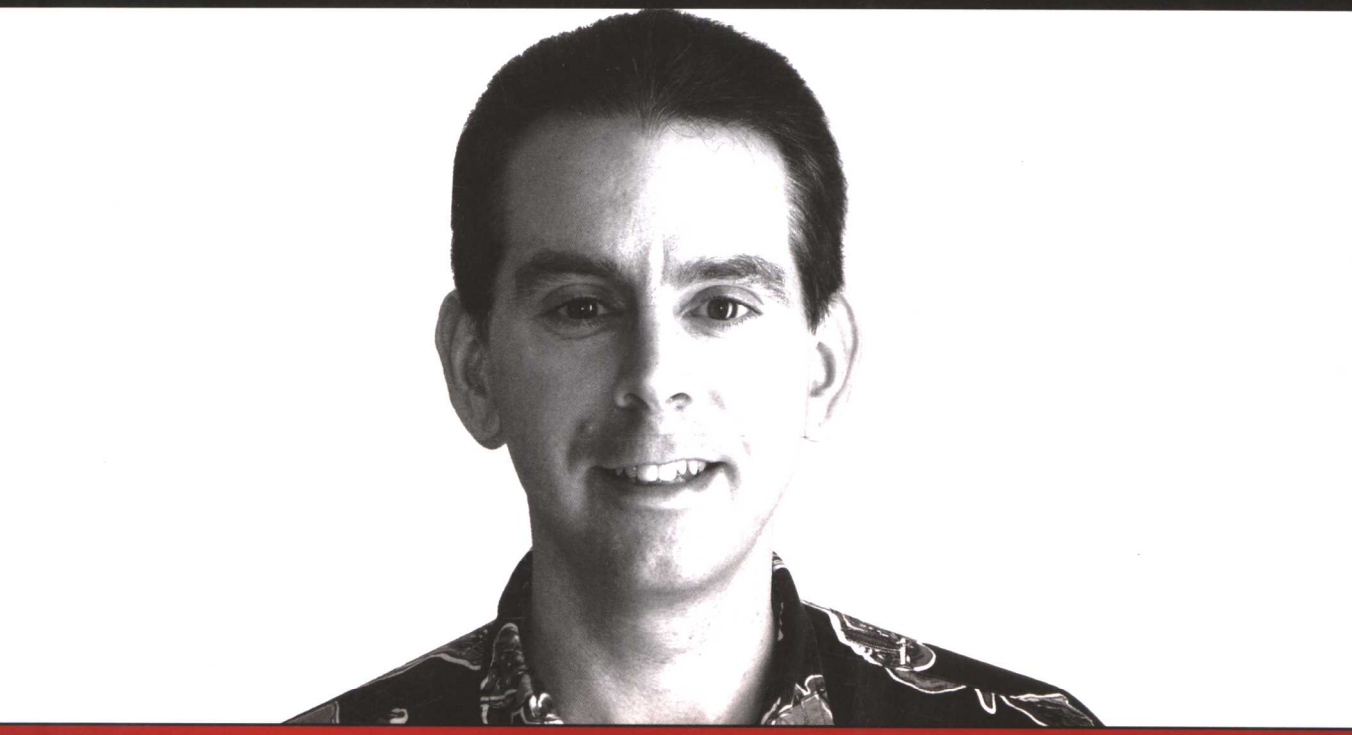
PROGRAMMER TO PROGRAMMER™



# Pro Java Programming
## Second Edition

# Java 高级编程
## (第2版)

(美) Brett Spell 著
董梁 刘艳 译

清华大学出版社

# Java 高级编程(第 2 版)

(美) Brett Spell 著

董梁 刘艳 译

清华大学出版社

北 京

EISBN：1-59059-474-6
Pro Java Programming, Second Edition
Brett Spell


北京市版权局著作权合同登记号　图字：01-2005-5286



**图书在版编目(CIP)数据**

Java 高级编程(第 2 版)/(美) 斯贝尔(Spell，B.) 著；董梁，刘艳 译. —北京：清华大学出版社，2006.11
书名原文：Pro Java Programming，Second Edition
ISBN 7-302-13909-1
Ⅰ.J…　Ⅱ. ①斯…②董…③刘…　Ⅲ. JAVA 语言—程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2006)第 116132 号

