

JAVA

核心技术 卷 I : 基础知识

Core Java , Volume I : Fundamentals **Eighth Edition**

(美) Cay S. Horstmann 著
Gary Cornell

叶乃文 邝劲筠 杜永萍 译

- 针对Java SE6平台进行了全面更新。
- 涵盖Java语言核心内容。
- 大量精心设计代码示例。
- CSDN Java大版主隆重推荐。



机械工业出版社
China Machine Press

原书第8版

Sun 公司核心技术丛书

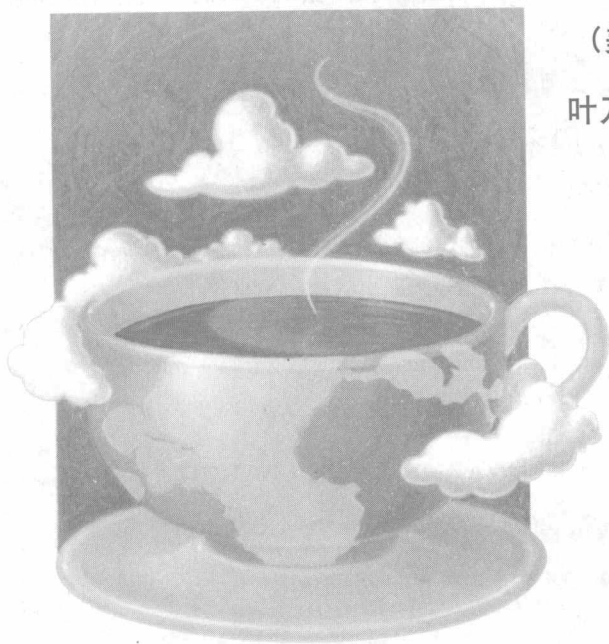
JAVA

核心技术 卷 I : 基础知识

Core Java , Volume I : Fundamentals **Eighth Edition**

(美) Cay S. Horstmann 著
Gary Cornell

叶乃文 邝劲筠 杜永萍 译



机械工业出版社
China Machine Press

原书第8版

《Java核心技术》出版以来一直畅销不衰，深受读者青睐，每个新版本都尽可能快地跟上Java开发工具箱发展的步伐，而且每一版都重新改写了部分内容，以便适应Java的最新特性。本版也不例外，它反映了Java SE 6的新特性。全书共14章，包括Java基本的程序结构、对象与类、继承、接口与内部类、图形程序设计、事件处理、Swing用户界面组件、部署应用程序和Applet、异常日志断言和调试、泛型程序设计、集合以及多线程等内容。

全书对Java技术的阐述精确到位，叙述方式深入浅出，并包含大量示例，从而帮助读者充分理解Java语言以及Java类库的相关特性。

Simplified Chinese edition copyright © 2008 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Core Java. Volume I, Fundamentals* (ISBN 978-0-13-235476-9) by Cay S. Horstmann, Gary Cornell. Copyright ©2008.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Sun Microsystems Press.

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。

版权所有，侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2008-2682

图书在版编目（CIP）数据

Java核心技术，卷I：基础知识（原书第8版）/（美）昊斯特曼（Horstmann, C. S.）著；叶乃文，邝劲筠，杜永萍译。—北京：机械工业出版社，2008.5

（Sun公司核心技术丛书）

书名原文：Core Java. Volume I, Fundamentals, Eighth Edition

ISBN 978-7-111-23950-5

I. J… II. ①昊… ②叶… III. JAVA语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字（2008）第053980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：李南丰 周茂辉

北京北制版厂印刷 · 新华书店北京发行所发行

2008年6月第1版第1次印刷

186mm×240mm · 44.5印张

标准书号：ISBN 978-7-111-23950-5

定价：98.00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核心技术》自第1版出版以来，一直备受广大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都具有编写程序设计方面书籍的丰富经验。

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，为了能够及时地跟上Java的前进步伐，在短短的10余年间，本书已经修订了7次，第8版同样是为了适应Java的最新特性而重新修订的。新版主要增加了对Java标准版（Java SE 6）特性的全面介绍，并对第7版中两卷的内容安排做了部分调整。即将第7版第I卷中的“流与文件”调到第II卷中，将第7版第II卷中的“集合”与“多线程”调到第I卷中。

