HZ BOOKS
PRENTICE HALL PTR

Sun公司核心技术丛书

JAVA 2

核心技术 卷Ⅰ：基础知识（原书第7版）

Core Java 2, Volume Ⅰ -Fundamentals Seventh Edition

（美） Cay S. Horstmann
Gary Cornell 著

叶乃文 邝劲筠 等译

- 为有经验的程序员提供Java程序设计语言的快速指南。
- 增加新的一章来介绍泛型程序设计。
- 涵盖自动打包、类型安全枚举、变量参数列表、静态导入、类型限定、通配符类型、Unicode 4.0支持以及其他的J2SE 5.0增强功能。
- 所有代码示例都针对J2SE 5.0进行了全面更新。

Sun microsystems

J2SE™ 5.0

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 卷 I：基础知识（原书第7版）

Core Java 2, Volume I -Fundamentals Seventh Edition

（美） Cay S. Horstmann
Gary Cornell 著

叶乃文 邝劲筠 等译

机械工业出版社
China Machine Press

J2SE™ 5.0

本书是Java技术经典参考书，多年畅销不衰，第7版在保留以前版本风格的基础上，涵盖Java 2开发平台标准版J2SE 5.0的基础知识，主要内容包括面向对象程序设计、反射与代理、接口与内部类、事件监听器模型、使用Swing UI工具箱进行图形用户界面设计、异常处理、流输入/输出和对象序列化、泛型程序设计等。

本书内容翔实、深入浅出，附有大量程序实例，极具实用价值，是Java初学者和Java程序员的必备参考书。

本书版权登记号：图字：01-2005-3050

图书在版编目（CIP）数据

Java 2核心技术，卷I：基础知识（原书第7版）/（美）霍斯特曼（Horstmann, C. S.），（美）科奈尔（Cornell, G.）著；叶乃文等译. －北京：机械工业出版社，2006.5

（Sun公司核心技术丛书）

书名原文：Core Java 2, Volume I- Fundamentals, Seventh Edition

ISBN 7-111-18523-4

Ⅰ.J…　Ⅱ.①霍…　②科…　③叶…　Ⅲ.Java语言－程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第012886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：隋　曦　　朱　劼

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2006年5月第1版第1次印刷

787mm × 1020mm 1/16 · 44.25印张

定价：88.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

《Java 2核心技术》自第1版出版以来，一直深受广大Java程序设计人员的青睐，是一本畅销不衰的Java经典书籍。本书的两位作者Cay S. Horstmann和Gary Cornell具有编写程序设计类书籍的丰富经验。

众所周知，Java程序设计语言仍处于不断完善和发展的活跃时期，为了能够及时跟上Java 的前进步伐，在不到10年的时间里，本书已经修订了6版，第7版仍然是为了适应Java的最新特性而编写的。在这一版中，主要增加了对JDK 5.0特性的介绍，其中包括面向对象程序设计、反射与代理、接口与内部类、事件监听器模型、使用Swing UI工具箱进行图形用户界面设计、异常处理、流输入/输出和对象序列化以及泛型程序设计等内容。

我们诚心地向广大读者推荐这本书。它囊括Java 2平台标准版（J2SE）的全部基础知识。作为一本精炼的技术指南和可信赖的参考书籍，其中提供了大量的应用实例，用来说明Java的重要语言规则和库功能，而且，这些例子程序都是完整且具有实际意义的。最重要的是，所有程序都已经被升级为J2SE 5.0，它们将成为你编写Java程序的良好开端。

参加本书翻译的有叶乃文、邝劲筠、俞晖和段勐。书中文字与内容力求忠实原著，但限于译者水平，加上时间仓促，翻译过程中难免有疏漏之处，敬请广大读者批评指正。

译 者

前　言

致读者

1995年底，Java程序设计语言在Internet舞台初一亮相便名声大噪。其原因在于它有望成为连接用户与信息的万能胶，而不论该信息来自Web服务器、数据库、信息提供商，还是任何其他的信息源。事实上，就发展前景而言，Java的地位是独一无二的。它是一种完全可信赖的程序设计语言，获得了除微软之外的所有主要厂商的认可。其固有的可靠性与安全性不仅令Java程序员放心，也令使用Java程序的用户放心。Java内建了对网络编程、数据库连接、多线程等高级程序设计任务的支持。

1995年以来，Sun Microsystems公司已经发布了Java开发工具箱（Java Development Kit，JDK）的6个主要版本。在过去的9年中，应用程序编程接口（API）已经从200个类扩展到3000个类，并跨越了诸如用户界面构建、数据库管理、国际化、安全性以及XML处理等各个不同的领域。发布于2004年的JDK 5.0是自Java的最初版本之后更新最显著的版本。

