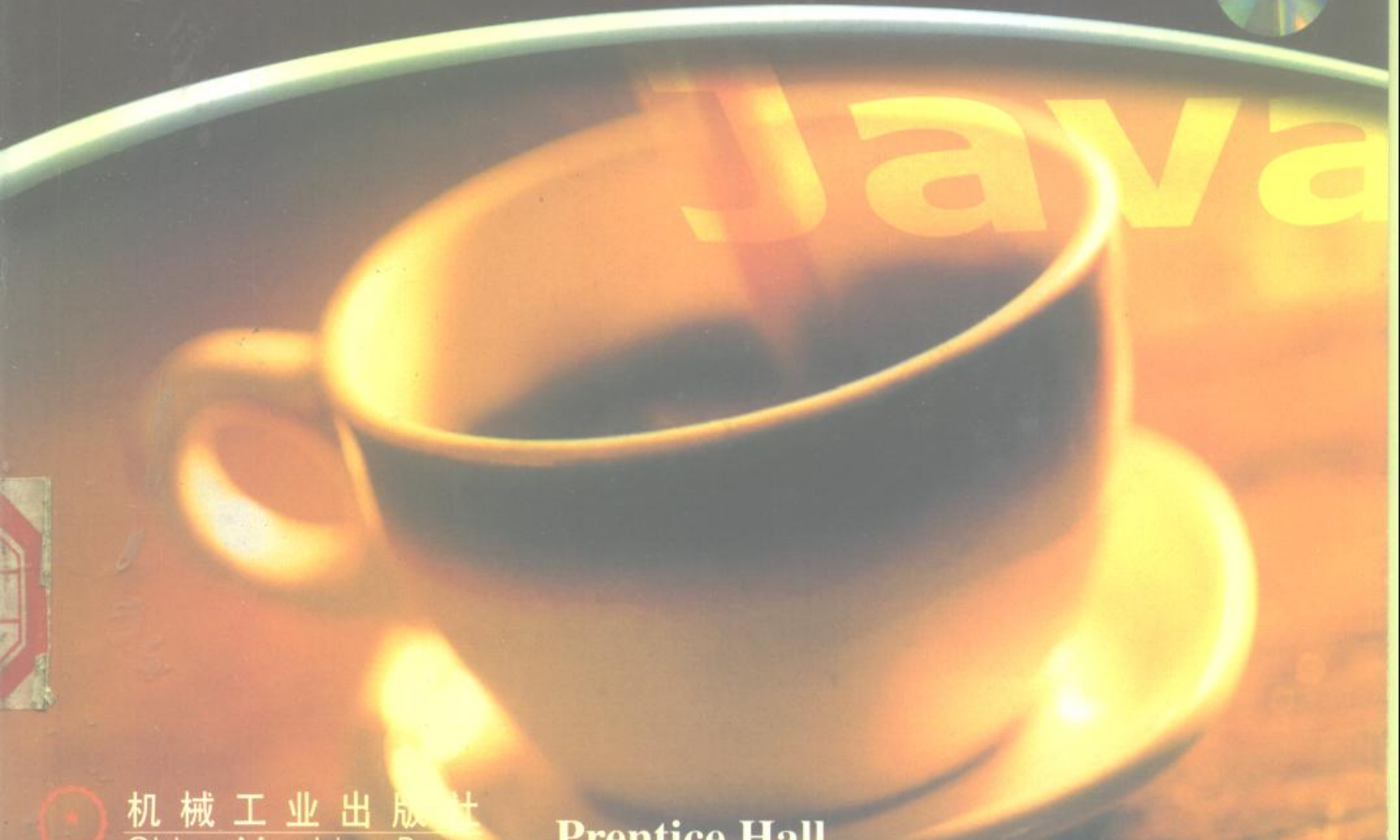
HZ BOOKS
软件开发技术丛书
Sun microsystems

Java 2 核心技术

卷 I: 基础知识

Core Java 2, Volume I: Fundamentals

（美）Cay S. Horstmann
Gary Cornell 著 京京工作室 译



机械工业出版社
China Machine Press
Prentice Hall

软件开发技术丛书

Java 2 核心技术
卷I：基础知识

（美） Cay S. Horstmann
Gary Cornell 著
京京工作室 译

机械工业出版社
China Machine Press

本书详细介绍Java语言的基础知识与主要功能，系统分析了Java语言的发展过程及成功因素，从Java语言的基本概念入手，阐述了Java语言的编程机制与技巧。本书配套光盘提供了最新Java开发包、共享软件、书中所有程序源代码等内容。本书既适合Java的初学者学习，又适合Java的编程高手参考。