我们诚心地向您推荐这本书籍。它囊括了Java 2平台、标准版（J2SE）的全部基础知识。作为一本精练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，这些示例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Java SE 6，它们将成为独自编写Java程序的良好开端。

参加本书翻译的有叶乃文、邝劲筠以及杜永萍。书中文字与内容力求忠实原著，但限于译者水平有限，加上时间仓促，译文中难免有疏漏之处，敬请广大读者批评指正。

译者

2008年4月于北京

前 言

致读者

1995年底,Java语言在Internet舞台一亮相便名声大噪。其原因在于它将有成为连接用户与信息的万能胶,而不论这些信息来自于Web服务器、数据库、信息提供商,还是任何其他渠道。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,得到了除微软之外的所有厂家的认可。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)的7个主要版本。在过去的11年中,应用程序接口(API)已经从200个类扩展到3000个类,并覆盖了用户界面构建、数据库管理、国际化、安全性以及XML处理等各个不同的领域。

本书是《Java核心技术》第8版的卷I。自《Java核心技术》出版以来,每个新版本都尽可能地跟上Java开发工具箱发展的步伐,而且每一版都重新改写了部分内容,以便适应Java的最新特性。在这一版中,已经反映了Java标准版(Java SE 6)的特性。

与前几版一样,本版仍然将读者群定位在那些打算将Java应用到实际工程项目中的程序设计人员。本书假设读者是一名具有程序设计语言(除Java之外)坚实背景知识的程序设计人员,并且不希望书中充斥着玩具式的示例(诸如:烤面包机、动物园的动物或神经质地跳动文本)。这些内容绝对不会在本书中出现。本书的目标是让读者充分地理解书中介绍的Java语言及Java类库的相关特性,而不会产生任何误解。

在本书中,我们选用大量的示例代码演示所讨论的每一个语言特性和类库特性。我们有意使用简单的示例程序以突出重点,然而,其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。

我们假定读者愿意(甚至渴望)学习Java提供的所有高级特性。本书将详细介绍下列内容:

- 面向对象程序设计
- 异常处理
- 反射与代理
- 泛型程序设计
- 接口与内部类
- 集合框架
- 事件监听器模型
- 并行操作
- 使用Swing UI工具箱进行图形用户界面设计

随着Java类库的爆炸式增长,一本书无法涵盖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(本书)集中介绍Java的基本概念以及图形用户界面程序设计的基础知识。卷II——高级特性,涉及企业特性以及高级的用户界面程序设计。其中包含下列内容:

- 文件与流
- 数据库
- 分布式对象
- 高级GUI组件

- 本地方法
- XML处理
- 网络编程
- 高级图形
- 国际化
- JavaBean
- 注释

在这一版中，我们对两卷中的内容进行了调整，特别是，鉴于多线程的日趋重要，我们将它编入卷I中，并以摩尔定律作为结尾。

在编写本书的过程中，难免出现错误和不准确之处。我们很想知道这些错误，当然，也希望同一个问题只被告知一次。我们在网页<http://horstmann.coni/coreJava.html>中以列表的形式给出了常见的问题、bug修正和出错位置。在勘误页（建议先阅读一遍）最后附有用来报告bug并提出修改意见的表单。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。我们会认真地阅读所有的来信，感谢您的建议使本书后续的版本更清晰、更有指导价值。

关于本书

第1章概述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了预期的效果。然后，简要叙述Java诞生和发展的历史。

第2章详细地论述如何下载和安装JDK以及本书的程序示例。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感觉一帆风顺，因为这些语言特性的语法本质上与C语言相同。对于没有C语言程序设计背景，但使用过其他程序设计语言（Visual Basic）的程序员，仔细地阅读这一章是非常必要的。

第4章介绍面向对象程序设计（Object-Oriented Programming, OOP）是当今程序设计的主流，而Java是完全面向对象的。本章将介绍面向对象两个基本成分中最重要的——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，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，而没有面向对象程序设计背景的程序员，应在进一步学习Java之前花一些时间了解OOP的有关概念。