本书是《Java 2核心技术》第7版的卷I。自《Java 2核心技术》出版以来，每个新版本都尽可能跟上Java开发工具箱发展的步伐，而且每一版我们都重写部分内容以适应Java的最新特性。在这一版中，将极力展示泛型集合、增强的for循环以及一些令人兴奋的JDK 5.0特性。

和本书的前几版一样，我们仍然把读者群定位在那些打算将Java应用到实际项目中去的严谨的程序设计人员。我们仍然确保本书不会出现令人沮丧的文字以及莫名其妙的字符。本书假设读者是一名具有程序设计语言坚实的背景知识的程序设计人员。然而，你可以不必了解C++或是面向对象程序设计。根据我们收到的本书前几版的反馈意见，我们确信具有使用Visual Basic、C以及COBOL经历的程序员可以顺利阅读本书。（他们甚至不需要具有建立Windows、UNIX或Macintosh图形用户界面的任何经验。）

我们假定读者想要：

- 编写实际的代码来解决实际的问题。
- 不希望本书中全部是玩具式样例（如烤面包机、水果或是动物园的动物）。

在本书中，我们用大量的例子代码演示所讨论的每一种语言和库的特性。我们有意使用简单的示例程序以突出重点，然而，它们中的大部分既不是赝品也没有偷工减料。它们将成为读者自己编写代码的良好开端。

我们假设读者希望（甚至渴望）学习Java的所有高级特性。我们将会详细介绍下列内容：

- 面向对象程序设计
- 反射与代理
- 接口与内部类
- 事件监听器模型
- 使用Swing UI工具箱进行图形用户界面设计

- 异常处理
- 流输入/输出和对象序列化
- 泛型程序设计

随着Java类库的爆炸式增长，一本书无论如何也不能涵盖真正的程序员需要了解的所有Java特性。因此，我们决定将本书分为两卷。卷I（即本书）集中介绍Java语言的基本概念以及用户界面编程的基础。卷II涉及企业特性以及高级的用户界面编程，其中包含下列内容:

- 多线程
- 分布式对象
- 数据库
- 高级GUI组件
- 本地方法
- XML处理
- 网络编程
- 集合类
- 高级图形
- 国际化
- JavaBeans

在编写本书的过程中，难免会出现错误和不准确之处，我们很想知道这些错误。当然，我们也希望同一个问题只被告知一次。我们在网页http://www.horstmann.com/corejava.html中以列表的形式给出了常见问题、bug修正和出错位置。在勘误页（建议你先阅读一遍）最后是用来报告bug并提出改进意见的表单。如果我们不能回答每一个问题或没有及时回复，请不要失望。我们会认真阅读所有的e-mail，感谢你的建议使本书后续版本更清晰、更有指导价值。

我们希望你发现本书生动有趣且有助于Java程序设计。

关于本书

第1章将概述Java与其他程序设计语言不同的性能，解释这种语言的设计初衷，以及在哪些方面达到预期的效果。然后，简要叙述Java诞生和发展的历史。

在第2章中，将详细论述如何下载和安装JDK以及本书的程序样例。然后，通过编译和运行三个典型的Java程序（一个控制台应用、一个图形应用、一个applet），指导读者使用简易的JDK、可启用Java（Java-enabled）的文本编辑器以及一个Java IDE。

第3章开始讨论Java语言。这一章涉及的基础知识有变量、循环以及简单的函数。对于C或C++程序员来说，学习这一章的内容将会一帆风顺，因为这些语言特性的语法本质上与C语言相同。而对于没有C编程背景，但使用过其他编程语言（如Visual Basic）的程序员来说，仔细阅读这一章是非常必要的。

面向对象程序设计（Object-Oriented Programming，OOP）是当今程序设计的主流，而Java是完全面向对象的。第4章介绍面向对象两个基本成分中最重要的成分——封装，以及Java语言实现封装的机制，即类与方法。除了Java语言规则之外，我们还对如何正确地进行OOP设计给出了忠告。最后，介绍奇妙的javadoc工具，它将代码注释转换为超链接的网页。熟悉C++的程序员可以

快速地浏览这一章。而没有面向对象编程背景的程序员，应在进一步学习Java之前花费一些时间了解OOP的有关概念。

类与封装仅仅是OOP中的一部分，第5章介绍另一部分——*继承*。继承使程序员可以使用现有的类，并根据需要进行修改。这是Java程序设计的基础。Java中的继承机制与C++的继承机制十分相似。C++程序员只需关注两种语言的不同之处即可。

第6章展示如何使用Java的*接口*。接口可以使你的理解超越第5章的简单继承模型。掌握接口的使用将可以获得Java完全面向对象程序设计的能力。我们还将介绍Java的一个很有用的技术特性——*内部类*。内部类可以使代码更清晰、更简洁。

在第7章中，将开始详细讨论应用程序设计。我们将展示如何制作窗口、如何在窗口中绘图、如何用几何图形绘画、如何用多种字体格式化文本以及如何显示图像。

第8章将详细讨论AWT（Abstract Window Toolkit，*抽象窗口工具箱*）的事件模型。我们将介绍如何编写代码来响应诸如鼠标点击或敲击键盘等事件。同时，我们还将介绍如何处理基本的GUI元素，比如按钮和面板。

