



Java 经典教材译丛

Java

程序设计高级主题

THOMSON



TM

著者: [美] Joe Wigglesworth
Paula Lumby

译者: 赵强 陈洪

北京大学出版社
<http://cbs.pku.edu.cn>

Java 经典教材译丛

Java 程序设计高级主题

Java Programming: Advanced Topics

[美] Joe Wigglesworth Paula Lumby 著
赵 强 陈 洪 译

北 京 大 学 出 版 社

• 北 京 •

内 容 简 介

本书主要向读者讲述了 Java 语言的高级编程技巧。首先介绍了 Java 语言的基本语法,然后又指导已经对 Java 语言具有一定基础的读者进行高级程序设计。内容包括多线程、图形用户界面、窗口和事件编程、JavaBean、网络编程,以及服务器编程等,而且详细讲解如何编写 Java applet。

本书适合于对 Java 编程有一定基础并想学习更高技术和编程技巧的读者。适合作为大学计算机专业高年级的教材,也可作为程序员的参考用书。

Original English language title: Java Programming: Advanced Topics, by Joe Wigglesworth & Paula Lumby.

EISBN: 0-7600-1098-6

Copyright © 2000 by Course Technology, a division of Thomson Learning.

Original language published by Thomson Learning (a division of Thomson Learning Asia Pte Ltd). All rights reserved. 本书原版由汤姆森学习出版集团出版。版权所有,盗印必究。

Peking University Press is authorized by Thomson Learning to publish and distribute exclusively this simplified Chinese edition. This edition is authorized for sale in the People's Republic of China only (excluding Hong Kong, Macao SARs and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

本书中文简体字翻译版由汤姆森学习出版集团授权北京大学出版社独家出版发行。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾)销售。未经授权的本书出口将被视为违反版权法的行为。未经出版者预先书面许可,不得以任何方式复制或发行本书的任何部分。

981-254-145-4

北京市版权局著作权合同登记号 图字: 01-2003-8586 号

图书在版编目(CIP)数据

Java 程序设计高级主题/(美)威格尔斯沃思(Wigglesworth,J.)等著;赵强,陈洪译. —北京:北京大学出版社, 2004.3

(Java 经典教材译丛)

ISBN 7-301-06946-4

I. J... II. ①威... ②赵... ③陈... III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2004)第 004741 号

书 名: Java 程序设计高级主题

著作责任者: [美] Joe Wigglesworth, Paula Lumby 著

译 者: 赵 强 陈 洪

责任编辑: 温丹丹

标准书号: ISBN 7-301-06946-4/TP·0755

出 版 者: 北京大学出版社

地 址: 北京市海淀区中关村北京大学校内 100871

网 址: <http://cbs.pku.edu.cn> <http://www.macrowin.net>

电 话: 邮购部 62752015 发行部 62750672 编辑部 62750581

电子信箱: xxjs@pup.pku.edu.cn macrowin@macrowin.net

排 版 者: 北京东方人华北大彩印中心 电话: 62754190

印 刷 者: 河北涞县鑫华书刊印刷厂

发 行 者: 北京大学出版社

经 销 者: 新华书店

787 毫米×1092 毫米 16 开本 32 印张 991 千字

2004 年 3 月第 1 版 2004 年 3 月第 1 次印刷

定 价: 49.00 元

内 容 简 介

本书主要向读者讲述了 Java 语言的高级编程技巧。首先介绍了 Java 语言的基本语法,然后又指导已经对 Java 语言具有一定基础的读者进行高级程序设计。内容包括多线程、图形用户界面、窗口和事件编程、JavaBean、网络编程,以及服务器编程等,而且详细讲解如何编写 Java applet。

本书适合于对 Java 编程有一定基础并想学习更高技术和编程技巧的读者。适合作为大学计算机专业高年级的教材,也可作为程序员的参考用书。

Original English language title: Java Programming: Advanced Topics, by Joe Wigglesworth & Paula Lumby.

EISBN: 0-7600-1098-6

Copyright © 2000 by Course Technology, a division of Thomson Learning.