第5章介绍类与封装仅仅是OOP中的一部分，本章将介绍另一部分——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程序设计中的基础。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。接口可以让你的理解超越第5章的简单继承模型。掌握接口的使用将可以获得Java完全的面向对象程序设计的能力。本章还将介绍Java的一个有用的技术特性——内部类。内部类可以使代码更清晰、更简洁。

第7章开始细致地讨论应用程序设计。每一个Java程序员都应该了解一些图形用户界面程序设计的知识，本卷中包含了其中的基本内容部分。本章将展示如何制作窗口、如何在窗口中绘图、如何用几何图形作画、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AWT（Abstract Window Toolkit）的事件模型。我们将介绍如何编写代码来响

应鼠标点击或敲击键盘等事件。同时，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。本章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、菜单以及对话框等等。一些更高级的组件将在卷II中讨论。

第10章阐述如何部署自己编写的应用程序或applet。在这里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，将解释Java部署之后如何存储、检索配置信息。

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的情况。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手段。当然，即使程序包含处理所有异常情况的功能，依然有可能无法按照预计的方式工作。这一章的后半部分，将给出大量的实用调试技巧。最后，讲述如何使用各种工具完成一个示例程序。

第12章概要介绍泛型程序设计，这是Java SE5.0的一项重要改进。泛型程序设计使得程序拥有更好的可阅读性和安全性。在这里，将展示如何使用强类型机制，而舍弃不安全的强制类型转换，以及如何处理与旧版本Java兼容而带来的复杂问题。

第13章介绍Java平台的集合框架。当需要将大量对象收集到一起，并在过后要对它们进行检索时，可能会想要使用集合，这是目前最为合适的，它取代了将这些元素放置在数组中。本章将介绍如何使用预先建立好的标准集合。

第14章是本书的最后一章。在这章中，将介绍多线程，这是一种可以让程序任务并行执行的特性（线程是程序中的控制流），并阐述如何建立线程、如何处理线程的同步问题。从Java SE 5.0开始，多线程有了很大的改进，本章将介绍所有这些新的机制。

附录列出了Java语言的保留字。

约定

本书使用以下图标表示特殊内容。



注释：这个图标表示为正文提供的“注释”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。



C++注释：在本书中有许多用来解释Java与C++的不同的C++注释。对于没有C++程序设计背景，或者不擅长C++程序设计的程序员，可以跳过这些注释。



应用程序编程接口

Java带有一个很大的程序设计库，即应用程序编程接口（API）。第一次调用API时，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，用来提示那些不希

望使用Java“风险”版本的读者。

程序源代码按照下列格式给出：

Listing 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的所有示例代码。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看第2章。

致 谢

写一本书需要投入大量的精力，改写一本书也并不是想像的那样轻松，尤其是Java一直在持续不断地更新。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。与以往一样，我要真诚地感谢我的编辑，Prentice Hall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。感谢Vanessa Moore提供了优秀的产品支持。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，使本书减少了许多错误。