本书版权登记号：图字：01-1999-2351

图书在版编目(CIP)数据

Java 2核心技术 卷I：基础知识/（美）霍斯曼（Horstmann,C.S.），（美）柯内尔（Cornell, G,）著；京京工作室译. - 北京：机械工业出版社，2000.1

（软件开发技术丛书）

书名原文：Core Java 2, Volume 1: Fundamentals

ISBN 7-111-07719-9

Ⅰ.J... Ⅱ.①霍...②柯...③京... Ⅲ.Java语言 - 程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999)第54115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：吴 怡

北京昌平第二印刷厂印刷 · 新华书店北京发行所发行

2000 年 1 月第 1 版 · 2000 年 5 月第 2 次印刷

787mm × 1092mm 1/16 · 33.25印张

印数：5 001-8 000 册

定价：68.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

前　言

1995年下半年，一个名为“爪哇”（Java）的编程语言闯入了Internet，立即受到了广泛关注，并随即赢得了良好名声。当时，一杯热腾腾的咖啡的图案简直成了Java的招牌广告。Java作出的最重要的承诺就是：它能成为一种“万能胶”，将用户同信息连接到一起，无论那些信息来自Web服务器、数据库、信息供应商还是能够想象到的其他任何一种信息源。事实上，Java是当时能真正实现这些许诺的唯一一种编程语言。由于拥有坚强的工程技术力量作为后盾，它获得了所有主要厂商的支持。其内建的安全及保密特性，可同时满足程序员和最终用户在这方面的要求。Java甚至内置了对许多高级编程任务的支持，如网络编程、数据库连接以及多线程等等。

从那时起，Sun Microsystems（太阳微系统公司）先后发布了Java语言三个重要的修订版本。其中，1996年发布1.02版，开始支持数据库的连接以及分布式对象。1997年发布1.1版，增加了健壮的事件模型、国际化支持以及Java Beans组件模型。而1.2版（即Java 2）发布于1998岁末，进行了多方面的改进。其中最值得一提的一项是“Swing”用户界面工具包！它使程序员终于能写出拥有真正移植能力的GUI（图形用户界面）应用程序。

本书是第4版。随着每一次新版Java开发包的发布，本书都会尽快修订；而且每一次，我们都会增加大量内容，以利用最新的Java特性。这一次也不例外。特别需要指出的是，书中所有示例都进行了更新，以便用到Swing工具包以及其他1.2特性。

同以前的几个修订版一样，本书的读者仍然是那些严肃的程序员，而这样的程序员希望在实际的项目中运用Java编程技术。我们仍然保证书中不会出现多余的废话，也不会出现令你神经紧张的字眼。我们希望读者是有一定资历的程序员，在编程语言方面已有相当多的经验。但是，这并不意味着读者必须懂得C++或者“面向对象的编程”。根据从以前版本的读者处收到的回音，我们仍然坚信，只要读者是一名有经验的Visual Basic、C、COBOL、Delphi或者PowerBuilder程序员，看懂本书都是一件轻而易举的事情，读者甚至不必具有在Windows、UNIX或Macintosh平台上构建图形用户界面的经验。

我们假定你阅读本书是出于下述目的：

- 编制实际代码，解决实际问题。
- 不喜欢那些充满花哨但不实用的例子的参考书（比如讲解Java在厨房电器或者水果树方面的应用）。

在本书的配套光盘中，你会发现大量示范程序，它们演示了我们讨论的几乎每一种语言及库特性。我们列举的示范程序都有一个特点，那就是只将重点放在主题上面，尽量做到不偏题。最重要的是，它们不花哨，并不脱离实际。在你编制自己的代码时，它们应当可起到很好的参考作用。

我们还假定你希望（甚至“渴望”）了解Java提供的所有高级特性。所以，我们详细讨论了下面这些主题：

- 面向对象的程序设计。
- Java的反射机制。
- 内部类。

- Java事件监听器模型。
- 用Swing GUI包完成图形用户界面设计。
- 违例控制（Java的错误控制机制）。
- 流输入／输出以及对象序列化。

在这一版中，我们仍然没在那些尽管有趣、但并不正统的Java程序上花费过多的时间，那些“东西”唯一的用途就是将你的Web页打扮得花枝招展！对于这方面的主题，市面上已有多本书籍详述。