第9章详细讨论Swing GUI工具箱。Swing工具箱允许建立一个跨平台的图形用户界面。我们将介绍如何建立各种各样的按钮、文本组件、边界、滑块、列表框、菜单以及对话框等。一些更高级的组件将在卷II中讨论。

在前9章中，讲述编写applet需要的基础知识。applet是可以嵌入网页中的微型程序，第10章的主题就是applet。在这一章中，我们将展示一些有用且有趣的applet，但更重要的是将applet看作*部署程序*的一种方法。然后，我们将描述如何将应用程序打包到JAR文件中，以及如何使用Java的Web Start 机制在Internet上发布应用程序。

最后，我们解释Java程序在部署之后如何存储和得到配置信息。

第11章讨论*异常处理*，即Java的健壮机制，它用于处理调试好的程序可能出现的意外情况。例如，网络连接在文件下载过程中可能断开、磁盘可能填满等。异常提供了一种将正常的处理代码与错误处理代码分开的有效的方法。当然，即使程序具有处理所有异常情况的功能，但依然可能无法按预计的方式工作。这一章的后半部分给出了大量实用的调试技巧。最后，我们讲述如何使用各种工具完成一个示例程序，这些工具包括JDB调试器、集成开发环境的调试器、剖析器、代码覆盖率测试工具以及AWT自动机。

第12章的主题是输入输出处理。Java中所有的I/O都是通过所谓的*流*来处理的。流可以让程序员用一致的方式与任何数据源（如文件、网络连接或内存块）进行通信。我们将详细介绍大量读取器和写入器类，使用这些类可以方便地处理Unicode代码。另外，我们介绍使用对象序列化机制可能会出现的一些情况，该机制使得对象的存储与加载非常容易。最后，我们还给出已经添加到JDK 1.4中的支持高级的、更有效的文件操作的“new I/O”类以及正则表达式库。

本书的最后一章是*泛型程序设计概述*。泛型程序设计是JDK 5.0的重要改进，它使程序拥有更好的可阅读性和安全性。在这里，展示如何使用强类型机制，而舍弃不安全的强制类型转换。

附录A列出Java语言的保留字。

附录B介绍如何修改代码示例，使之可以在旧的编译器（JDK 1.4）下编译。

约定

本书使用以下图标表示特殊内容。


C++注意：在本书中有许多C++注释用来解释Java与C++的不同。对于没有C++编程背景，或者不擅长C++编程的读者，可以跳过这些注释。


注意：这个图标表示为正文提供的“注意”信息。


提示：这个图标表示为正文提供的“提示”信息。


警告：这个图标表示为正文提供的“警告”信息。


应用程序编程接口（API）

Java带有一个很大的程序设计库，即应用程序编程接口（Application Programming Interface，API）。第一次调用一个API时，我们将会在这一节的结尾给出一个概要描述，并标有API图标。这些描述十分通俗易懂，希望能够比联机API文档提供更多的信息。我们在每一个API注释特性上都标记了版本号，可提示那些不希望使用Java“风险”版本的读者。

程序源代码按照以下格式列出：

例2-4　WelcomeApplet.java

示例代码

本书的网站http://www.phptr.com/corejava⊖以压缩的形式提供了书中所有示例代码。你可以用相应的解压缩程序或者用Java开发工具箱中的jar实用程序展开这个文件。有关安装Java开发工具箱和示例程序的详细信息请参考第2章。

致谢

写一本书需要投入大量的精力，重写也并不像想象的那样轻松，尤其是Java技术一直在不断地更新。编著一本书让很多人耗费了很多心血，在此衷心地感谢Java核心技术编写小组的每一位成员。

Prentice Hall PTR和Sun Microsystems出版公司的许多人提供了非常有价值的帮助，却甘愿做幕后英雄。在此，我希望每一位都能够知道我对他们努力的感恩。和以往一样，我要真诚地感谢我的编辑，Prentice Hall PTR出版公司的Greg Doench，从本书的写作到出版一直给予了指导，同时感谢那些不知其姓名的为本书做出贡献的幕后人士。我还要感谢早期版本中我的合作者，Gary Cornell，他已经转向其他的事业。

感谢早期版本的许多读者，他们指出了许多令人尴尬的错误并给出了许多具有建设性的修改意见。我还要特别感谢本书优秀的审阅小组，他们仔细地审阅我的手稿，使我少犯了许多错误。