Original language published by Thomson Learning (a division of Thomson Learning Asia Pte Ltd). All rights reserved. 本书原版由汤姆森学习出版集团出版。版权所有,盗印必究。

Peking University Press is authorized by Thomson Learning to publish and distribute exclusively this simplified Chinese edition. This edition is authorized for sale in the People's Republic of China only (excluding Hong Kong, Macao SARs and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

本书中文简体字翻译版由汤姆森学习出版集团授权北京大学出版社独家出版发行。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾)销售。未经授权的本书出口将被视为违反版权法的行为。未经出版者预先书面许可,不得以任何方式复制或发行本书的任何部分。

981-254-145-4

北京市版权局著作权合同登记号 图字: 01-2003-8586 号

图书在版编目(CIP)数据

Java 程序设计高级主题/(美)威格尔斯沃思(Wigglesworth,J.)等著;赵强,陈洪译. —北京:北京大学出版社, 2004.3

(Java 经典教材译丛)

ISBN 7-301-06946-4

I. J... II. ①威... ②赵... ③陈... III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2004)第 004741 号

书 名: Java 程序设计高级主题

著作责任者: [美] Joe Wigglesworth, Paula Lumby 著

译 者: 赵 强 陈 洪

责任编辑: 温丹丹

标准书号: ISBN 7-301-06946-4/TP·0755

出 版 者: 北京大学出版社

地 址: 北京市海淀区中关村北京大学校内 100871

网 址: <http://cbs.pku.edu.cn> <http://www.macrowin.net>

电 话: 邮购部 62752015 发行部 62750672 编辑部 62750581

电子信箱: xxjs@pup.pku.edu.cn macrowin@macrowin.net

排 版 者: 北京东方人华北大彩印中心 电话: 62754190

印 刷 者: 河北涿县鑫华书刊印刷厂

发 行 者: 北京大学出版社

经 销 者: 新华书店

787 毫米×1092 毫米 16 开本 32 印张 991 千字

2004 年 3 月第 1 版 2004 年 3 月第 1 次印刷

定 价: 49.00 元

编 者 序

欢迎学习 Java 程序设计语言!

自计算机发明以来,无处不在显示着它对整个人类文明和社会进步产生着如此巨大、如此深刻的影响,推动着人类社会一日千里地向前发展

历经半个多世纪,计算机软件——计算机的重要组成部分,也发生了日新月异的变化。从机器语言到汇编语言再到高级语言,从结构化语言到面向对象语言,人们不断地探索和发明新的、功能强大又简单易学的计算机语言。

用计算机语言编程,有如我们用世界上的任何一种语言写作一样,两者都是创造性的工作,而且都要用到相应的技术。如果您掌握了一种语言,熟知它的词汇和语法规则,也许您能熟练地运用这种语言进行表达,但这并不能使您成为一位作家。写作是一种创造性的工作,它需要有创造性的人来做。用计算机语言编程也是同样的道理。也许您掌握了当今最流行的程序设计语言,而且也能迅速地写出程序语句,但是,程序的内容只能靠您的创造性来产生。

1. 为什么要学习 Java 程序设计语言?

工欲善其事,必先利其器。没有得心应手的工具又怎么能随心所欲地发挥创意?计算机高级语言众多,而 Java 语言的得天独厚的优点,使它从众多语言中脱颖而出,越来越受到人们的青睐。Java 是一种跨平台、适合于分布式计算环境的面向对象编程语言。具体来说,它具有如下特性:简单、面向对象、分布式、易解释、可靠、安全、与平台无关、体系结构中立、可移植、高性能、多线程、动态等。

鉴于此,我们从国外遴选了 8 本流行的 Java 书,组成了《Java 经典教材译丛》,以期读者能从中受益。

2. 您渴望从《Java 经典教材译丛》学到什么?

博大精深是我们选此套丛书的宗旨,从 Java 基本概念和技术到其高级主题,丛书的内容涵盖了 Java 的全部知识。原书多由国外教学经验丰富的教师编写,书中提供大量的实例和案例。为了给读者提供良好的服务,我们将大部分书中的实例中的源代码放在 <http://cbs.pku.edu.cn> 中的【下载专区】上,读者可从中下载。

3. 什么人使用《Java 经典教材译丛》?

无论您是一位计算机语言初学者,还是一位熟知计算机语言的编程人员,《Java 经典教材译丛》中必有一本适合您。当然,本套丛书主要针对在校大专院校的学生而编写,所以大多数读者都可以轻松上手,教师更可以从中找到教学用书和参考读物,部分书中带有星号(*)的章节则是为更高层次的读者而设计,为选学内容。

4. 对计算机的硬件和软件有什么需求?

