HZ BOOKS
PRENTICE HALL

Sun公司核心技术丛书

# JAVA

# 核心技术 卷Ⅱ：高级特性

Core Java , Volume Ⅱ : Advanced Features Eighth Edition

（美）Cay S. Horstmann
Gary Cornell 著

陈昊鹏 王浩 姚建平 等译

- 针对Java SE 6平台进行了全面更新。
- 涵盖Java语言高级特性。
- 精心设计大量代码示例。
- CSDN Java大版主隆重推荐。

Sun microsystems

机械工业出版社
China Machine Press

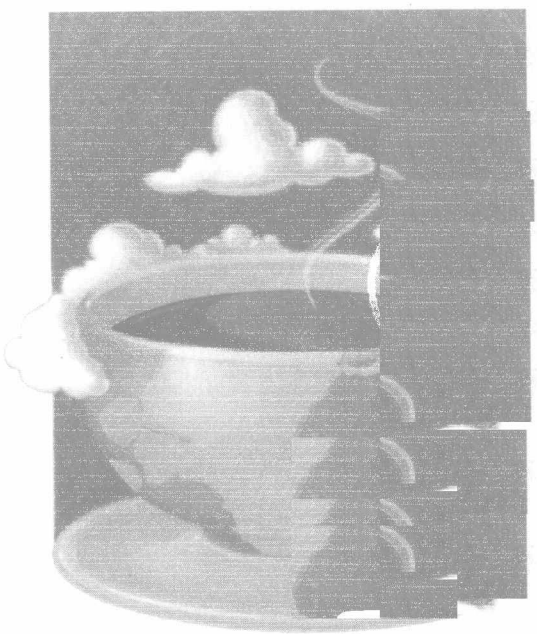
原书第8版

Sun公司核心技术丛书

# JAVA

# 核心技术 卷Ⅱ：高级特性

Core Java , Volume Ⅱ : Advanced Features Eighth Edition



(美) Cay S. Horstmann
Gary Cornell 著

陈昊鹏 王浩 姚建平 等译

机械工业出版社
China Machine Press

原书第8版

本书是Java技术权威指南，全面覆盖Java技术的高级主题，包括流与文件、XML、网络、数据库编程、高级Swing、高级AWT、JavaBean构件、安全、分布式对象、脚本、编译与注解处理等，同时涉及本地化、国际化以及Java SE 6的内容。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。

本书适合软件开发人员、高等院校教师和学生参考。



**本书版权登记号：图字：01-2008-3492**

**图书在版编目（CIP）数据**

Java核心技术，卷Ⅱ：高级特性（原书第8版）/（美）霍斯特曼（Horstmann, C. S.）等著；陈昊鹏等译. —北京：机械工业出版社，2008.12

（Sun公司核心技术丛书）

书名原文：Core Java, Volume II, Advanced Features, Eighth Edition

ISBN 978-7-111-25611-3

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

中国版本图书馆CIP数据核字（2008）第180687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：周茂辉　　陈佳媛

北京京北印刷有限公司印刷 · 新华书店北京发行所发行

2008年12月第1版第1次印刷

186mm × 240mm · 54.25印张

标准书号：ISBN 978-7-111-25611-3

定价：118.00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》的第8版又推出了，它已经在广大Java程序员和爱好者们当中产生了巨大的影响力。该书覆盖面广，几乎囊括了Java 2标准版的所有方面。其以接近实战的实例来展开内容的书写方式更是容易让读者理解和接受Java的精髓。

Java已经受到越来越多的程序员的青睐，但是Java语言包罗万象，而且其自身发展的速度更是惊人，我们在Sun的网站上几乎每个月都会看到有新的基于Java的规范出台。JDK 6.0使得Java又呈现出了新的面貌，其新添加的特性更好地能够应对Java在构建企业应用时所需要面对的挑战。因此，《Java核心技术》第8版在第7版的基础上，对JDK 6.0中的新特性进行了重点介绍，对原有部分章节进行了更新和调整，删除了一些过时的内容，并新增加了一些章节，而且对第7版的很多示例程序进行了调整，以此来使得程序员们能够更加透彻地理解和熟练地掌握这些新特性。