**出 版 者**：清华大学出版社　　**地　　址**：北京清华大学学研大厦
　　http://www.tup.com.cn　　**邮　　编**：100084
　　**社 总 机**：010-62770175　**客户服务**：010-62776969
**组稿编辑**：曹　康　　**文稿编辑**：于　平
**封面设计**：康　博　　**版式设计**：康　博
**印 刷 者**：北京鑫丰华彩印有限公司
**装 订 者**：三河市李旗庄少明装订厂
**发 行 者**：新华书店总店北京发行所
**开　　本**：185×260　**印张**：38.25　**插页**：1　**字数**：884 千字
**版　　次**：2006 年 11 月第 1 版　2006 年 11 月第1 次印刷
**书　　号**：ISBN 7-302-13909-1/TP·8360
**印　　数**：1～4000
**定　　价**：69.80 元

# 前　言

既然市面上已经有很多 Java 相关书籍，再写本书又有何意义呢？Java 这一概念非常宽泛，其中仍有不少领域没有得到详尽的介绍，而有些领域正是 Java 程序员们需要了解的。于是，本书应运而生了。本书主要介绍了 Java 编程中遇到的一些最常见的问题及其解决方案。

要成为优秀的 Java 编程人员，一定要打下坚实的专业基础。能够建立相当复杂的用户界面虽然很好，但如果代码臃肿，很耗内存和效率低下，则用户不会满意。本书不是介绍 Java 开发人员可以选择的大量开发选项，而是介绍 Java 开发人员经常遇到的常见任务，集中讲解了一些核心语言特性，如线程与内存管理，这些特性能够让专业的高质量 Java 应用程序脱颖而出。另外，大型 Java 程序的开发需要通晓 Java 的各个方面的知识，本书正是要起到这样的帮助作用。本书各章都有一个主题，分别讨论了专业 Java 开发人员需要掌握的一项技术。有些内容相当复杂，如 XML、线程和数据库编程，难以在一章内介绍详尽，不过本书仍为这些主题提供了相当丰富的信息。

本书的读者最好具有软件开发的背景知识和面向编程概念的基本认识。由于 Java 越来越普及，采用 Java 语言编程已成为大部分专业开发人员的选择。本书可以帮助您提高 Java 编程技能。

另外，本书介绍的部分功能是 Java 1.5(也称 Java 5)中新引入的，因此早期 Java 版本可能无法使用这些功能。不过本书将尽量标识出 Java 5 新版本中的特定功能，如果早期版本中无法使用这些功能，希望您不会因此感到惊讶或失望。

本书的大部分代码是在 Java 2 Platform，Standard Edition SDK 上测试通过的，但有些章节还需要其他工具：

- 关系型数据库系统和适当的 JDBC 驱动程序，我们用 Microsoft Access 2000 和 Java SDK 所带的 JDBC-ODBC 驱动程序
- Xalan XSLT 处理器
- JavaHelp 1.1

如果您有任何问题或意见以及勘误信息，请随时发送 E-mail 到 fwkbook@tup.tsinghua.edu.cn。

# 目　录

第 1 章

# Java 深入介绍

根据 Sun Microsystems 公司的定义，Java 是“一个简单、健全、面向对象、平台独立、多线程、动态并且通用的编程环境”。如此简单的定义却将 Java 拓展到许多全新的领域，某些领域最初甚至看不到与 Java 有丝毫联系。如今，有微处理器的地方，几乎就有 Java。无论是大型企业或是微型器件，无论是廉价手机器件还是巨型机，都用到了 Java。为了让 Java 支持如此多变的复杂环境，由此开发了大量不同版本的 API(Application Programming Interface，应用程序编程接口)，这些 API 都建立在一组共同的核心类上。