本书及早期版本的审阅专家包括：Chuck Allison (《C/C++Users Journal》的特约编辑)，Alec Beaton (PointBase, Inc.)，Cliff Berg (iSavvix Corporation)，Joshua Bloch (Sun Microsystems)，David Brown, Corky Cartwright, Frank Cohen (PushToTest)，Chris Crane (devXsolution)，Dr. Nicholas J. De Lillo (曼哈顿学院)，Rakesh Dhoopar (Oracle)，David Geary (Sabreware)，Brian Goetz (Qiotix公司首席顾问)，Angela Gordon (Sun Microsystems)，Dan Gordon (Sun Microsystems)，Rob Gordon, John Gray (哈特福大学)，Cameron Gregory (olabs.com)，Marty Hall (约翰斯·霍普金斯大学应用物理实验室)，Vincent Hardy (Sun Microsystems)，Dan Harkey (圣约瑟州立大学)，William Higgins (IBM)，Vladimir Ivanovic (PointBase)，Jerry Jackson (ChannelPoint Software)，Tim Kimmet (Preview Systems)，Chris Laffra，Charlie Lai (Sun Microsystems)，Angelika Langer，Doug Langston，Hang Lau (麦吉尔大学)，Mark Lawrence，Doug Lea (SUNY Oswego)，Gregory Longshore，Bob Lynch (Lynch Associates)，Philip Milne (顾问)，Mark Morrissey (俄勒冈研究院)，Mahesh Neelakanta (佛罗里达大西洋大学)，Hao Pham，Paul Phillion，Blake Ragsdell，Stuart Reges (亚利桑那大学)，Rich Rosen (Interactive Data Corporation)，Peter Sanders (ESSI 大学，Nice, France)，Dr. Paul Sanghera (圣约瑟州立大学与布鲁克斯学院)，Paul Sevinc (Teamup AG)，Devang Shah (Sun Microsystems)，Bradley A. Smith，Steven Stelting (Sun Microsystems)，Christopher Taylor，Luke Taylor (Valtech)，George Thiruvathukal，Kim Topley (《Core JFC》的作者)，Janet Traub，Paul Tyma (顾问)，Peter van der Linden (Sun Microsystems)，and Burt Walsh.

Cay Horstmann

2007于旧金山

目 录

译者序
前言
致谢

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	2
1.2.1 简单性	2
1.2.2 面向对象	3
1.2.3 网络技能	3
1.2.4 健壮性	3
1.2.5 安全性	4
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	6
1.3 Java Applet 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	12
2.1 安装Java开发工具箱	12
2.1.1 下载JDK	12
2.1.2 设置执行路径	13
2.1.3 安装源代码库和文档	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 运行图形化应用程序	22

2.6 建立并运行applet	24
第3章 Java基本的程序设计结构	28
3.1 一个简单的Java应用程序	28
3.2 注释	31
3.3 数据类型	31
3.3.1 整型	32
3.3.2 浮点类型	32
3.3.3 char类型	33
3.3.4 boolean类型	35
3.4 变量	35
3.4.1 变量初始化	36
3.4.2 常量	36
3.5 运算符	37
3.5.1 自增运算符与自减运算符	38
3.5.2 关系运算符与boolean运算符	38
3.5.3 位运算符	39
3.5.4 数学函数与常量	40
3.5.5 数值类型之间的转换	41
3.5.6 强制类型转换	41
3.5.7 括号与运算符级别	42
3.5.8 枚举类型	43
3.6 字符串	43
3.6.1 子串	43
3.6.2 拼接	44
3.6.3 不可变字符串	44
3.6.4 检测字符串是否相等	45
3.6.5 代码点与代码单元	46
3.6.6 字符串API	47
3.6.7 阅读联机API文档	48
3.6.8 构建字符串	50
3.7 输入输出	51
3.7.1 读取输入	52