本书及早期版本的审阅者，他们是Chuck Allison (*C/C++ Users Journal*), Alec Beaton (PointBase,

⊖ 读者也可在华章网站（www.hzbook.com）下载相关代码。——编辑注

Inc.), Joshua Bloch(Sun Microsystems), David Brown, Dr. Nicholas J.De Lillo (曼哈顿学院), Rakesh Dhoopar (Oracle), David Geary (Sabreware), Angela Gordon(Sun Microsystems), Dan Gordon (Sun Microsystems), Rob Gordon, Cameron Gregory (olabs.com), Marty Hall (约翰·霍普金斯大学应用物理实验室), Vincent Hardy (Sun Microsystems), Vladimir Ivanovic (PointBase), Jerry Jackson (ChannelPointSoftware), Tim Kimmet (Preview Systems), Chris Laffra, Charlie Lai (Sun Microsystems), Doug Langston, Doug Lea (SUNY Oswego), Gregory Longshore, Bob Lynch (Lynch Associates), Mark Morrissey (The Oregon Graduate Institute), Mahesh Neelakanta (佛罗里达大西洋大学), Paul Philion, Blake Ragsdell, Stuart Reges (亚利桑那大学), 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 (*Core JFC*的作者), Janet Traub, Peter van der Linden (Sun Microsystems), Burt Walsh, Corky Cartwright, Frank Cohen (PushToTest), Chris Crane (devXsolution), Bill Higgins(IBM), Hang Lau (McGill University), Angelika Langer, Mark Lawrence, Dr. Paul Sanghera (圣何塞州立大学和布鲁克斯学院), Steve Stelting (Sun Microsystems)。

Cay Horstmann

2004年于旧金山

目　录

译者序
前言

第1章　Java程序设计概述 ……1
1.1　Java程序设计平台 ……1
1.2　Java“白皮书”的关键术语 ……1
1.2.1　简单性 ……2
1.2.2　面向对象 ……2
1.2.3　分布式 ……3
1.2.4　健壮性 ……3
1.2.5　安全性 ……3
1.2.6　体系结构中立 ……4
1.2.7　可移植性 ……4
1.2.8　解释型 ……5
1.2.9　高性能 ……5
1.2.10　多线程 ……5
1.2.11　动态性 ……5
1.3　Java与Internet ……6
1.4　Java发展简史 ……7
1.5　关于Java的常见误解 ……9
第2章　Java程序设计环境 ……13
2.1　安装Java开发工具箱 ……13
2.1.1　下载JDK ……13
2.1.2　设置执行路径 ……14
2.1.3　安装库源代码和文档 ……15
2.1.4　安装本书中的示例 ……16
2.1.5　导航Java目录 ……16
2.2　选择开发环境 ……17
2.3　使用命令行工具 ……17
2.4　使用集成开发环境 ……20
2.5　使用文本编辑器编译并运行程序 ……22
2.6　运行图形化应用程序 ……25
2.7　建立并运行applet ……27
第3章　Java基本的程序设计结构 ……30
3.1　一个简单的Java应用程序 ……30
3.2　注释 ……33
3.3　数据类型 ……34
3.3.1　整型 ……34
3.3.2　浮点型 ……35
3.3.3　char类型 ……35
3.3.4　boolean类型 ……37
3.4　变量 ……37
3.4.1　初始化变量 ……38
3.4.2　常量 ……39
3.5　运算符 ……39
3.5.1　自增运算符与自减运算符 ……40
3.5.2　关系运算符与boolean运算符 ……41
3.5.3　位运算符 ……41
3.5.4　数学函数与常量 ……42
3.5.5　数值类型之间的转换 ……43
3.5.6　强制类型转换 ……44
3.5.7　括号与运算符级别 ……44
3.5.8　枚举类型 ……45
3.6　字符串 ……46
3.6.1　代码点与代码单元 ……46
3.6.2　子串 ……47
3.6.3　字符串编辑 ……47
3.6.4　拼接 ……48
3.6.5　检测字符串是否相等 ……49
3.6.6　阅读联机API文档 ……51
3.7　输入输出 ……53
3.7.1　读取输入 ……54
3.7.2　格式化输出 ……56
3.8　控制流程 ……59
3.8.1　块作用域 ……60
3.8.2　条件语句 ……60