尽管如此，要成为一名优秀的 Java 程序员，仍要先打好基础。高度复杂的用户界面制作固然有用，但如果代码臃肿、既耗内存、效率又低，是无法让用户满意的。Java 开发人员可能会用到成百上千种开发方法，但本书不会介绍这些内容，而是阐述作为一名 Java 开发人员，您应该如何处理那些频繁出现的问题。本书将会着重介绍 Java 的核心语言特性，例如多线程和内存管理，这些特性能使一个 Java 应用程序焕然一新，更为专业。

Java 广为适用的核心，也就是其普及的原因，在于其平台独立性。Java 的 WORA(Write Once, Run Anywhere)特性源自其自身的运行方式，尤其是利用抽象执行环境来分离 Java 代码和底层操作系统。本章首先简要介绍 Java 的工作原理，包括 JVM(Java Virtual Machine，Java 虚拟机)，后续各章节将会探讨 Java 编程语言和各个 API 的细节。作为一名程序员，理解 Java 的内部机理有助于更好地理解语言本身，从而提高编程水平。

本章将涵盖如下内容：

- Java 平台的各种组件
- 通过 JVM 形成的 Java 平台独立性
- Java 程序的运行机制
- Java 类文件的实际内容
- 使用 JVM 的关键工具

那么，首先来剖析一下 Java 究竟是什么。

## 1.1 Java 体系结构

人们很容易把 Java 当作是开发各种应用程序的编程语言——用来编写源文件并将它们编译为字节码。不过，作为编程语言只是 Java 的众多用途之一，而真正形成 Java 众多优点(包括平台独立性)的是其底层架构。

完整的 Java 体系结构实际上是由 4 个组件组合而成：

- Java 编程语言
- Java 类文件格式
- Java API
- JVM

因此，使用 Java 开发时，就是用 Java 编程语言编写代码，然后将代码编译为 Java 类文件，接着在 JVM 中执行类文件。

JVM 与核心类共同构成了 Java 平台，也称为 JRE(Java Runtime Environment，Java 运行时环境)，该平台可以建立在任意操作系统上。图 1-1 显示了 Java 不同功能模块之间的相互关系，以及它们与应用程序、与操作系统之间的关系。