最后，随着Java类库的爆炸性成长，再仅仅用一卷书来讲述Java的所有特性，已显得有点不太现实。因此，我们决定将出版两卷书。第1卷（即本书）将重点放在Java语言的基本概念上，同时附带讲述用户界面的编程基础。第2卷则将重点放在Java一些高级特性，以及高级的用户界面编程技术上面。第2卷主要讨论的内容包括：

- 多线程。
- 网络编程。
- 分布式对象。
- 容器类。
- 数据库。
- 高级图形。
- 高级GUI组件。
- 国际化。
- 固有方法。
- JavaBeans。

编写这样的一本书时，错误和不精确的地方在所难免。我们非常乐意知道这方面的问题，当然，每个错误也只想知道一次。为此，我们建设了一个主页，专门陈列本书的常见问题（FAQ）、错误修正以及解决方案等等，地址是：http://www.horstmann.com。FAQ的末尾是一份表格（希望你先看完FAQ的内容），可用它提交你发现的错误及问题。

希望本书能提高你的Java编程水平。

本书结构

第1章对Java的各种功能进行了概述，解释了它为何与其他编程语言不同。我们介绍了该语言的设计者的宗旨，以及他们在哪些方面取得了成功。随后，对Java的历史，以及Java的进化过程，作了一番简要的介绍。

在第2章，我们将指导你把Java以及本书的配套软件从配套光盘上安装到你的计算机。随后，详细讲述了如何编译及运行三个典型的Java程序：一个控制台应用、一个图形应用以及一个小应用程序（Applet）。

第3章将正式开始Java语言的讨论。在这一章中，我们将探讨一些基础知识，包括变量、循环以及简单的函数等等。如果你本来是一名C或C++程序员，整个过程将非常轻松，因为Java的语言特性和C基本上是相同的。但假如你本身没有C方面的知识，以前学的是Visual Basic或者Pascal/Delphi，本章便需仔细阅读。

面向对象的程序设计（OOP）目前是一种主流的编程技术，而Java是完全“面向对象”的。第4章将向大家介绍有关“封装”的知识，它是面向对象程序设计两大基础的第一个。同时还要介绍用于实现它的Java机制，即类与方法。除Java语言在这方面的规则以外，我们也提供了有

关正统的OOP设计的一些建议。如果你熟悉C++，这一章便可快速浏览通过。如果从前学习的是一种非面向对象的语言，便需在OOP上多花费些时间，才能继续学习Java一些深层次的东西。

类和封装仅属于OOP的一部分，第5章要向大家介绍的是另一部分：继承。通过“继承”，我们可以一个现成的类为基础，根据自己的需要进行修改。这是Java的一种基本编程技术。由于Java的继承机制类似于C++的继承机制，所以C++程序员可将重点放在对比两种语言的差异上面。

第6章向大家展示了在Java语言中如何使用“接口”这个概念。使用接口，我们就可在第5章介绍的“继承”模型基础上有所超越。只有完全掌握了接口的概念，才能充分运用Java的“面向对象”编程机制，编制出真正符合自己要求的程序。我们在本章也探讨了Java一项有用的技术特性，名为“内部类”。内部类有助于使代码更加清爽，同时也显得更加简洁。

自第7章开始，我们终于能一心一意地着手应用程序的编写。在此，我们将展示如何制作窗口、如何向其中填充内容、如何描绘几何图形、如何用多种字体对文本进行格式化，以及如何显示图像等等。

第8章将详细讨论AWT（抽象窗口工具包）的事件模型（在此讨论的是自Java1.1加入的事件模型，而非过时的、过于简单的1.0版事件模型）。大家将在这一章中学习如何编写代码，对鼠标点击或键盘按键这样的事件作出响应。同时，我们也会介绍如何对一些基本的GUI元素进行控制，诸如按钮及面板等等。

第9章则是本书的重头戏，以极大的篇幅、极其深入地讨论了Swing GUI工具包。通过这种工具包，我们可用Java构建出“跨平台”的图形用户界面。大家在此将学习到各式各样的按钮、文本组件、边框、滚动条、列表框、菜单和对话框等等。然而，仍有部分更高级的组件要等到第2卷才会讨论。

完成第9章的学习后，事实上已掌握了编写“小应用程序”所需的全部机制，所以第10章的主题便围绕这种小程序展开，它们实际上与Web页密不可分。我们在此展示了大量有用和有趣的小应用程序；但更重要的是，强调了如何才能超越这种“低级趣味”。我们讲述了如何使用Java插件，通过它来调用小应用程序，以便利用Java的所有最新特性——无论使用的是何种类型的Web浏览器。