3.7.2 格式化输出	54	4.3.9 Final实例域	110
3.7.3 文件输入与输出	57	4.4 静态域与静态方法	110
3.8 控制流程	58	4.4.1 静态域	110
3.8.1 块作用域	59	4.4.2 静态常量	111
3.8.2 条件语句	59	4.4.3 静态方法	111
3.8.3 循环	62	4.4.4 Factory方法	112
3.8.4 确定循环	65	4.4.5 Main方法	113
3.8.5 多重选择: switch语句	68	4.5 方法参数	115
3.8.6 中断控制流程语句	70	4.6 对象构造	120
3.9 大数值	72	4.6.1 重载	120
3.10 数组	74	4.6.2 默认域初始化	121
3.10.1 For each循环	75	4.6.3 默认构造器	121
3.10.2 数组初始化以及匿名数组	76	4.6.4 显式域初始化	122
3.10.3 数组拷贝	76	4.6.5 参数名	123
3.10.4 命令行参数	78	4.6.6 调用另一个构造器	123
3.10.5 数组排序	79	4.6.7 初始化块	124
3.10.6 多维数组	82	4.6.8 对象析构与finalize方法	127
3.10.7 不规则数组	84	4.7 包	128
第4章 对象与类	87	4.7.1 类的导入	128
4.1 面向对象程序设计概述	87	4.7.2 静态导入	130
4.1.1 类	88	4.7.3 将类放入包中	130
4.1.2 对象	89	4.7.4 包作用域	133
4.1.3 识别类	89	4.8 类路径	134
4.1.4 类之间的关系	90	4.9 文档注释	136
4.2 使用现有类	91	4.9.1 注释的插入	136
4.2.1 对象与对象变量	91	4.9.2 类注释	137
4.2.2 Java类库中的GregorianCalendar类	94	4.9.3 方法注释	137
4.2.3 更改器方法与访问器方法	95	4.9.4 域注释	138
4.3 用户自定义类	101	4.9.5 通用注释	138
4.3.1 一个Employee类	101	4.9.6 包与概述注释	139
4.3.2 多个源文件的使用	104	4.9.7 注释的抽取	140
4.3.3 解析Employee类	104	4.10 类设计技巧	140
4.3.4 从构造器开始	105	第5章 继承	143
4.3.5 隐式参数与显式参数	106	5.1 类、超类和子类	143
4.3.6 封装的优点	107	5.1.1 继承层次	149
4.3.7 基于类的访问权限	109	5.1.2 多态	149
4.3.8 私有方法	109	5.1.3 动态绑定	151

5.1.4 阻止继承: final类和方法	152	6.5 代理	234
5.1.5 强制类型转换	154	第7章 图形程序设计	239
5.1.6 抽象类	155	7.1 Swing概述	239
5.1.7 受保护访问	160	7.2 创建框架	242
5.2 Object: 所有类的超类	160	7.3 框架定位	244
5.2.1 Equals方法	161	7.4 框架属性	246
5.2.2 相等测试与继承	162	7.5 决定框架大小	246
5.2.3 hashCode方法	164	7.6 在组件中显示信息	249
5.2.4 ToString方法	166	7.7 2D图形	253
5.3 泛型数组列表	171	7.8 颜色	260
5.3.1 访问数组列表元素	173	7.9 为文本设定特殊字体	263
5.3.2 类型化与原始数组列表的兼容性	176	7.10 图像	270
5.4 对象包装器与自动打包	177	第8章 事件处理	274
5.5 参数数量可变的方法	179	8.1 事件处理基础	274
5.6 枚举类	181	8.1.1 实例: 处理按钮点击事件	276
5.7 反射	182	8.1.2 建议使用内部类	280
5.7.1 Class类	183	8.1.3 创建包含一个方法调用的监听器	282
5.7.2 捕获异常	184	8.1.4 实例: 改变观感	283
5.7.3 利用反射分析类的能力	186	8.1.5 适配器类	286
5.7.4 在运行时使用反射分析对象	191	8.2 动作	290
5.7.5 使用反射编写泛型数组代码	195	8.3 鼠标事件	296
5.7.6 方法指针	198	8.4 AWT事件继承层次	302
5.8 继承设计的技巧	201	第9章 Swing用户界面组件	306
第6章 接口与内部类	204	9.1 Swing和模型-视图-控制器设计模式	306
6.1 接口	204	9.1.1 设计模式	306
6.1.1 接口的特性	209	9.1.2 模型-视图-控制器模式	307
6.1.2 接口与抽象类	210	9.1.3 Swing按钮的模型-视图-控制器分析	310
6.2 对象克隆	211	9.2 布局管理器概述	311
6.3 接口与回调	216	9.2.1 边框布局	313
6.4 内部类	219	9.2.2 网格布局	314
6.4.1 使用内部类访问对象状态	220	9.3 文本输入	318
6.4.2 内部类的特殊语法规则	223	9.3.1 文本域	319
6.4.3 内部类是否有用、必要和安全	224	9.3.2 标签和标签组件	320
6.4.4 局部内部类	226	9.3.3 密码域	321
6.4.5 由外部方法访问final变量	226	9.3.4 文本区	322
6.4.6 匿名内部类	229	9.3.5 滚动窗格	322
6.4.7 静态内部类	231	9.4 选择组件	325