3.8.3 循环……63
3.8.4 确定循环……65
3.8.5 多重选择——switch语句……68
3.8.6 中断控制流程语句……70
3.9 大数值……72
3.10 数组……74
3.10.1 “for each”循环……75
3.10.2 数组初始化器以及匿名数组……76
3.10.3 数组拷贝……76
3.10.4 命令行参数……77
3.10.5 数组排序……78
3.10.6 多维数组……80
3.10.7 不规则数组……83
第4章 对象与类……86
4.1 面向对象程序设计概述……86
4.1.1 OOP词汇表……87
4.1.2 对象……88
4.1.3 类之间的关系……89
4.1.4 OOP与传统的过程化程序设计技术对比……90
4.2 使用现有类……91
4.2.1 对象与对象变量……92
4.2.2 Java库中的GregorianCalendar类……94
4.2.3 更改器方法与访问器方法……95
4.3 用户自定义类……100
4.3.1 一个Employee类……100
4.3.2 多个源文件的使用……103
4.3.3 解析Employee类……103
4.3.4 从构造器开始……104
4.3.5 隐式参数与显式参数……105
4.3.6 封装的优点……106
4.3.7 基于类的访问权限……108
4.3.8 私有方法……108
4.3.9 final实例域……109
4.4 静态域与静态方法……109
4.4.1 静态域……109
4.4.2 常量……110
4.4.3 静态方法……110
4.4.4 Factory方法……111
4.4.5 main方法……112
4.5 方法参数……114
4.6 对象构造……119
4.6.1 重载……119
4.6.2 默认域初始化……119
4.6.3 默认构造器……120
4.6.4 显式域初始化……121
4.6.5 参数名……121
4.6.6 调用另一个构造器……122
4.6.7 初始化块……122
4.6.8 对象析构与finalize方法……126
4.7 包……126
4.7.1 类的导入……127
4.7.2 静态导入……128
4.7.3 将类放入包中……129
4.7.4 虚拟机如何定位类……131
4.7.5 包作用域……134
4.8 文档注释……135
4.8.1 注释的插入……135
4.8.2 类注释……136
4.8.3 方法注释……136
4.8.4 域注释……137
4.8.5 通用注释……137
4.8.6 包与概述注释……138
4.8.7 注释的抽取……138
4.9 类设计技巧……139
第5章 继承……142
5.1 类、超类和子类……142
5.1.1 继承层次……148
5.1.2 多态……148
5.1.3 动态绑定……149
5.1.4 阻止继承：final类和final方法……151
5.1.5 强制类型转换……152
5.1.6 抽象类……154
5.1.7 受保护的访问……158
5.2 Object：所有类的超类……159
5.2.1 equals方法……160

5.2.2 相等测试与继承 ……161
5.2.3 hashCode方法 ……163
5.2.4 toString方法 ……164
5.3 泛型数组列表 ……169
5.3.1 访问数组列表元素 ……171
5.3.2 类型化与原始数组列表的兼容性 …175
5.4 对象包装器与自动打包 ……176
5.5 反射 ……179
5.5.1 Class类 ……180
5.5.2 使用反射分析类的能力 ……183
5.5.3 在运行时使用反射分析对象 ……187
5.5.4 使用反射编写通用的数组代码 ……191
5.5.5 方法指针 ……195
5.6 枚举类 ……198
5.7 继承设计技巧 ……199
第6章 接口与内部类 ……202
6.1 接口 ……202
6.1.1 接口的特性 ……207
6.1.2 接口与抽象类 ……208
6.2 对象克隆 ……209
6.3 接口与回调 ……213
6.4 内部类 ……216
6.4.1 使用内部类访问对象状态 ……217
6.4.2 内部类的特殊语法规则 ……220
6.4.3 内部类是否实用、必要和安全 ……220
6.4.4 局部内部类 ……222
6.4.5 匿名内部类 ……224
6.4.6 静态内部类 ……226
6.5 代理 ……229
第7章 图形程序设计 ……234
7.1 Swing概述 ……234
7.2 创建框架 ……237
7.3 框架定位 ……239
7.4 在面板中显示信息 ……243
7.5 2D图形 ……246
7.6 颜色 ……253
7.7 为文本设定特殊字体 ……256
7.8 图像 ……262
第8章 事件处理 ……267
8.1 事件处理基础 ……267
8.1.1 实例：处理按钮点击事件 ……269
8.1.2 建议使用内部类 ……273
8.1.3 将组件变成事件监听器 ……275
8.1.4 实例：改变观感 ……277
8.1.5 实例：捕获窗口事件 ……280
8.2 AWT事件继承层次 ……283
8.3 AWT的语义事件和低级事件 ……285
8.4 低级事件类型 ……287
8.4.1 键盘事件 ……287
8.4.2 鼠标事件 ……292
8.4.3 焦点事件 ……298
8.5 动作 ……302
8.6 多点传送 ……308
8.7 实现事件源 ……311
第9章 Swing用户界面组件 ……316
9.1 模型-视图-控制器设计模式 ……316
9.2 布局管理器概述 ……321
9.2.1 边界布局 ……322
9.2.2 面板 ……323
9.2.3 网格布局 ……324
9.3 文本输入 ……328
9.3.1 文本域 ……328
9.3.2 标签与标签组件 ……330
9.3.3 文本域变化跟踪 ……331
9.3.4 密码域 ……335
9.3.5 格式化的输入域 ……336
9.3.6 文本区 ……349
9.4 选择组件 ……352
9.4.1 复选框 ……353
9.4.2 单选按钮 ……355
9.4.3 边界 ……358
9.4.4 组合框 ……362
9.4.5 滑块 ……365
9.4.6 JSpinner组件 ……370
9.5 菜单 ……377
9.5.1 菜单创建 ……378