第11章讨论了“违例控制”。这是Java可引以为荣的一种机制，用于应付那些即便最好的程序偶尔也会遇到的问题。例如，在一次文件下载的过程中，网络连接有可能中断，磁盘空间有可能用完等等。“违例”为我们提供了一种有效的机制，可将正常的处理代码同错误控制代码分隔开。当然，即便你的程序能够控制“所有”预料中的违例局面，仍有一些事情是完全预料不到的，所以它仍有可能出错。在本章的第二部分，我们将提供一些程序调试方面的建议。最后，我们将引导大家使用JDB调试程序，经历一次示范性的程序调试过程。尽管在真正干活的时候，JDB并不好用，但在理解了JDB之后，再换用专业开发环境提供的调试器，你就会发现自己如鱼得水，很快就能掌握。

第12章讨论输入与输出（I/O）控制的问题。在Java中，I/O是通过一种所谓的“流”来控制的。通过“流”，我们可以采用统一的办法，来实现同任何数据源的通信，比如文件、网络连接或者内存块等等。在这一章中，我们详细讨论了阅读器（Reader）和写入器（Writer）这两个类。使用这些类，可以非常轻松地操纵Unicode。同时，我们也展示了对象序列化机制的背后是个什么样的工作原理，该机制使得对象的保存与装载变得非常容易。

本书的附录为大家整理出了Java的关键字、Java自动归档（javadoc）机制以及配套光盘的安装步骤。

约定

正文中的“C++注意事项”和“VB注意事项”解释了Java和这些语言间的差异。如果读者对这两种语言不感兴趣，或者本来就没这方面的背景，跳过不读便是。

“注意”和“提示”指出了一些技巧和参考信息。

如果要指出一项潜在的危险，“警告”栏会提醒你注意。

Java配套提供了一个大型编程库，或称“应用程序编程接口”（API）。如果是首次用到一个API调用，我们便对其进行简要注释。虽然这种注释不太正式，但我们希望它比正式的联机API文档更具参考价值。

源码保存在配套光盘上的程序以示例的形式列出，比如：“例2-4 WelcomeApplet.java”。

配套光盘

本书配套CD-ROM提供了最新的Java开发包。截止本书完稿时为止，你还只能在Windows 9x/NT或Solaris 2平台上安装使用。

当然，光盘里包括了本书用到的所有源码，压缩成ZIP文件的形式。可用你喜欢的一个解压工具来解开文件，或直接使用Java开发包随附的jar工具。

光盘上也提供了一系列共享软件，可用它们辅助进行程序开发。请参考附录C，了解Java开发包、示范代码以及其他内容的具体安装及使用方法。

本书英文版书号为：ISBN 0-13-081933-6

原出版社站点地址：http: //www. phptr. com

目　录

第1章 Java入门

很久以来，随便打开一本计算机杂志，都可看到Java的身影。Java成为最热门的话题。甚至《纽约时报》、《华盛顿邮报》和《商业周刊》等一些主流报刊和杂志都刊载了许多关于Java的文章。看来Java形势大好（或者不好，不同的人有不同的看法）。还记得“国家公众广播电台”（National Public Radio）上次播出的有关计算机语言的那个10分钟的小故事吗？或者，还记得那项专为用一种“特定”计算机语言开发的产品设立的、高达一亿美元的风险投资基金吗？CNN、CNBC……您还可以列举出大量媒体，都对Java表现出了极大兴趣。仿佛人人都已经或将要达到非Java不谈的地步!

然而，我们在此决定为那些严肃的程序员编著本书。而且由于Java本身就是一种严肃的编程语言，还有许多地方不容忽视。所以，本书开篇，我们并没有立即深入剖析出现Java热的原因，或试图解决Java热现象幕后有限的（如果仍然有趣的话）一些真相，我们在本章只想向大家谈及Java作为一种编程语言的详细情况（当然，还提到了Java热产生以后，它对互联网产生的一些影响）。接下来，我们将对Java能够做到的和不能做到的事进行阐述，以便把人们从Java的幻想拉回到现实中来。

早期的Java，其实际能力与其被各媒体炒起来的Java热所说的有相当大的出入。随着日趋成熟，Java这项技术才慢慢变得更加稳定和可靠，人们对它的期望也越来越回落到合理的程度。我们撰写本书之时，Java正日益成为客户机、数据库和其他服务器资源之间进行通信的“中间件”。Java之所以受到众人瞩目，基本上源于其强大的移植能力、多线程处理和连网能力，这才是Java的魅力所在。Java正在进入嵌入系统，并已站稳脚跟，逐渐成为手持设备、互联网电子广告牌、车内计算机等等的一项标准。然而，早期用Java对熟悉的PC程序进行重新编程是非常费力不讨好的——做出来的应用程序不够强大，速度也很慢。但使用Java最新的版本，部分问题可以迎刃而解了，但仍有用户并不关心用哪种编程语言来写程序。我们认为Java的好处是来自新设备及其应用程序的，而非在于对现有程序进行改写。