- 软件 可以通过访问 Sun 的网站(<http://java.sun.com>)下载 JDK 编译程序,下载的软件大约为 20MB。
- 硬件 必须具备可以运行 Windows 操作系统的计算机或者 Windows NT 工作站,100MB 的空闲磁盘空间,最少 32MB 的内存(如果使用 Windows NT,建议准备 64MB 的内存)。

丛书编委会

2003 年 11 月

本书使用说明

本书用来指导初级程序设计员使用 Java 程序设计语言开发应用程序和 applet(小程序)。本书假定读者有一定的程序设计语言的经验。其中书中的方框符号(■)是 Java 认证考试要考的内容,箭头符号(➤)是程序员无论如何要掌握的内容,也是各章的重点内容。

学生文件

要完成对本书的各章和各章习题的学习,需要从 <http://cbs.pku.edu.cn> 的【下载专区】中下载学生文件,包括程序练习文件夹和 appendix 以及 internationalization。

使用自备的计算机

可以使用自己的计算机完成本书中各章和习题的学习。要使用自备的计算机,需要满足下列条件:

- 软件 可以通过访问 Sun 的网站(<http://java.sun.com>)下载 JDK 编译程序,下载的软件大约为 20MB。
- 硬件 必须具备可以运行 Windows 操作系统的计算机或者 Windows NT 工作站,100MB 的空闲磁盘空间,最少 32MB 的内存(如果使用 Windows NT,建议准备 64MB 的内存)。

启动 JDK 编译程序

执行本书中的各个步骤需要具备以下条件:您可以编译保存在各个路径上的*.java 文件,当前的路径是包括当前指南的文件夹的驱动器,例如, A:\Tutorial.06>。要用这样的方式配置 Java 运行,请执行下列步骤:

- (1) 转到命令提示符。
- (2) 键入 `path = drive:\pathname\bin`, 其中 `drive` 是包括 JDK 安装程序的驱动器号, `pathname` 是 JDK 文件夹的完整路径名。例如, 如果安装程序在驱动器 C 上的 JDK1.2 文件夹中, 则命令为 `path = C:\JDK1.2\bin`。
- (3) 按 Enter 键。现在可以编译任何文件夹中的 Java 文件。然而, 每当启动 Java 时, 必须键入路径命令。如果您愿意, 可以将路径命令行添加到计算机的 `autoexec.bat` 文件, 以便 Java 能够自动地启动。

目 录

第 1 章 Java 程序设计语言简介 1

1.1 简介	1
1.2 Java 平台概述	1
1.3 能使用 Java 平台的地方	2
1.4 什么是 Java	3
1.5 Java 技术的发展历史	5
1.6 Java 工具	6
1.7 如何在 Java 2 SDK 工具中查找文件	8
1.8 第一个 Java 程序	8
1.9 运行 Java 程序	12
1.10 Java 程序的注释	14
1.10.1 文档注释	14
1.10.2 使用 javadoc 标记	15
1.10.3 用 javadoc 产生 HTML	16
1.11 程序打包用于分发	20
1.12 小结	21
1.13 习题	22
1.14 训练题	23

第 2 章 程序基本概念 25

2.1 简介	25
2.2 变量和标识符	25
2.3 Java 编程语言的关键字	27
2.4 表达式	28
2.4.1 操作符	28
2.4.2 数学类中的浮点运算	30
2.5 常量	31
2.6 语句	33
2.6.1 if 语句	34
2.6.2 switch 语句	34
2.6.3 while 语句	35
2.6.4 do 语句	35
2.6.5 for 语句	36
2.6.6 跳离循环	36
2.7 方法	39
2.7.1 参数	40
2.7.2 返回值和返回语句	40
2.7.3 局部变量	41
2.7.4 main 方法	41

2.8 I/O 控制台	41
2.8.1 简单的输入输出流	42
2.8.2 一个输入字符串的语法分析	43
2.8.3 一个 I/O 控制台程序示例	44
2.9 小结	45
2.10 习题	46
2.11 训练题	48

第 3 章 类型 51

3.1 简介	51
3.2 强类型	51
3.3 基本类型和引用类型	51
3.3.1 布尔类型	52
3.3.2 字符类型	52
3.3.3 转义字符	53
3.3.4 整数类型	53
3.3.5 浮点类型	54
3.3.6 强制转换基本类型	54
3.4 对象引用	55
3.4.1 创建一个引用类型的对象	55
3.4.2 对象引用的赋值	55
3.4.3 传递作为参数的对象引用	56
3.4.4 删除引用类型的对象	56
3.4.5 自动地回收垃圾	56
3.5 Object 类	57
3.6 基本类型的包装类	58
3.7 数组	59
3.7.1 数组索引和长度	61
3.7.2 一个方法返回一个数组	62
3.7.3 复制数组	63
3.7.4 使用多维数组	63
3.8 字符串	65
3.8.1 字符串操作	65
3.8.2 字符串缓冲器对象的操作	68
3.9 小结	69
3.10 习题	70
3.11 训练题	73