9.4.1 复选框	325	10.3.1 一个简单的 applet	438
9.4.2 单选按钮	327	10.3.2 将应用程序转换为applet	440
9.4.3 边框	331	10.3.3 Applet的HTML 标记和属性	441
9.4.4 组合框	335	10.3.4 Object 标记	444
9.4.5 滑块	338	10.3.5 使用参数向applet传递信息	444
9.5 菜单	344	10.3.6 访问图像和音频文件	449
9.5.1 菜单创建	344	10.3.7 Applet上下文	450
9.5.2 菜单项中的图标	346	10.4 应用程序存储的配置	457
9.5.3 复选框和单选按钮菜单项	347	10.4.1 属性映射	457
9.5.4 弹出菜单	348	10.4.2 Preferences API	462
9.5.5 快捷键和加速器	349	第11章 异常、日志、断言和调试	468
9.5.6 启用和禁用菜单项	351	11.1 处理异常	468
9.5.7 工具栏	355	11.1.1 异常分类	470
9.5.8 工具提示	356	11.1.2 声明已检查异常	471
9.6 复杂的布局管理	359	11.1.3 如何抛出异常	473
9.6.1 网格组布局	360	11.1.4 创建异常类	474
9.6.2 组布局	369	11.2 捕获异常	475
9.6.3 不使用布局管理器	377	11.2.1 捕获多个异常	477
9.6.4 定制布局管理器	378	11.2.2 再次抛出异常与异常链	477
9.6.5 遍历顺序	382	11.2.3 Finally子句	478
9.7 对话框	383	11.2.4 分析堆栈跟踪元素	481
9.7.1 选项对话框	383	11.3 使用异常机制的建议	483
9.7.2 创建对话框	392	11.4 断言	486
9.7.3 数据交换	396	11.4.1 启用和禁用断言	487
9.7.4 文件对话框	402	11.4.2 使用断言的建议	487
9.7.5 颜色选择器	412	11.4.3 为文档使用断言	488
第10章 部署应用程序和applet	418	11.5 记录日志	489
10.1 JAR文件	418	11.5.1 基本日志	490
10.1.1 清单文件	419	11.5.2 高级日志	490
10.1.2 可运行JAR文件	420	11.5.3 修改日志管理器配置	492
10.1.3 资源	421	11.5.4 本地化	493
10.1.4 密封	423	11.5.5 处理器	494
10.2 Java Web Start	424	11.5.6 过滤器	496
10.2.1 沙箱	427	11.5.7 格式化器	497
10.2.2 签名代码	428	11.5.8 日志记录说明	497
10.2.3 JNLP API	430	11.6 调试技术	505
10.3 Applet	437	11.6.1 使用控制台窗口	510

11.6.2 跟踪AWT事件	511	13.2.4 树集	573
11.6.3 AWT的Robot类	515	13.2.5 对象的比较	574
11.7 使用调试器	519	13.2.6 队列与双端队列	579
第12章 泛型程序设计	523	13.2.7 优先级队列	580
12.1 为什么要使用泛型程序设计	523	13.2.8 映射表	581
12.2 简单泛型类的定义	525	13.2.9 专用集与映射表类	585
12.3 泛型方法	526	13.3 集合框架	589
12.4 类型变量的限定	527	13.3.1 视图与包装器	592
12.5 泛型代码和虚拟机	529	13.3.2 批操作	598
12.5.1 翻译泛型表达式	531	13.3.3 集合与数组之间的转换	599
12.5.2 翻译泛型方法	531	13.4 算法	599
12.5.3 调用遗留代码	533	13.4.1 排序与混排	601
12.6 约束与局限性	534	13.4.2 二分查找	603
12.6.1 不能用基本类型实例化类型参数	534	13.4.3 简单算法	604
12.6.2 运行时类型查询只适用于原始类型	534	13.4.4 编写自己的算法	605
12.6.3 不能抛出也不能捕获泛型类实例	535	13.5 遗留的集合	606
12.6.4 参数化类型的数组不合法	535	13.5.1 Hashtable 类	606
12.6.5 不能实例化类型变量	536	13.5.2 枚举	607
12.6.6 泛型类的静态上下文中类型 变量无效	537	13.5.3 属性映射表	608
12.6.7 注意擦除后的冲突	537	13.5.4 栈	608
12.7 泛型类型的继承规则	538	13.5.5 位集	609
12.8 通配符类型	540	第14章 多线程	613
12.8.1 通配符的超类型限定	541	14.1 线程的概念	613
12.8.2 无限定通配符	544	14.2 中断线程	623
12.8.3 通配符捕获	544	14.3 线程状态	625
12.9 反射和泛型	547	14.3.1 新生线程	626
12.9.1 使用Class<T>参数进行类型匹配	548	14.3.2 可运行线程	626
12.9.2 虚拟机中的泛型类型信息	549	14.3.3 被阻塞线程和等待线程	626
第13章 集合	554	14.3.4 被终止的线程	627
13.1 集合接口	554	14.4 线程属性	628
13.1.1 将集合的接口与实现分离	554	14.4.1 线程优先级	628
13.1.2 Java类库中的集合接口和迭代器接口	557	14.4.2 守护线程	629
13.2 具体的集合	561	14.4.3 未捕获异常处理器	629
13.2.1 链表	562	14.5 同步	631
13.2.2 数组列表	570	14.5.1 竞争条件的一个例子	631
13.2.3 散列集	570	14.5.2 详解竞争条件	635
		14.5.3 锁对象	636