卷Ⅱ面向的是已经熟读并掌握了卷Ⅰ内容的读者，或者是已经对Java语言的基本特性相当熟悉的读者。卷Ⅱ的内容包含了流与文件、XML、网络、数据库编程、国际化、高级Swing、高级AWT、JavaBeans、安全、分布式对象、脚本编写、编译与注解处理，以及本地方法等内容，把读者引入了Java世界的更深处。

我们在翻译本书的过程中力求忠于原著。对于本书中出现的大量的专业术语尽量遵循标准的译法，并在有可能引起歧义之处注上了英文原文，以方便读者的对照理解。

全书的翻译由陈昊鹏、王浩、姚建平和龚斌合作完成，楼钢、李伟、郭嘉和方小丽对全书的翻译也做了大量的工作。由于译者水平有限，书中出现错误与不妥之处在所难免，恳请读者批评指正。

# 前言

## 致读者

您手中的这本书是按照Java SE 6完全更新后的《Java核心技术卷Ⅱ：高级特性（原书第8版）》。卷Ⅰ主要介绍了Java语言的一些关键特性；而本卷主要介绍编程人员进行专业软件开发时需要了解的高级主题。因此，与卷Ⅰ及之前的版本一样，我们仍将本书定位于为那些将Java技术运用于实际项目的编程人员提供帮助。

请注意：如果你是一个经验丰富的开发人员，能够灵活运用像内部类和泛型这样的高级语言特性，那么你就不需要阅读完卷Ⅰ再学习本卷。不过，本卷会根据适当情况去参考引用卷Ⅰ的有关内容，（我们希望读者会购买或者已经购买了卷Ⅰ，）当然，读者也可以在任何一本综合介绍Java平台的书中找到所需的背景知识。

最后要说明的一点是，编写任何一本书籍都难免会有一些错误或不准确的地方。我们非常乐意听到读者在本书中找到这方面的内容。当然，我们更希望对这些问题的报告只听到一次。为此，我们创建了一个FAQ、bug修正以及应急方案的网站http://www.horstmann.com/corejava。你可以在bug报告网页（该网页的目的是鼓励读者阅读以前的报告）的末尾处添加bug报告来发布bug和问题、给出建议，以便改进本书将来的版本。

## 内容提要

本书中的章节大部分是相互独立的。你可以研究最感兴趣的任何主题，并可以按照任意顺序阅读这些章节。

**第1章**的主题是输入输出处理。在Java中，所有I/O都是通过所谓“流”来处理的。流使你可以按照统一的方式来处理与各种数据源之间的通信，例如文件、网络连接或内存块。我们对各种读入器和写出器类进行了详细的讨论，它们使得对Unicode的处理变得很容易。我们还向你展示了：在使用对象序列化机制从而使保存和加载对象变得容易而方便时，其背后的原理是什么。最后，我们讨论了支持高效文件操作的“新I/O”类（它们曾作为最新内容添加到Java SE 1.4中）和正则表达式类库。

**第2章**介绍XML，介绍怎样解析XML文件，怎样生成XML以及怎样使用XSL转换。在一个实用示例中，我们将展示怎样在XML中指定Swing格式的布局。我们对该章进行了更新修正，将XPath API纳入其中，它使得“在XML的干草堆中寻找绣花针”变得更加容易。

**第3章**介绍网络API。Java使复杂的网络编程工作变得很容易实现。我们将介绍怎样创建连接到服务器上的网络连接，怎样实现你自己的服务器，以及怎样创建HTTP连接。

**第4章**介绍数据库编程，重点讲解JDBC，即Java数据库连接API，这是用于将Java程序与关系数据库进行连接的API。我们将介绍怎样通过使用JDBC API的核心子集，编写能够处理实际

的数据库日常操作事务的实用程序。（如果要完整介绍JDBC API的功能，可能需要编写一本像本书一样厚的书才行。）最后我们简要介绍了层次数据库，探讨了一下JNDI（Java命名及目录接口）以及LDAP（轻量级目录访问协议）。