第 4 章 类和包 76

4.1 简介	76
--------------	----

4.2 类和面向对象程序设计.....	76
4.3 Java 的类.....	77
4.4 类的定义.....	78
4.4.1 定义成员.....	79
4.4.2 定义方法.....	79
4.4.3 定义域.....	80
4.4.4 初始化域.....	80
4.5 使用一个已定义的类.....	81
4.6 使用构造方法和终结器.....	83
4.6.1 构造方法.....	83
4.6.2 终结器.....	84
4.7 引用对象和垃圾收集.....	86
4.8 重载方法.....	86
4.9 使用作为对象引用的 this.....	89
4.10 使用静态成员.....	90
4.10.1 初始化静态成员.....	92
4.10.2 类方法和继承.....	93
4.11 使用包.....	93
4.11.1 包名、文件名和文件夹名.....	94
4.11.2 子包.....	95
4.11.3 引入语句.....	95
4.11.4 核心 Java 包.....	96
4.12 小结.....	98
4.13 习题.....	99
4.14 训练题.....	102
第 5 章 继承.....	107
5.1 继承与面向对象程序设计.....	107
5.2 Java 程序设计语言对继承的支持.....	108
5.3 继承类.....	109
5.3.1 final 类.....	110
5.3.2 抽象类.....	111
5.3.3 多重继承的问题.....	112
5.4 实现接口.....	112
5.5 访问控制符对成员的作用.....	115
5.6 对象引用 super.....	117
5.7 构造方法和继承.....	118
5.8 终结器方法与继承.....	120
5.9 重置方法.....	122
5.10 动态绑定.....	124
5.11 小结.....	127
5.12 习题.....	128
5.13 训练题.....	133

第 6 章 异常.....	136
6.1 简介.....	136
6.2 使用 try 程序块包装异常.....	136
6.2.1 异常处理与调用堆栈.....	137
6.2.2 声明未被捕获的异常.....	137
6.3 抛出异常.....	138
6.3.1 受检验的异常和不受检验的异常.....	139
6.3.2 在终结器中抛出异常.....	140
6.4 捕获异常.....	140
6.5 使用 finally 程序块.....	145
6.6 小结.....	147
6.7 习题.....	148
6.8 训练题.....	151
第 7 章 克隆及运行类型信息.....	154
7.1 简介.....	154
7.2 克隆对象.....	154
7.2.1 使对象可克隆.....	156
7.2.2 覆盖默认的 clone 方法.....	157
7.2.3 定义 Cloneable 类.....	159
7.3 运行时类型信息.....	160
7.3.1 决定对象类型.....	160
7.3.2 访问运行时的类信息.....	161
7.4 类型转换.....	163
7.5 使用反射 API.....	165
7.6 小结.....	172
7.7 习题.....	173
7.8 训练题.....	175
第 8 章 输入和输出.....	178
8.1 Java 平台如何支持 I/O.....	178
8.2 I/O 程序设计.....	178
8.3 字节流.....	179
8.3.1 预定义的流对象.....	180
8.3.2 输入输出方法.....	181
8.3.3 过滤流.....	182
8.3.4 其他的字节 I/O 类.....	182
8.4 文件 I/O 基础.....	183
8.4.1 浏览文件.....	185
8.4.2 随机访问文件 I/O.....	186
8.5 字符流.....	187
8.5.1 在字节流和字符流 I/O 上建立连接.....	188
8.5.2 使用其他字符 I/O 类.....	188