9.5.2 菜单项中的图标……380
9.5.3 复选框和单选按钮菜单项 ……381
9.5.4 弹出菜单 ……382
9.5.5 快捷键和加速器……383
9.5.6 启用和禁用菜单项……385
9.5.7 工具栏 ……389
9.5.8 工具提示 ……390
9.6 复杂的布局管理 ……393
9.6.1 箱式布局 ……395
9.6.2 网格组布局 ……398
9.6.3 弹簧布局 ……408
9.6.4 不使用布局管理器……415
9.6.5 定制布局管理器……416
9.6.6 遍历顺序 ……419
9.7 对话框……421
9.7.1 选项对话框 ……421
9.7.2 创建对话框 ……431
9.7.3 数据交换 ……434
9.7.4 文件对话框 ……440
9.7.5 颜色选择器 ……451
第10章 部署applet和应用程序 ……456
10.1 applet基础 ……456
10.1.1 一个简单的applet……457
10.1.2 查看applet ……458
10.1.3 将应用程序转换为applet ……460
10.1.4 applet的生命周期……461
10.1.5 安全基础 ……462
10.1.6 applet中的弹出式窗口 ……463
10.2 applet的HTML标记和属性……465
10.2.1 用于定位的applet属性 ……465
10.2.2 用于编码的applet属性 ……466
10.2.3 不支持Java兼容浏览器的applet属性……468
10.2.4 object标记……468
10.2.5 使用参数向applet传递信息 ……468
10.3 多媒体 ……473
10.3.1 封装URL ……473
10.3.2 获取多媒体文件 ……473
10.4 applet上下文 ……475
10.4.1 applet间的通信……475
10.4.2 在浏览器中显示信息 ……475
10.4.3 书签applet ……477
10.4.4 既是applet，又是应用程序 ……479
10.5 JAR文件 ……484
10.6 应用程序打包 ……485
10.6.1 清单文件 ……486
10.6.2 自运行JAR文件 ……486
10.6.3 资源……487
10.6.4 密封……490
10.7 Java Web Start ……490
10.8 应用程序配置的存储 ……502
10.8.1 属性映射 ……502
10.8.2 系统信息 ……506
10.8.3 Preferences API ……509
第11章 异常与调试 ……515
11.1 处理错误 ……515
11.1.1 异常分类 ……516
11.1.2 声明已检查异常 ……518
11.1.3 如何抛出异常 ……520
11.1.4 创建异常类 ……521
11.2 捕获异常 ……522
11.2.1 捕获多个异常 ……523
11.2.2 再次抛出异常与链异常 ……524
11.2.3 finally子句……525
11.2.4 堆栈跟踪元素分析 ……527
11.2.5 Java错误与异常处理 ……530
11.3 使用异常机制的建议 ……533
11.4 记录日志 ……537
11.4.1 基本日志 ……537
11.4.2 高级日志 ……537
11.4.3 修改日志管理器配置 ……539
11.4.4 本地化 ……540
11.4.5 处理器 ……541
11.4.6 过滤器 ……543
11.4.7 格式化器 ……544
11.5 使用断言 ……551

11.5.1 启用和禁用断言 ……552
11.5.2 使用断言的建议 ……553
11.6 调试技术 ……555
11.6.1 调试的常用技巧 ……555
11.6.2 使用控制台窗口 ……560
11.6.3 跟踪AWT事件 ……561
11.6.4 AWT的Robot类 ……565
11.7 使用调试器 ……568
11.7.1 JDB调试器 ……568
11.7.2 Eclipse调试器 ……573
第12章 流与文件 ……575
12.1 流 ……575
12.2 完整的流结构 ……577
12.2.1 流过滤器的分层 ……581
12.2.2 数据流 ……585
12.2.3 随机存取文件流 ……588
12.2.4 文本流 ……589
12.2.5 字符集 ……589
12.2.6 文本输出 ……595
12.2.7 文本输入 ……597
12.3 ZIP文件流 ……598
12.4 流的使用 ……605
12.4.1 分隔符输出 ……605
12.4.2 字符串记号处理器和带分隔符的文本 ……606
12.4.3 读取带分隔符的输入 ……607
12.4.4 StringBuilder类 ……611
12.4.5 随机存取流 ……612
12.5 对象流 ……617
12.5.1 存储可变类型的对象 ……617
12.5.2 理解对象序列化文件格式 ……620
12.5.3 保存对象引用问题的解决 ……624
12.5.4 理解对象引用的输出格式 ……629
12.5.5 修改默认的序列化机制 ……631
12.5.6 单元素与类型安全枚举的序列化 …633
12.5.7 版本 ……634
12.5.8 使用序列化进行克隆 ……636
12.6 文件管理 ……638
12.7 新的I/O ……643
12.7.1 内存映射文件 ……643
12.7.2 缓冲区数据结构 ……648
12.7.3 文件锁定 ……650
12.8 正则表达式 ……651
第13章 泛型程序设计 ……660
13.1 为什么要使用泛型程序设计 ……660
13.2 简单泛型类的定义 ……661
13.3 泛型方法 ……663
13.4 类型变量的限定 ……664
13.5 泛型代码和虚拟机 ……665
13.5.1 翻译泛型表达式 ……667
13.5.2 翻译泛型方法 ……667
13.5.3 调用遗留代码 ……669
13.6 约束与局限性 ……670
13.6.1 基本类型 ……670
13.6.2 运行时类型查询 ……670
13.6.3 异常 ……670
13.6.4 数组 ……671
13.6.5 泛型类型的实例化 ……671
13.6.6 静态上下文 ……672
13.6.7 擦除后的冲突 ……672
13.7 泛型类型的继承规则 ……673
13.8 通配符类型 ……674
13.8.1 通配符的超类型限定 ……675
13.8.2 无限定通配符 ……677
13.8.3 通配符捕获 ……677
13.9 反射和泛型 ……681
13.9.1 使用Class<T>参数进行类型匹配 …682
13.9.2 虚拟机中的泛型类型信息 ……682
附录A Java关键字 ……687
附录B 更新的JDK 5.0代码 ……689