**第5章**讨论了一个我们认为重要性将会不断提升的特性——国际化。Java编程语言是几种一开始就被设计为可以处理Unicode的语言之一，不过Java平台的国际化支持则走得更加深远。因此，你可以对Java应用程序进行国际化，使得它们不仅可以跨平台，而且还可以跨越国界。例如，我们会展示怎样编写一个退休金计算器的Applet，对它可以根据本地浏览器的情况使用英语、德语或者汉语进行浏览。

**第6章**涵盖了没有纳入卷 I 的所有Swing知识，尤其是重要但很复杂的树型构件和表格构件。随后我们介绍了编辑面板的基本用法、“多文档”界面的Java实现、在多线程程序中用到的进度指示器，以及诸如闪屏和支持系统托盘这样的“桌面集成特性”。我们仍着重介绍在实际编程中可能遇到的最为有用的构件，因为对Swing类库进行百科全书般的介绍可能会占据好几卷书的篇幅，并且只有专业编程人员才感兴趣。

**第7章**介绍Java 2D API，你可以用它来创建实际的图形和特殊的效果。该章还介绍了抽象窗口操作工具包（AWT）的一些高级特性，这部分似乎应该在卷 I 中做专门介绍。虽然如此，这些技术还是应该成为每一个编程人员工具包的一部分。这些特性包括打印和用于剪切粘贴及拖放的API。

**第8章**介绍了用于Java平台的构件API——JavaBean。你将会看到怎样编写自己的Bean，以及其他编程人员怎样在集成构建环境中对它们进行操作。最后我们展示怎样使用JavaBean的持久性，以某种与适用于长期存储的对象序列化不同的格式来存储自己的数据。

**第9章**继续介绍Java安全模式。Java平台一开始就是基于安全而设计的，该章会带你深入内部，查看这种设计是怎样实现的。我们将展示怎样编写用于特殊目的的应用的类加载器以及安全管理器。然后介绍允许使用消息、代码签名、授权以及认证和加密等重要特性的安全API。最后，我们用一个使用AES和RSA加密算法的示例进行了总结。

**第10章**介绍分布式对象。我们详细介绍了RMI（远程方法调用）。这个API可以让你运行分布在多台机器上的Java对象。然后简要讨论了Web Service，并给出了一个实现了Java程序和Amazon Web Service之间进行通信的示例。

**第11章**讨论了三种处理代码的技术。脚本机制和编译器API是在Java SE 6中引入的，它们允许程序去调用使用诸如JavaScript或Groovy之类的脚本语言编写的代码，并且允许程序去编译Java代码。可以使用注释向Java程序中添加任意信息（有时称为元数据）。我们将展示注释处理器怎样在源码级别或者在类文件级别上收集这些注释，以及怎样运用这些注释来影响运行时的类行为。注释只有在工具的支持下才有用，因此，我们希望我们的讨论能够帮助你根据需要选择有用的注释处理工具。

**第12章**介绍本地化方法，它可以让你调用为微软Windows API这样的特殊机制而编写的各种调用方法。很显然，这种特性具有争议性：使用本地化方法，那么Java平台的跨平台本质将会随之消失。虽然如此，每个为特定平台编写Java应用程序的严谨的编程人员都需要了解这些技术。有时，当你与不支持Java平台的设备或服务进行交互时，为了你的目标平台，你可能需

要求助于操作系统API。我们将通过展示如何从某个Java程序访问Windows注册表API来阐明这一点。

所有章节都按照最新版本的Java进行了修订，过时的材料都删除了，Java SE 6的新API也都详细地进行了讨论。

## 约定

我们使用等宽字体表示计算机代码，这种格式在众多的计算机书籍中极为常见。各种图标的含义如下：

**注意**：需要引起注意的地方。

**提示**：有用的提示。

**警告**：关于缺陷或危险情况的警告信息。

**C++注意**：本书中有一些C++注释，用于解释Java程序设计语言和C++语言之间的不同。如果你对这部分不感兴趣，可以跳过。

**应用编程接口**

Java平台配备有大量的编程类库或者应用编程接口（API）。当第一次使用某个API时，我们添加了一个简短的描述，并用一个API图标进行标识。这些描述可能有点不太规范，但是比起那些正式的在线API文档来说要更具指导性一点。