1.1 作为编程工具的Java

作为一种编程语言，Java热确是炒过了头：事实上，Java的确是一种优秀的编程语言。毫无疑问，对严肃认真的编程人员来说，它是最适合他们的、最出色的编程语言之一。我们认为，它最终的确有可能成为一种出色的编程语言；但此时这样说，尚为时过早。一旦某种语言脱颖而出，就会产生与现有代码不兼容的问题。而且，在不破坏现有代码的情形下稍做改动也会如此，就连Java的缔造者也很难说：“哦，我们可能错用了X，Y可能要好一些”之类的话。总而言之，尽管我们预计会有一些改进，但大体上，Java语言以后的结构会和今天的大致相同。

说到这里，大家会提出一个显而易见的问题：Java那些显著的改进来自于何处呢？答案是不改动Java编程语言的基本根基，而是“对Java库做出重大改动”。Sun Microsystems对许多方面进行了修改，其中包括许多库函数的名字（使其更加连贯）；图形的工作方式（更改了事件控制模型，重新设计了其中的部件）；增加了Java 1.0没有的一些重要特性（比如打印）；等等。这样，Java就不仅仅是一个有用的编程工具，而且比其早期版本更健壮、更易于移植。

但Java的每个版本都会对老版本的库进行改动，这一次也不例外。事实上，在Java 2中，库的改动最大——新增的库函数大约比Java 1.1多一倍。

注意 微软正在研制一种叫作J++的产品，与Java家族有密切联系。和Java一样，J++的编译是通过一个虚拟机来实现的，这个虚拟机与用于执行Java字节码的Java Virtual Machine（Java虚拟机）兼容，但在与字节码的接口上，却存在重要的区别。其基本语法和Java大致相同。但是，微软也增加了自己的一些语言结构，除了Windows API接口之外，还有一些含糊的应用。Java和J++除共用一套语法外，其基本的库（字串、实用程序、连网、多线程和算术运算等等）也几乎完全相同。然而，用于访问图形、用户界面和远程对象的库却是全然不同的。除了要在第3到第6章讨论的通用语法之外，对于J++，本书将不再赘述！

1.2 Java的优点

Java一个最显而易见的优点是与平台无关的“运行时间”（Run Time）库：在Windows 95/98、NT、Solaris、Unix、Macintosh以及其他平台上，都可使用相同的代码。这一点对互联网编程来说，毫无疑问是必要的（但就其在Windows和Solaris之外的其他平台上的表现来说，则稍稍落后于它在Windows及Solaris平台上的表现。例如，我们在写作本书时，Java 2甚至还没有开始针对MAC平台的β测试）。

另一个编程方面的优点是Java的语法与C++的语法相似，这使得学习起来更有效率，而且很容易过渡。Visual Basic（VB）编程人员可能发现Java的语法令人有些恼火，而且Java抛弃了一些比较好的VB语法，比如“选择条件”（Select Case）等等。

注意 如果您以前使用的语言不是C++，某些章节中采用的术语也许对您比较生疏——只需跳过这些章节。到第6章之后，就会非常熟悉这些术语了。

Java也是完全面向对象的——甚至做得比C++更全面。在Java语言中，除极少数基本类型（比如数字）外的任何一个东西都是对象（面向对象设计已取代了早期的结构化编程，因为在处理复杂项目时，面向对象的程序设计要优越许多。如果你对面向对象的程序设计（OOP）不是很熟，那么只需经过第4章和第6章的学习，就会深刻理解到这一点）。

然而，如果只是将C++改头换面，然后推出一种新的语言，这种做法显然是不够的。最重要的一点是：使用Java比C++更易写出没有错误（bug-free）的代码。

为什么呢？Java设计者们绞尽脑汁去推究C++代码如此容易出错的原因。他们为Java增加了一些新特性，藉此排除构建代码时出现一些常见错误的可能性（据估计，现在大约每50行C++代码中，至少存在1个错误）。

- Java设计者消除了人工分配及取消分配内存的必要。Java中的内存是自动进行“垃圾收集”的。这样，您完全不必担心出现内存漏洞了。
- 他们引入了真正的数组，取消了指针算法。使用指针时，完全不必担心可能由于一次微不足道的出错，而导致改写一个关键的内存区。
- 他们排除了将赋值同条件语句中的一个相等性测试混淆起来的可能性。