第1章 Java程序设计概述

- ▼Java程序设计平台
- ▼Java发展简史
- ▼Java“白皮书”的关键术语
- ▼关于Java的常见误解
- ▼Java与Internet

1996年Java第一次发布就引起了人们的极大兴趣。关注Java的人士不仅限于计算机出版界，还有诸如《纽约时报》、《华盛顿邮报》、《商业周刊》这样的主流媒体。Java是第一种也是唯一一种在National Public Radio 上占用了10分钟时间进行介绍的程序设计语言，并且还得到了$100 000 000的风险投资基金。这些基金全部用来支持用这种特别的计算机语言开发的产品。重温那些令人兴奋的日子是很有意思的。在这一章中，将简要介绍Java语言的发展历史。

1.1 Java程序设计平台

在本书的第一版中，对Java是这样描写的：“作为一种计算机语言，Java的广告词实在有点夸大其辞。然而，Java确实是一种优秀的程序设计语言。对于一个名副其实的程序设计人员来说，使用Java无疑是一个好的选择。有人认为：Java将有望成为一种最优秀的程序设计语言，但还需要一个相当长的发展时期。一旦一种语言应用于某个领域，与现存代码的相容性问题就摆在了人们的面前。”

我们的编辑手中有许多这样的广告词。这是Sun公司高层的某位不愿透露姓名的人士提供的。然而，现在看起来，当初的这些预测还是有一定准确程度的。Java有许多非常优秀的语言特性，在本章稍后将会详细地讨论这些特性。由于相容性这个严峻的问题确实存在于现实中，所以或多或少会有一些“累赘”被添加到语言中，这就导致Java并不如想象中的那样完美无瑕。

但是，正像我们在第1版中已经指出的那样，Java并不只是一种语言。在此之前出现的那么多种语言也没有能够引起那么大的轰动。Java是一个完整的平台，有一个庞大的库，其中包含很多可重用的代码和一个提供诸如安全性、跨操作系统的可移植性以及自动垃圾回收等服务的执行环境。

作为一名程序设计人员，常常希望能够有一种语言，它具有令人赏心悦目的语法和易于理解的语义（C++不是这样的）。与许多其他的优秀语言一样，Java恰恰满足了这些要求。有些语言提供了可移植性、垃圾回收等等，但是，没有提供一个大型的库，如果想要有奇特的绘图功能、网络连接功能和数据库存取功能就必须自己动手编写代码。Java这种功能齐全的出色语言，具有高质量的执行环境以及庞大的库。正是因为它集多种优势于一身，所以对于广大的程序设计人员来说有着不可抗拒的吸引力。

1.2 Java“白皮书”的关键术语

Java的设计者已经编写了颇有影响力的“白皮书”，用来解释设计的初衷以及完成的情况，

并且发布了一个简短的摘要。该摘要用下面11个关键术语进行组织：

简单性	可移植性
面向对象	解释型
分布式	高性能
健壮性	多线程
安全性	动态性
体系结构中立	

在本节中，将论述下面两个问题：

- 给出白皮书中对每个关键术语的概述，这是Java设计者对相关术语的论述。
- 凭借Java当前版本的使用经验，给出对于这些术语的理解。

注意：白皮书可以在http://java.sun.com/docs/white/langenv/上找到。对于11个关键术语的概述请参看ftp://ftp.javasoft.com/docs/papers/java-overview.ps。

1.2.1 简单性

人们希望构建一个无需深奥的专业训练就可以进行编程的系统，并且要符合当今的标准惯例。因此，尽管人们发现C++不太适用，但在设计Java的时候还是尽可能地接近C++，以便系统更易于理解。Java剔除了C++中许多很少使用、难以理解、易混淆的特性。在目前看来，这些特性带来的麻烦远远大于带来的好处。