其源代码包含在与本书相关的代码中的程序都被作为示例程序而将其代码列举了出来。例如，

**程序清单11-1** ScriptTest.java

可以从网站http://horstmann.com/corejava[⊖]下载相关代码。

## 致谢

写书总是需要付出极大的努力，而重写也并不像看上去那么容易，特别是在Java技术方面，要跟上其飞快的发展速率，更是如此。一本书的面世需要众多有奉献精神的人共同努力，我非常荣幸地在此向整个《Java核心技术》团队致谢。

Prentice Hall出版社和Sun Microsystems出版社的许多人都提供了颇有价值的帮助，但是他们甘愿居于幕后。我希望他们都能够知道我是多么感谢他们付出的努力。与以往一样，我要热切地感谢我的编辑，Prentice Hall出版社的Greg Doench，他对本书从编写到出版进行全程掌舵，并使我可以十分幸福地根本意识不到幕后那些人的存在。我的感谢还要送给本书以前版本的合著者Gary Cornell，他后来转向其他具有挑战性的领域了。

⊖ 也可登录华章网站（http://www.hzbook.com）下载相关代码。——编辑注

我非常感谢找到了很多令人尴尬的错误并提出了许多颇具创见性的建议的早先版本的许多读者们。我特别要感谢十分出色的评审团队，他们用令人惊异的眼睛仔细浏览了所有原稿，并将我从许多令人尴尬的错误中拯救了出来。

这一版及以前版本是由以下人员评审的：Chuck Allison（特约编辑，《C/C++ Users Journal》）、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）、Robert Evans（资深教师，约翰霍·普金斯大学应用物理实验室）、David Geary（Sabreware）、Brian Goetz（首席顾问，Quiotix Corp.）、Angela Gordon（Sun Microsystems）、Dan Gordon（Sun Microsystems）、Rob Gordon、John Gray（Hartford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（圣何塞州立大学）、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（McGill 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 Philion、Blake Ragsdell、Ylber Ramadani（Ryerson 大学）、Stuart Reges（亚利桑那大学）、Rich Rosen（Interactive Data Corporation）、Peter Sanders（ESSI 大学，Nice, France）、Dr. Paul Sanghera（圣何塞州立大学和布鲁克学院）、Paul Sevinc（Teamup AG）、Devang Shah（Sun Microsystems）、Richard Slywczak（NASA/Glenn研究中心）、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）和Burt Walsh。

Cay Horstmann

旧金山2008

# 目 录

# 第1章 流与文件

- ▲ 流
- ▲ 文本输入与输出
- ▲ 读入和写出二进制数据
- ▲ ZIP文档
- ▲ 对象流与序列化
- ▲ 文件管理
- ▲ 新I/O
- ▲ 正则表达式

本章将介绍Java用于输入和输出的各种应用编程接口（Application Programming Interface，API）。你将要学习如何访问文件与目录，以及如何以二进制格式和文本格式来读写数据。本章还要向你展示对象序列化机制，它可以使存储对象像存储文本和数字数据一样容易。然后，我们将介绍在Java SE 1.4中引入的“新I/O”包java.nio所带来的种种改进。最后，本章将讨论正则表达式，尽管这部分内容实际上与流和文件并不相关，但是我们确实也找不到更合适的地方来处理这个话题。很明显，Java设计团队在这个问题的处理上和我们一样，因为正则表达式API的规格说明隶属于阐述Java SE 1.4的“新I/O”特性的规格说明。

## 1.1 流

在Java API中，可以从其中读入一个字节序列的对象称做输入流，而可以向其中写入一个字节序列的对象称做输出流。这些字节序列的来源地和目的地可以是文件，而且通常都是文件，但是也可以是网络连接，甚至是内存块。抽象类InputStream和OutputStream构成了有层次结构的输入/输出（I/O）类的基础。