您甚至可能无法编译if (ntries = 3). . . .（VB编程人员也许不会碰到这种情况；但请相信我们，它是在C/C++代码中造成的混乱的罪魁祸首）。

- 他们取消了多重继承，取而代之的是从面向对象的C（Objective C）衍生出来的新型接口概念。

通过接口，完全可以做到用多重继承能够做到的全部事情，而且不必牵涉到管理多重继承

结构时的复杂局面（如果继承对你来说有些陌生，第5章对此有详细论述）。

注意 Java语言的规范是公开的，可在其Web站点上找到它。可通过下面的链接，访问Java主页：http://java.sun.com。

1.3 Java的关键特点

Java的作者已写过一本颇有影响的“白皮书”（White Paper），详尽解释了他们设计该语言的初衷以及取得的成就。基本上可整理成下面这11项关键特点：

- 简单
- 可移植
- 面向对象
- 解释型
- 分布式
- 高性能
- 健壮
- 多线程
- 安全
- 动态
- 中性结构

我们会在最后一小节讲述其中的一些特点。在本小节，我们将：

■ 摘录白皮书中Java设计人员对各关键特点的一些解释，对它们进行大致总结。

■ 根据我们自己对Java最新版本的体验，和大家谈一谈我们对某些关键特点的看法。

注意 到本书完稿时为止，白皮书仍可在下述地址找到：
http://java.sun.com/doc/language_environment
如果现在找不到了，请转到Java主页，依链接前往。

1.3.1 简单

“我们想构建一个系统，它应当使编程更加容易，毋需进行大量的专业培训，而且影响到当今的标准。我们发现C++已不合潮流后，便开始着手Java的设计，使其尽可能地接近C++，目的是让这个系统更容易上手。Java省略了C++中所用的一些罕见的、难以理解的和极易混淆的特性。根据我们的体验，简单就是美，简单就是一切！”

说实在的，Java的语法就是C++语法的清错版本。没有了头文件、指针算法（甚至没有指针语法）、结构、单元、运算符过载、虚拟基础类等等（参见本书随处出现的“C++注意事项”，就可对Java和C++这两者之间的差别有更深的了解）。然而，Java的设计者们仍然没有对C++某些愚蠢的特性（如switch语句）进行改进。如果你熟悉C++，就会发现转向Java语法是个非常容易的过程。

但假如你是一名VB程序员，就会发现Java并不简单。Java中有许多陌生的语法（尽管掌握它并花不了多少时间）。更重要的是，在Java语言中，必须进行更多的编程。VB的妙处在于其可视化的编程环境，它为应用程序提供了大量基本构造，而且几乎是完全自动的。但在Java中，必须牵涉到大量代码。为实现同样的机能，往往需要大量的人工编程（请参见后文，讲述Sun Microsystems公司如何通过“JavaBeans”，为Java编程环境引入以组件为基础的“粘合”模型

——JavaBeans——它是用来开发即插即用组件的一种规范)。

但实际上，作为一种面向对象的编程语言，Java语言仍然是非常简单的。只是有些微妙之处还有待大家去体验，去解决实际应用中出现的问题。随着时间的推移，积累的经验便会越来越丰富，同时Java库及其开发环境也会日趋成熟。像Sun公司的Java WorkShop、Symantec的Visual Cafe和Inprise JBuilder等一系列产品，它们都提供了窗体设计器，有助于我们更方便地设计自己的程序界面。尽管它们距离完美还相当遥远，但确是一个可喜的进步（大家知道，用JDK来设计窗体非常乏味，效果非常之糟)。目前，网上正在涌现出越来越多的第三方类库(即由其他厂商提供的类库)，以及大量示范代码（包括许多库)，它们实际上大都是免费的!

“简单的另一方面就是‘小’。Java成功的秘诀之一就是采用了特殊的软件构造方法，可在小机器上独立运行。其基本的解释器以及类支持模块大概仅40KB。即便再加入基本的标准库，以及对线程的支持（特别是一个自主型的微内核)，也只需多加175KB。”

这无疑是一个极大的进步。但请注意，GUI库的体积要明显大得多!