8.5.3 每次读取一个标记.....	188	10.6 习题.....	245
8.6 对象序列化.....	189	10.7 训练题.....	247
8.6.1 序列化对象.....	189	第 11 章 多线程	250
8.6.2 使用对象流.....	189	11.1 线程和多线程.....	250
8.6.3 抑制域的序列化.....	189	11.2 线程的生命周期.....	250
8.6.4 指定版本号.....	192	11.3 创建并运行线程.....	251
8.6.5 序列化格式的兼容性.....	192	11.4 停止一个线程.....	254
8.7 小结.....	192	11.4.1 stop 方法和其他的方法被取消的原因.....	254
8.8 习题.....	193	11.4.2 如何停止进程.....	254
8.9 训练题.....	195	11.5 创建服务线程.....	256
第 9 章 嵌套类和内部类	198	11.6 编写稳固的多线程程序.....	257
9.1 简介.....	198	11.7 同步线程.....	257
9.2 嵌套类和接口.....	198	11.7.1 何时同步代码.....	259
9.3 内部类.....	201	11.7.2 内部类的同步方法.....	259
9.3.1 为什么要使用内部类.....	202	11.8 线程间通信.....	260
9.3.2 定义成员内部类.....	202	11.8.1 让线程保持等待.....	262
9.3.3 避免内部类中名字冲突.....	205	11.8.2 唤醒一个线程.....	262
9.3.4 内部类的封闭对象.....	205	11.8.3 唤醒所有线程.....	262
9.3.5 研究成员内部类的子类.....	207	11.8.4 线程间通信的例子.....	262
9.4 局部内部类.....	208	11.9 线程组.....	266
9.5 匿名局部内部类.....	210	11.10 小结.....	269
9.5.1 创建匿名内部类.....	210	11.11 习题.....	269
9.5.2 使用实例初始器.....	210	11.12 训练题.....	272
9.6 对于产生出类文件的影响.....	211	第 12 章 图形用户界面	275
9.7 小结.....	212	12.1 简介.....	275
9.8 习题.....	212	12.2 图形用户界面的发展.....	275
9.9 训练题.....	216	12.3 图形用户界面的主要术语和特征.....	276
第 10 章 工具和集合类	219	12.4 Java 基础类概述.....	277
10.1 简介.....	219	12.5 事件模型.....	279
10.2 java.util 包.....	219	12.5.1 Swing 应用程序接口.....	279
10.3 Observer 和 Observable 对象.....	220	12.5.2 分离的模型架构.....	281
10.4 集合框架.....	223	12.6 JFC 例程.....	282
10.4.1 集合框架中的接口.....	223	12.6.1 Java 应用程序: Hello World!.....	283
10.4.2 用叠代器检索集合.....	230	12.6.2 3 个面板例程.....	285
10.4.3 常用的工具.....	231	12.7 小结.....	290
10.4.4 作为集合的数组.....	237	12.8 习题.....	291
10.4.5 算法.....	237	12.9 训练题.....	292
10.4.6 包装工具.....	239	第 13 章 窗口和事件编程	295
10.4.7 继承抽象工具.....	239	13.1 用 JFC 编程.....	295
10.4.8 遗留集合框架类.....	239	13.2 Swing API 组件.....	296
10.4.9 用枚举方式检索集合.....	240	13.3 JComponent 类.....	297
10.5 小结.....	245		

13.4 窗口、对话框和面板.....	298	14.11 小结.....	382
13.5 布局管理器.....	303	14.12 习题.....	383
13.5.1 边缘布局管理器.....	306	14.13 训练题.....	384
13.5.2 流式布局管理器.....	307	第 15 章 JavaBean	388
13.5.3 网格布局管理器.....	308	15.1 简介.....	388
13.5.4 网格块布局管理器.....	308	15.2 JavaBean 组件模型.....	388
13.5.5 卡片式布局管理器.....	309	15.3 JavaBean 的组成.....	389
13.5.6 箱式布局管理器.....	309	15.4 JavaBean 开发环境.....	390
13.5.7 标签窗格.....	310	15.5 使用 Sun BeanBox.....	391
13.5.8 拆分窗格.....	310	15.6 建立 JavaBean 类.....	393
13.5.9 手工设置组件的位置.....	310	15.6.1 ImageBean1 解析.....	395
13.6 标签、按钮、复选框.....	311	15.6.2 在 BeanBox 中使用 ImageBean1.....	397
13.7 事件.....	316	15.7 其他类型的 JavaBean 属性.....	399
13.8 菜单、工具栏、动作.....	320	15.7.1 索引属性.....	399
13.9 滑杆、进度条、滚动条.....	327	15.7.2 绑定属性.....	399
13.10 列表框、组合框.....	330	15.7.3 限制属性.....	400
13.11 文本输入组件.....	335	15.8 增加自定义的事件类型.....	400
13.12 选择器.....	340	15.9 建立带有自定义事件的 JavaBean.....	401
13.13 表格和树.....	345	15.9.1 ImageBean2 解析.....	404
13.14 绘制.....	348	15.9.2 ImageBean2 中的自定义事件.....	405
13.15 二维图形.....	349	15.9.3 自定义事件类的监听器接口.....	406
13.16 小结.....	354	15.9.4 在 BeanBox 中使用 ImageBean2.....	406
13.17 习题.....	354	15.10 提供 JavaBean 的信息.....	407
13.18 训练题.....	356	15.11 提供一个定制的属性编辑器.....	409
第 14 章 客户端的 Java 和 applet	360	15.12 建立带有 BeanInfo 类的 JavaBean.....	409
14.1 applet 概述.....	360	15.12.1 ImageBean3 解析.....	413
14.2 什么是 applet.....	360	15.12.2 ImageBean3 类的属性编辑器.....	414
14.2.1 HTML 文档概览.....	361	15.12.3 ImageBean3 类的 BeanInfo 类.....	414
14.2.2 HTML 支持 applet.....	361	15.12.4 ImageBean3BeanInfo 解析.....	415
14.2.3 创建 applet.....	362	15.13 建立使用 BeanContext API 的 JavaBean.....	417
14.3 applet 例程.....	362	15.13.1 ImageBeanChildContextSupport	
14.4 使用 HTML applet 标记.....	363	类的定义.....	418
14.5 使用 java.applet 包.....	365	15.13.2 ImageBeanContextChildSupport	
14.6 继承 applet 类.....	366	类解析.....	419
14.7 TickerMessage applet.....	369	15.14 Enterprise JavaBean.....	421
14.8 在 applet 中使用多媒体.....	375	15.15 小结.....	421
14.8.1 添加声音.....	375	15.16 习题.....	422
14.8.2 添加图像.....	376	15.17 训练题.....	423
14.9 安全.....	378	第 16 章 网络编程	428
14.9.1 理解 applet 的安全.....	378	16.1 简介.....	428
14.9.2 标记 applet.....	379	16.2 使用 URL.....	428
14.9.3 权限与策略文件.....	380	16.3 使用套接字.....	429
14.10 将 applet 打包成 jar 文件.....	381		