因为面向字节的流不便于处理以Unicode形式（回忆一下，Unicode中每个字符都使用了多个字节来表示）存储的信息，所以从抽象类Reader和Writer中继承出来的专门用于处理Unicode字符的类构成了一个单独的层次结构。这些类拥有的读入和写出操作都是基于两字节的Unicode码元的，而不是基于单字节的字符。

### 1.1.1 读写字节

InputStream类有一个抽象方法：

```
abstract int read()
```

这个方法将读入一个字节，并返回读入的字节，或者在遇到输入源结尾时返回－1。在设计具体输入流类时，必须覆盖这个方法以提供适用的功能，例如，在FileInputStream类中，这个方法将从某个文件中读入一个字节，而System.in（这个InputStream的一个子类的预定义对象）却是从键盘读入信息的。

InputStream类还有若干个非抽象的方法，它们可以读入一个字节数组，或者跳过大量的

字节。这些方法都要调用抽象的read方法，因此，各个子类都只需覆盖这一个方法。

与此类似，OutputStream类定义了下面的抽象方法：

```
abstract void write(int b)
```

它可以向某个输出位置写出一个字节。

read和write方法在执行时都将阻塞，直至字节确实被读入或写出。这就意味着如果流不能被立即访问（通常是因为网络连接忙），那么当前的线程将被阻塞。这使得在这个方法等待指定的流变为可用的这段时间里，其他的线程就有机会去执行有用的工作。

available方法使我们可以去检查当前可用于读入的字节数量，这意味着像下面这样的代码片段就不可能被阻塞：

```
int bytesAvailable = in.available();
if (bytesAvailable > 0)
{
   byte[] data = new byte[bytesAvailable];
   in.read(data);
}
```

当你完成对流的读写时，应该通过调用close方法来关闭它，这个方法会释放掉十分有限的操作系统资源。如果一个应用程序打开了过多的流而没有关闭它们，那么系统资源将被耗尽。关闭一个输出流的同时也就是在清空用于该输出流的缓冲区：所有被临时置于缓冲区中，以便用更大的包的形式传递的字符在关闭输出流时都将被送出。特别是，如果不关闭文件，那么写出字节的最后一个包可能将永远也得不到传递。当然，我们还可以用flush方法来人为地清空这些输出。

即使某个流类提供了使用原生的read和write功能来工作的某些具体的方法，应用系统的程序员还是很少使用它们，因为大家感兴趣的数据可能包含数字、字符串和对象，而不是原生字节。

Java提供了众多从基本的InputStream和OutputStream类导出的类，这些类使我们可以处理那些以常用格式表示的数据，而不只是在字节级别上表示的数据。

**API** java.io.InputStream 1.0

- abstract int read()

  从数据中读入一个字节，并返回该字节。这个read方法在碰到流的结尾时返回－1。

- int read(byte[] b)

  读入一个字节数组，并返回实际读入的字节数，或者在碰到流的结尾时返回－1。这个read方法最多读入b.length个字节。

- int read(byte[] b, int off, int len)

  读入一个字节数组。这个read方法返回实际读入的字节数，或者在碰到流的结尾时返回－1。

  参数：b　　数据读入的数组

  　　off　　第一个读入字节应该被放置的位置在b中的偏移量

len 读入字节的最大数量

- long skip(long n)

  在输入流中跳过n个字节，返回实际跳过的字节数（如果碰到流的结尾，则可能小于n）。

- int available()

  返回在不阻塞的情况下可用的字节数（回忆一下，阻塞意味着当前线程将失去它对资源的占用）。

- void close()

  关闭这个输入流。

- void mark(int readlimit)

  在输入流的当前位置打一个标记（并非所有的流都支持这个特性）。如果从输入流中已经读入的字节多于readlimit个，则这个流允许忽略这个标记。

- void reset()

  返回到最后的标记，随后对read的调用将重新读入这些字节。如果当前没有任何标记，则这个流不被重置。

- boolean markSupported()

  如果这个流支持打标记，则返回true。

API java.io.OutputStream 1.0

- abstract void write(int n)

  写出一个字节的数据。

- void write(byte[] b)
- void write(byte[] b, int off, int len)