的确，Java语法是C++语法的一个“纯净”版本。这里没有头文件、指针运算（甚至指针语法）、结构、联合、操作符重载、虚基类等等。（请参看本书各个章节给出的C++注释，那里比较详细地解释了Java与C++之间的区别。）然而，设计者并没有试图清除C++中所有不适当的特性。例如，switch语句的语法在Java中就没有改变。如果熟悉C++就会发现可以轻而易举地将它转换成Java。

如果已经习惯于可视化的编程环境（例如Visual Basic），就不会觉得Java简单了。Java有许多奇怪的语法（尽管掌握其要领并不需要很长时间），更重要的是，使用Java需要自己编写大量的程序。Visual Basic的魅力在于它的可视化设计环境几乎自动地为应用程序提供了大量的基础结构。而使用Java实现同样的功能却需要手工地编制代码，通常代码量还相当大。然而，已经有一些支持“拖放”风格程序开发的第三方开发环境。

简单性的另一个方面是小。Java的目标之一是支持开发能够在小型机器上独立运行的软件。基本的解释器以及类支持大约仅为40KB；再加上基础的标准类库和对线程的支持（基本上是一个自含的微内核）大约需要增加175KB。

这是一个了不起的成就。然而，需要注意的是支持图形用户界面（GUI）的类库相当大。

1.2.2 面向对象

简单地讲，面向对象设计是一种程序设计技术。它将重点放在数据（即对象）和对象的接口上。用木匠来打一个比方，一个“面向对象的”木匠始终关注的是所制作的椅子，第二位才是所使用的工具；一个“非面向对象的”木匠则首先考虑的是所用的工具。在本质上，Java的面向对象能力与C++是一样的。

在过去的30年里，面向对象已经证明了自身的价值，一种现代的程序设计语言不使用面向对象技术简直让人难以置信。的确，Java的面向对象特性与C++旗鼓相当。Java与C++的主要不同点在于多继承，在Java中，取而代之的是简单的接口概念，以及Java的元类（metaclass）模型。反射机制（请参看第5章）以及对象序列化特性（请参看第12章）使得Java更加容易实现持久对象和GUI构建器（可以整合外来组件）。

注意：如果没有使用面向对象程序设计语言的经验，一定要仔细阅读第4章～第6章。这些章节解释了什么是面向对象程序设计以及为什么在编程实现复杂的项目时比传统的像C或Basic那样的面向过程的语言更加有效。

1.2.3 分布式

Java有一个扩展的例程库，用于处理像HTTP 和 FTP这类的TCP/IP协议。Java应用程序能够通过URL打开和访问网络上的对象，其便利程度如同访问本地文件系统一样。

人们已经看到Java的网络能力强大且易于使用。任何曾经试图使用其他语言进行网络编程的人都会惊呼Java竟然把类似打开socket连接这类繁重的任务都变得如此简单。（在本书的卷II中介绍网络连接。）另外，远程方法调用机制使得分布式对象之间可以进行通信。（也将在卷II中介绍。）

现在有一种独立的体系结构，Java 2企业版（J2EE），它支持大规模的分布式应用。

1.2.4 健壮性

Java的设计目标之一在于使用Java编写的程序具有多方面的可靠性。Java投入了大量的精力进行早期的问题检测、后期动态的（运行时）检测，并消除了有出错倾向的状态……Java和C++最大的不同在于Java采用的指针模型可以消除重写内存和损坏数据的可能性。

这个特性非常有用。Java编译器能够检测许多在其他语言中仅在运行时刻才能够检测出来的问题。至于第二点，对于曾经花费几个小时来检查由于指针bug而引起内存冲突的人来说，一定很喜欢Java的这一特性。

如果曾经只使用过Visual Basic这类没有显式指针的语言，就会感觉这么说似乎有些小题大做。然而，C程序员就没有这样幸运了。他们需要利用指针来存取串、数组、对象，甚至文件。在Visual Basic中，根本不必使用指针来访问这些实体，也不必关心有关内存分配的问题。另一方面，在没有指针的语言中，许多数据结构很难实现。Java具有双方的优势。它不需要使用指针构造诸如串、数组这样的结构。如果需要，也能够具有指针的能力，如链表。Java绝对是安全的，其原因是永远不会存取一个“坏的”指针，造成内存分配的错误，也不必防范内存泄漏。

1.2.5 安全性

Java适用于网络/分布式环境。为了达到这个目标，在安全方面投入了很大精力。使用Java可以构建防病毒、防篡改的系统。

在本书的第1版中，曾经说过：“永远不要把话说绝！”事实证明这是正确的。在Java开发工具箱第1版启用后不久，普林斯顿大学的一些安全专家们发现了在Java 1.0中的某些安全特性方