16.3.1 TCP/IP 套接字	429	17.5.3 在 session 中保存状态数据	470
16.3.2 数据报套接字	435	17.5.4 使用 cookie 提供连续性	472
16.4 RMI	438	17.6 servlet 驱动的 Web 编程	474
16.4.1 开发对象的远程类	439	17.6.1 使用多线程编写 servlet	475
16.4.2 开发 RMI 服务器端类	440	17.6.2 明智地使用 applet	475
16.4.3 开发 RMI 客户端类	441	17.6.3 使用 MVC 的设计模式	476
16.4.4 运行服务器端程序和客户端程序	443	17.6.4 小心浏览器的 Back 按钮	476
16.4.5 JNDI	444	17.6.5 安全	476
16.4.6 Java IDL 与 CORBA	444	17.7 JSP	477
16.5 小结	444	17.7.1 服务器如何处理 JSP	479
16.6 习题	445	17.7.2 JSP 中的 Java 编程	479
16.7 训练题	446	17.7.3 在应用 MVC 的设计模式中 使用 JSP 和 JavaBean	483
第 17 章 服务器端编程	450	17.7.4 JSP 规范的改进	487
17.1 简介	450	17.8 数据库连接	488
17.1.1 建立动态 Web 站点的一些考虑	450	17.8.1 JDBC API	489
17.1.2 理解 HTTP 协议	451	17.8.2 加载数据库驱动程序	490
17.2 servlet 简介	452	17.8.3 建立数据库连接	490
17.2.1 编写第一个 servlet	454	17.8.4 执行动态 SQL 语句	493
17.2.2 安装、运行 servlet 例程	456	17.8.5 处理结果集	496
17.3 开发 Web 和 servlet	456	17.9 小结	497
17.4 servlet API	457	17.10 习题	498
17.5 编写 servlet 接收请求和发送响应	459	17.11 训练题	500
17.5.1 编写处理 HTML 表单的 servlet	464		
17.5.2 servlet 的输出重定向	469		

第 1 章 Java 程序设计语言简介

本章将学到以下内容:

- 了解 Java 语言的特性和能力。
- 了解 Java 语言的发展过程。
- 学习使用 Java 2 SDK 的基本开发工具。
- 编辑、编译和运行你的第一个 Java 程序。
- 为源程序代码加入注释, 也包括文档注释。
- 使用 Javadoc 工具和文档注释来系统管理制作的类。
- 使用 jar 工具打包所有的 Java 应用程序、applet、组件或 servlet 用于发布使用。

1.1 简 介

欢迎走入 Java 编程世界。本书将介绍 Java 语言的特性和历史、制作 Java 程序和组件的方法, 及其广泛的用途。无论你是一名在校的学生, 还是一位资深的软件开发专家, 本书都将是最好的辅导资料和首选用书。

如果你是一位资深的程序员, 非常渴望实践更高的 Java 实例, 那么会发现本章的大部分内容是以一套完整的、小巧的 Java 实例, 讲述 Java 语言的开发程序和方法。其内容包括: 实例开发背景和要求、代码生成、编译和运行。

本章的其他部分介绍 Java 平台涉及的各要素和 Java 技术发展的历史。同时, 介绍互联网、面向对象的技术、与 Java 技术的关系, 以及为什么 Java 技术正在变革计算机工业等。