  写出所有字节或者某个范围的字节到数组b中。

  参数：b 数据写出的数组
  off 第一个写出字节在b中的偏移量
  len 写出字节的最大数量

- void close()

  清空并关闭输出流。

- void flush()

  清空输出流，也就是将所有缓冲的数据发送到目的地。

### 1.1.2 完整的流家族

与C语言只有单一类型FILE*即可工作良好不同，Java拥有一个包含各种流类型的流家族，其数量超过60个！请参见图1-1和图1-2。

让我们把流类家族中的成员按照它们的使用方法来进行划分，这样就形成了处理字节和字符的两个单独的层次结构。正如所见，InputStream和OutputStream类可以读写单个的字节或字节数组，这些类构成了图1-1所示的层次结构的基础。要想读写字符串和数字，就需要功能

更强大的子类，例如，DataInputStream和DataOutputStream可以以二进制格式读写所有的基本Java类型。最后，还包含了多个很有用的流，例如，ZipInputStream和ZipOutputStream可以以常见的ZIP压缩格式读写文件。

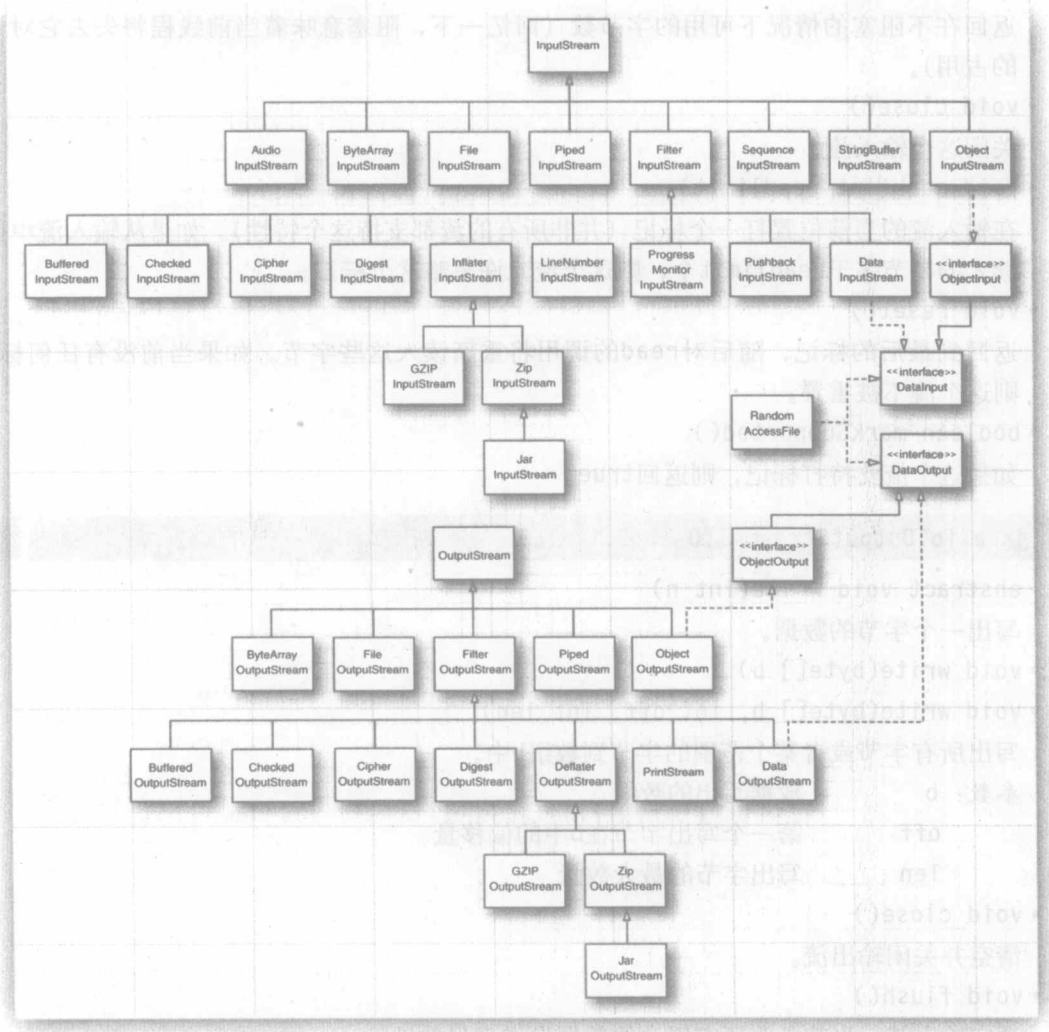



图1-1 输入流与输出流的层次结构

另一方面，对于Unicode文本，可以使用抽象类Reader和Writer的子类（请参见图1-2），Reader和Writer类的基本方法与InputStream和OutputStream中的方法类似。

```
abstract int read()