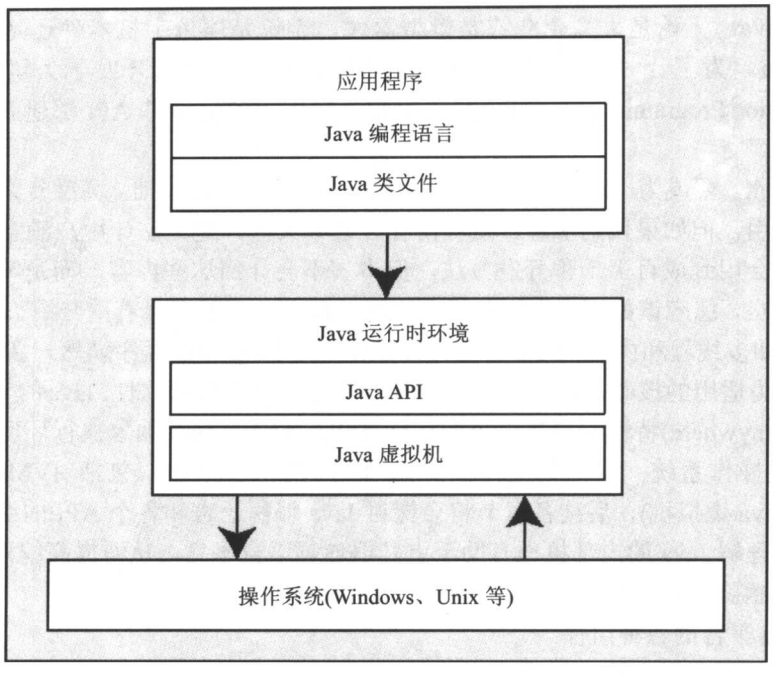



图 1-1　Java 各组件功能的概述

Java API 是预先编写的代码，并按相似主题分成多个包。Java API 主要分为 3 大平台：

- J2SE(Java 2 Platform, Standard Edition)：该平台中包含核心 Java 类和 GUI 类。
- J2EE(Java 2 Platform, Enterprise Edition)：该包中包含开发 Web 应用程序所需的类和接口，有 Servlet、JavaServer Page、以及 Enterprise JavaBean 类等。
- J2ME(Java 2 Platform, Micro Edition)：该包体现了 Java 的传统优势，为消费类产品提供了一个已优化的运行时环境，用于如传呼机、手机或汽车导航系统。

## 1.2 Java 虚拟机

介绍编写 Java 应用程序的各方面之前，这一节先花些篇幅来讨论运行这些程序的引擎。这个引擎正是 JVM，一种抽象计算系统用于解释编译过的 Java 程序。

其他编程语言，如 C 或 C++，中的编译器会根据特定处理器的、通常还根据特定的操作系统将源代码编译为可执行程序。该可执行程序无需外部支持就能在计算机上运行。

这种模式的缺点是不具有可移植性：在一个操作系统下编译的代码无法在另一个操作系统上运行，而必须在程序需要运行的各个操作系统上重新编译。另外，由于每个处理器供应商对编译器特性要求不同，因此在特定操作系统中为特定处理器系列(例如 Intel x86、SPARC 或 Alpha 系列)编译的代码，在相同操作系统但不同类型的处理器上可能也无法运行。例如，同样是运行 Windows NT 的 Alpha 工作站和 Intel PC，前一平台上所编译的代码就无法在后一平台上工作。

当人们开始为 Internet 编写应用程序时，该问题就变得尤为严重。这些应用程序需要兼容各种不同的操作系统、各种不同的平台以及各种不同的浏览器。解决问题的惟一办法是开发一种平台独立的语言。

20 世纪 90 年代初期，Sun Microsystems 的开发人员们提出了用于消费类电子产品的平台独立的语言，遗憾的是，平台独立的概念在当时因太过超前而搁置。不过随着 Internet 的出现，开发人员意识到这种语言的巨大潜力，Java 由此诞生。

Java 语言可移植性的关键在于 Java 编译器输出的并非标准可执行代码。Java 编译器会生成一组已优化的指令，称为字节码(bytecode)程序。字节码是按规定格式排列的一系列字节，稍后将会详细解释。字节码程序由运行时系统(即 JVM)进行解释。因此在一个平台上生成字节码程序后，其他安装了 JVM 的平台就能运行该程序。

虽然不同平台上的 JVM 各不相同，但上文做法是可行的。换句话说，在 Unix 工作站上编译的 Java 程序能够在 PC 或 Mac 上运行。只要按 Java 语言的标准形式编写源代码，然后编译为字节码程序，每个 JVM 都能将字节码解释为所在平台对应的本地调用(也就是解释为指定处理器能够识别的语言)。正是通过这种抽象形式，不同平台上的各种操作系统能够以相同的方式完成一些操作，如打印、文件访问以及硬件操作。

字节码的一个特点(也有人认为是一个缺点)，就是字节码在计算机上运行时并不是由处理器直接执行的。字节码程序是通过 JVM 运行的，由 JVM 解释字节码，因此 Java 是一种解释型语言(interpreted language)。Java 作为解释型语言而具有平台独立性，但相比标准的可执行类型代码，Java 的性能也稍差。不过，自 JSDK(Java Software Development Kit，Java 软件开发包)1.3 版发布后，Java 程序与其他语言程序之间的速度差距已经基本消除。

表 1-1 中列出了编译型语言与解释型语言。