1.2 Java 平台概述

Java 编程语言具有独特威力, 因为它不仅是一个编程语言, 还是一个完善的开发平台。许多编程语言不支持多线程、图形用户界面或网络环境。对于这些语言编制的程序, 用户需要使用特定的应用程序接口, 将其绑定在特定的硬件或操作系统上。Java 开发平台是第一个完全将编程语言与应用程序接口(API)整合在一起的技术, 如整合线程接口、套接字(socket)接口、图形用户界面组件(GUI Component)接口等。因此, 通过 Java 开发平台生成的程序, 可以运行在众多的硬件平台和操作系统上。

用户安装完 Java 开发工具, 需要将其加载到本地系统中。Java 开发工具包括: Java 虚拟机(JVM)和程序包。这些程序包为 Java 语言提供编译器、程序解释程序和实时运行环境。这些程序包类似其他开发语言为预定义函数或类而设立的类库。每个程序包都装载了许多可重用的类, 程序员可在程序中对其进行访问。构成应用程序接口的类叫核心类。所有的核心类都被归入到程序包中, 以供程序员调用。

程序员可以选择多种 Java 应用程序开发工具, 例如, Java 2 SDK。它可以运行在 Windows 操作系统和 Sun 的 Solaris 操作系统中。

一些软件供应商提供集成化的 Java 语言开发环境。每个集成开发环境提供一个图形用户界面, 使用户进行代码编辑、编译、调试和运行。有些集成化的开发工具还提供一些附加的功能, 例如, 代码生成器、源代码版本控制器等。

1.3 能使用 Java 平台的地方

近些年, Java 平台对计算机工业产生了巨大的影响。越来越多的企业开始应用 Java 平台。Java 语言被用来编写极其重要的应用程序, 其优秀的特性为企业的工作带来了深远影响。它正在帮助企业建立基于 Web 的业务操作并跻身于电子商务的行列。Java 平台可以加入到微芯片制造商提供的芯片中, 程序员可以在任何环境中开发程序, Java 的这一特性已吸引了更多的人士关注 Java 技术。

Java 具有以下使用方式:

- Java 生成的独立运行程序, 可运行在不同规模和不同复杂度的计算机上。小到嵌入芯片的一小段程序, 大到运行在大型超级市场的订单录入系统。
- 生成基于 Web 的 applet。在网络中, 它们可随 HTML 文档被下载并被 Web 浏览器运行。
- 编写 Java 类, 作为 Web 服务器的扩充。在 Web 服务器上的页面被发送到浏览器之前, 为其准备动态内容。
- 在网络中发布程序。Java 类可以被用来在不同机器间传递数据, 或调用彼此机器上类的方法。
- 应用基于 Java 的技术, 满足企业广域范围内的数据交换处理。Sun、IBM 等公司, 在研究 Java 的最前线技术时, 制定出了许多技术标准规范, 其中主要规范为: 定义 Java 的类与事务处理器、其他开发语言生成的类或其他不同开发环境的窗体之间的工作方式。至今为止, 这些标准仍在继续发展之中。

不同的系统环境需要不同的 Java 程序。本书主要介绍 3 种 Java 程序的编制方法。

- 应用程序能够独立运行在任何规模的计算机上, 它可以是大型应用软件, 也可以是简单的小型实用程序。它们既可通过命令行运行, 也可通过图形用户界面运行。一个应用程序是否包含连接数据库、网络或其他应用程序的接口, 取决于该应用程序所使用的应用程序接口(API)集。可以使用本书所提供的所有 API, 除了第 17 章涉及的 servlet 应用程序接口。

Java 应用程序只运行在 Java 虚拟机(JVM)上。它们不是为某个特定的处理器指令集或操作系统编译的, 而是为 Java 平台编译的。最近, IBM 等在 Java 技术上处于领先地位的公司开发出一个新编译器。它可以将 Java 源代码处理成本地可执行模块, 并被动态加载到其特定操作系统中或制作成共享库。

- applet 是一种可通过 HTML 文档运行的程序。HTML 表示超文本标记语言, 是一种全局性的信息机构, 是网页使用的标记语言, 它将文档中的不同部分通过关键字建立连接, 使信息以交互方式进行搜索。HTML 具有专门为 applet 定义的标识符。当浏览器显示 HTML 文档时, 基于 Java 的 Web 浏览器要有一个 JVM 来运行 applet。applet 常被用来增加网页的视觉或多媒体效果。在本书的第 14 章中, 将着重介绍 applet 的具体特点和使用方法。
- servlet 就像 applet, 是用来优化网页设计的 Java 程序。与 applet 不同的是, servlet 运行在基于 Java 的 Web 服务器上, 用来生成被发送到客户端浏览器上进行显示的 HTML 文档。例如, 可以使用 servlet 对 Web 客户端提交的 HTML 窗体进行加工, 制作成一个新的回应网页。