1.3.2 面向对象

“简单地说，面向对象是着眼于数据（即对象）以及对象的接口的一种编程技术。以木匠为例，作为一名‘面向对象’的木匠，他的精力主要集中在自己正在做的椅子上，其次考虑用来制造它的工具；而作为一名“非面向对象”的木匠，他主要考虑的便是自己的工具。Java这一面向对象的特性在本质上与C++相同。”

在过去的30年间，“面向对象”的程序设计充分证明了自己的价值。一种现代的编程语言假如不具有这一特性，简直是不敢想象的！确实，Java的面向对象特性和C++存在许多共通之处。两者最主要的区别就是多重继承。为此，Java已找到了一种更好的解决办法——元类模型。通过反射机制（参见第5章）以及对象序列化特性（参见第2卷)，使其更容易实现永久性对象，而GUI构建器也更易集成现成的组件。

注意 如果还没有使用OOP语言的经验，就需要仔细阅读第4、5、6章。这三章详尽阐述了OOP是什么；为什么它比传统的、面向过程的语言（如Basic和C）更适合用来进行复杂项目的编程。

1.3.3 分布式

“Java提供了包容广泛的例程库，可处理像HTTP和FTP这样的TCP/IP协议。Java应用程序可通过一个特定的URL，来打开及访问对象，就像访问本地文件系统那样简单。”

我们已发现Java具备强大且易于使用的连网能力。任何人只要试过用另一种语言进行Internet编程，就会发现用Java来完成一些复杂的任务时，比如打开一个Socket连接，整个过程将是多么的容易！Java甚至使公共网关接口（CGI）的脚本编制更加容易。而另一种称作Servlet（服务器小应用程序）的机制，使得用Java进行服务器端编程时，可获得非常高的效率。许多流行的Web服务器都支持Servlet（我们将在本丛书的第2卷为大家讲述Java的连网能力)。远程方法调用机制则可实现分布式对象之间的通信（也将在第2卷讲述)。

1.3.4 健壮

“Java的目的是为了不管在何种条件下，都能保证编程绝对可靠。Java将大量重点放在早期的潜在问题检查、以后的动态（运行时间）检查以及避免出现潜在错误的情况上面……。Java与C++之间最重要、最大的差异是Java有一个指针模型，可避免改写内存和毁损数据的情

况发生。”

这个特性也非常有用。Java编译器（同时存在原始版本，以及由其他厂商提供的改进版本）可提前侦测到许多问题；而在其他语言中，这些问题只有在运行时间才会显露出来（或者干脆就隐藏下来了）。而就指针模型这一特性来说，任何人如果曾花数小时的时间来追踪由于一个指针错误而造成的内存破坏故障，就会非常高兴地看到Java提供了这一特性。

如果从前学习的语言（比如VB）并不明确使用指针，那么你可能还不理解Java这一特性的重要性。C程序员可就没那么幸运。他们需要不断地用指针来访问字串、数组、对象、甚至文件。在VB中，不必为任何一种这样的实体使用指针，也不必关心它们的内存分配问题。但另一方面，如果你在VB中使用一些比较生僻的数据结构，而且要求通过类模块来使用指针，就不得不自行管理内存。而Java同时具备这两种机制的优点。对于平时普通的构建任务，比如字串和数组的构建，就根本不需用到指针。而一旦需要，也完全能够获得指针的强大能力；比如在使用链接列表的时候。同时，我们可获得高度的安全保障，因为永远都不可能访问一个“坏掉”的指针；不可能造成内存分配错误；也毋需专门提防可能出现的内存漏洞。

1.3.5 安全

“Java将重点放在连网／分布式环境中的使用。为满足这个要求，我们将大量重点放在安全性上。通过Java可以轻松构建出防病毒、防黑客的系统。”

在“Core Java”第一版中，我们曾认为“永远都不要把话说绝”。事实证明我们是对的。普林斯顿大学的一组安全专家发现了Java 1.0安全特性中的第一例错误——就在JDK 1.0发布后不久。而且，他们和另外一些人又在Java的所有后续版本的安全机制中发现了其他错误。为此，我们建议大家经常访问下面的新闻组或者论坛，了解安全专家对于Java安全机制目前的状态的评价。

1) 普林斯顿大学新闻组的URL：http://www.cs.princeton.edu/sip/。

2) comp.risks新闻组。

好消息是Java开发组已经宣称，他们对系统安全方面的Bugs非常注重，会立即修复在小应用程序安全机制中发现的任何错误（浏览器公司也会立即着手工作）。特别要指出的是，由于Sun公开了Java解释器的内部工作细节，人们可以更加容易地发现Java安全机制中潜藏的问题——在外界人员的参与下，即使那些最不易察觉的问题，也会变得无所遁迹。采取这种做法，人们对Java能够尽快地发现错误有了更大的信心。不管在哪种情况下，都很难欺骗Java的安全机制。迄今为止发现的Bugs都非常细微，而且数量不多（相对来说）。

注意 Sun专门为安全方面的问题建立了一个网站，URL是：http://java.sun.com/sfaq/。

这儿有个例子，解释了在Java安全机制的保护下，如何能有效地防范一个Java程序干下面这些“坏事”：

1) 过载运行时间堆栈，就像那些臭名昭著的Internet蠕虫们干的好事!