14.5.4	条件对象	639	14.9	执行器	668
14.5.5	synchronized关键字	643	14.9.1	线程池	669
14.5.6	同步阻塞	646	14.9.2	预定执行	673
14.5.7	监视器概念	647	14.9.3	控制任务组	673
14.5.8	Volatile域	648	14.10	同步器	675
14.5.9	死锁	649	14.10.1	信号量	675
14.5.10	锁测试与超时	652	14.10.2	倒计时门栓	675
14.5.11	读/写锁	653	14.10.3	障栅	676
14.5.12	为什么弃用stop和suspend方法	654	14.10.4	交换器	676
14.6	阻塞队列	655	14.10.5	同步队列	677
14.7	线程安全的集合	661	14.10.6	例子: 暂停动画与恢复动画	677
14.7.1	高效的映像、集合和队列	662	14.11	线程与Swing	682
14.7.2	写数组的拷贝	663	14.11.1	运行耗时的任务	683
14.7.3	旧的线程安全的集合	663	14.11.2	使用Swing工作器	687
14.8	Callable与Future	664	14.11.3	单一线程规则	693

第1章 Java程序设计概述

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

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是第一种也是惟一的一种在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。本章将简要地介绍一下Java语言的发展历史。

1.1 Java程序设计平台

本书的第1版是这样描写Java的：“作为一种计算机语言，Java的广告词确实有点夸大其辞。然而，Java的确是一种优秀的程序设计语言。作为一个名副其实的程序设计人员，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性的。Java有许多非常优秀的语言特性，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，或多或少地还是有一些“累赘”被加到语言中，这就导致Java并不如想像中的那么完美无瑕。

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，其中包含了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收集等服务的执行环境。

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、垃圾收集器等等，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，所以对广大的程序设计人员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。这个摘要用下面11个关键术语进行组织：

简单性	可移植性
面向对象	解释型
网络技能 (Network-Savvy)	高性能
健壮性	多线程
安全性	动态性
体系结构中立	

本节将论述下列主要内容：

- 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，给出对这些术语的理解。



注释：白皮书可以在<http://java.sun.com/docs/white/langenv/>上找到。对于11个关键术语的论述请参看<http://java.sun.com/docs/overviews/java/java-overview-1.html>。

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多于其带来的好处。

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（请参阅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。如果知道C++就会发现可以轻而易举地将其转换成Java。

如果已经习惯于使用可视化的编程环境（例如Visual Basic），你就不会觉得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（基本上是一个自包含的微内核）大约需要增加175KB。

在当时，这是一个了不起的成就。当然，由于不断的扩展，类库已经相当庞大了。现在有一个独立的具有较小类库的Java微型版（Java Micro Edition）用于嵌入式设备。