abstract void write(int c)
```

read方法将返回一个Unicode码元（作为一个在0～65535的整数），或者在碰到文件结尾时返回－1。write方法在调用时，要传递一个Unicode码元（请查看第Ⅰ卷第3章有关Unicode码

元的讨论）。

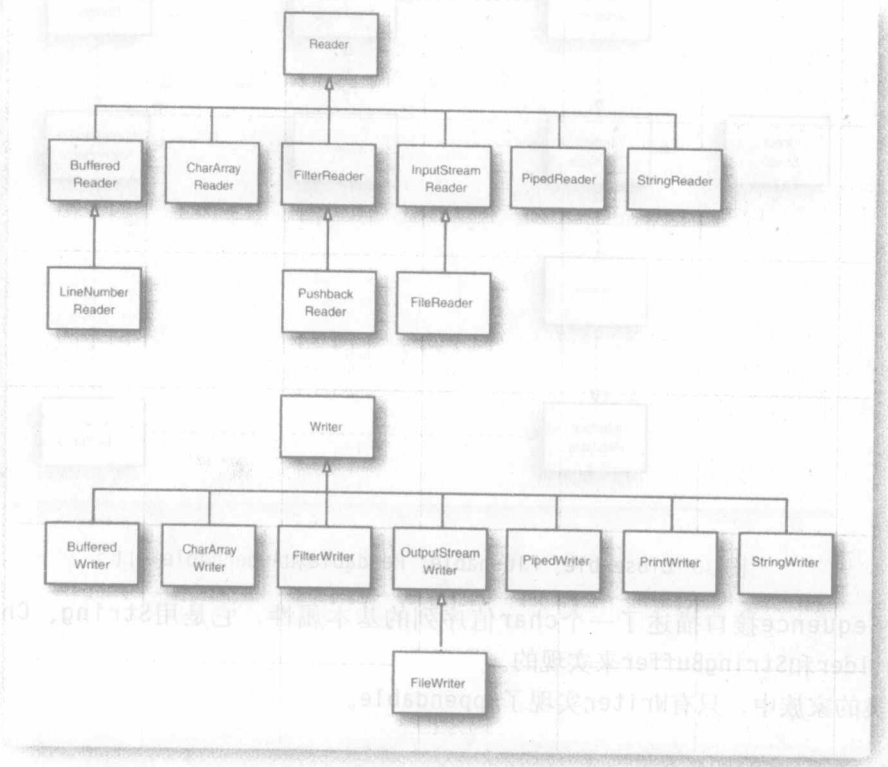



图1-2 Reader和Writer的层次结构

Java SE 5.0引入了4个附加的接口：Closeable、Flushable、Readable和Appendable（请查看图1-3）。前两个接口非常简单，它们分别拥有下面的方法：

```
void close() throws IOException
```

和

```
void flush()
```

InputStream、OutputStream、Reader和Writer都实现了Closeable接口，而OutputStream和Writer还实现了Flushable接口。

Readable接口只有一个方法：

```
int read(CharBuffer cb)
```

CharBuffer类拥有按顺序和随机地进行读写访问的方法，它表示一个内存中的缓冲区或者一个内存映像的文件（请查看第1.7.2节以了解细节）。

Appendable接口有两个用于添加单个字符和字符序列的方法：

```
Appendable append(char c)
Appendable append(CharSequence s)
```