2) 破坏自己进程空间以外的内存。

3) 通过一个已具备安全特性的类装载模块调用时，非法读写本地（本机）文件。如同一个进行了特殊编程的Web浏览器，它已经禁止了此类访问。

所有这些特性均已实现，而且大多数时候都能正常发挥功用。Java的确是目前最安全的一种程序设计语言。然而，我们也不可以对它的安全问题掉以轻心：尽管到目前为止发现的安全隐患都显得那么微不足道，而且安全漏洞的完整细节都不为世人所知，但这仍然不足以证明

Java“绝对安全”！在此，有必要再次重复我们一直以来的观点：

“永远都不要把话说绝！”

无论Java以前是否被证明安全，Java 1.1现在又提出了“签名类”（Signed Class）的概念（详情见本书第2卷）。通过签名类，我们可确定一个类是由谁写的。一旦签名机制投入正常运作，只要明确表示自己信任一个类的作者，那个类在自己的机器上就可拥有更大的特权。

注意 微软也提出了一套与之竞争的代码发行机制，它利用微软自己的ActiveX技术，以数字签名为基础，来确保程序代码的安全。但它做得显然还不够充分——因为使用微软产品的任何一名用户都可以证实：来自著名厂商的程序有时会发生莫名其妙的崩溃，而且完全可能造成对用户数据的破坏！Java拥有一个比ActiveX更为强壮的安全模型，因为它会在程序运行时加以控制，并在它试图造成破坏的时候将其中止（例如，即便是一个签名类，亦可随时禁止本机文件的输入及输出。主动权完全掌握在用户手中）。

1.3.6 中性结构

“编译器生成的是一种中性的对象文件格式——编译好的代码可在很多处理器中执行，只要对方安装了Java运行时间库。那么，Java编译器是如何做到这一点的呢？关键在于它生成的是特殊字节码指令，同任何一种特定的计算机体系都无关！它不仅在任何机器上都易于解释，也易于动态翻译成本机代码。”

这其实并不是一个新概念。大约在20年前，UCSD Pascal系统就已在一种商业产品中做到了这一点。甚至还能追溯到更早，早在Niklaus Wirth实现的Pascal中，采用的也是同样的机制。使用字节码（bytecode），性能当然会有一定程度的降低（大多数时候，准实时编译器可缓解这个问题）。Java的设计者在这方面进行了完美的发挥，他们设计的字节码指令集能够很好地兼容于当今大多数流行的计算机体系。代码经过特别设计，可非常方便地翻译成实际的机器指令。

1.3.7 可移植

“和C和C++不同，Java没有‘与具体实施有关’的概念。原始数据类型的大小是固定的，可应用于它们的算法亦是如此。”

例如，Java中的一个int肯定是个32位的整数。而在C/C++中，int可以是个16位整数，亦可是个32位整数，或者是编译器的作者乐意的其他任何一个大小的整数。唯一的限制是，int类型必须拥有和short int至少一样多的字节，而且不可多过long int指定的字节。而将数值类型的大小固定下来以后，就可轻松避开移植时可能出现的大量问题。二进制数据采用一种固定格式保存，消除了“big endian/little endian”的混淆。字串则用一种标准的Unicode格式加以保存。

“作为系统一部分的库定义了可移植接口。例如，我们提供了一个抽象Window类，并同时在Unix、Windows以及Macintosh平台上实现了它。”

编写出在Windows、Macintosh以及Unix的10种变种平台上都运行良好的一个程序，这几乎是每个开发者的梦想！Java 1.0首次将这个梦想变成了现实。它提供了一个简单的工具包，将标准的用户界面元素映射到为数众多的操作系统平台。但不幸的是，最后生成的库尽管已作了大量优化工作，但在不同系统上仍然会产生不可预料的结果（在不同平台的图形应用中，经常会出现莫名其妙的错误）。但这毕竟是一个良好的开端！对许多应用程序而言，良好的移植性才是第一重要的，所以它们未能从早期版本的Java中得到多少好处。而在新版本的Java中，用户界面包已进行了完全改写，使其不再对主机的用户界面有特别的依赖。所以同老版本的Java相比，