当客户端请求阅览的网页都一样时, 就能使用 applet; 如果该网页在显示时需要调用一些客户端本地操作(如动画网页), 那么不能使用 applet。当 HTML 文档向客户端的特定请求提供的是基于服务器数据库的数据视图时, 就能使用 servlet。servlet 通常比较小巧, 但是常用来调用比较大的程序, 如数据库管理器, 从而为网页中需要的信息提取数据。Java 服务器网页技术(Java Server Pages, JSP)是一种简化的 servlet。JSP 中包含 HTML 代码, 它也包含嵌入其中的、在服务器上编译和运行的 Java 代码。在第 17 章中, 将详细介绍 servlet 和 JSP。

Java 平台支持软件组件。组件是大型程序使用的可重用的类。大型程序通常将其当做软件构造块来调用。尽管这不是一个新概念, 其他的技术也对其有论述, 但以下两个技术标准的推出, 证明 Java 技术

具有名副其实的真正的代码可重用性:

- **JavaBean** 是 Java 类, 或者是 Java 类组, 但是要遵循一套非常严格的技术标准。可以通过构建 JavaBean 组件, 将 Java 类变成可重用的和可配置的软件组件。通过使用开发工具, 程序员可以将 JavaBean 组件连接起来并应用到 Java 应用程序中, 这就像用标准微处理芯片焊接一个集成电路板一样。许多 Java 核心类都是 JavaBean 组件, 如用于 GUI 的可视化图形对象。在第 15 章中, 将详细介绍 JavaBean 标准和制作使用 JavaBean 组件的方法。
- **企业型 Java 小系统(Enterprise Java Bean, EJB)**是一种非常有前途的新兴技术。EJB 标准定义了分布式企业环境下, 服务器端组件的软件体系结构。它的目标是让程序员专注于 Java 类需要执行的业务逻辑。企业型 Java 服务器(Enterprise Java Server, EJS)用于管理以下服务: 事务处理、安全、命名服务来定位和鉴别 EJB。通常, 编程员需要一个 EJB 开发工具来构造和发布 EJB。因为 EJB 是更高层次的专题, 在本书中只对其进行一些简单介绍。

1.4 什么是 Java

根据 Java 编程语言, Sun 公司推出了一个语言和开发模式, 即平台独立性和基于互联网的安全性。在 Java 白皮书上, Sun 公司对 Java 进行了非常完善和综合的定义:

Java 是一个简单的、面向对象的、基于网络的、解释型的、健壮的、安全的、结构中立的、可移植的、高性能的、多线程的动态语言。

Sun 公司是 Java 开发语言的权威, 在其对 Java 的定义中, 详细列举了 Java 的特性。下面将针对这些特性一一进行论述。

Java 的简单性 Java 语言设计的初衷, 就是要将 Java 语言及相应 API 设计的比 C 和 C++更易于使用和学习。Java 是一个充分可靠的通用编程语言, 是为有经验的编程人员而设计的。同时, 编程人员将会发现, 将其以前的编程技巧应用在 Java 上, 是一个非常简单和快速的过程。

Java 语言的语法是基于 C 语言的。许多面向对象的语言的定义与 C++一模一样, 实施构造也相同。基于这一点, Java 的设计者没有引进 C 和 C++中易于出错或难于使用的特性。具体做法表现为: 舍弃了指针运算、淘汰了一些最常提出的过度复杂的特性、易造成存储单元处理错误的特性, 以及其他在 C 和 C++程序中常引发错误的特性。JVM 还用来管理内存存储单元的自动分配和收回, 因此极大地减轻了程序员的负担。C++程序员通常都同意这样一个观点: Java 是一个更简易的编程语言。因为许多 C++的概念, 如模板和操作符重载, 在 Java 中已经找不到对应概念。

另一方面, Java 语言仍在发展完善中, 新的特性增加其复杂性。最早版本的 API 或核心类集, 比现在的版本要小得多。例如, API 在最早的版本中只有 200 多个, 在 1.1 版本中大约有 500 个, 现在的 Java 2 有近 1 600 个。

Java 的基于网络性 Java 的成功, 部分来自于它的网络方面特性和可在企业环境中发布对象的特性。Java 平台包括一个程序包 java.net, 它提供基于网络的编程功能。也就是说, 在包 java.net 中使用核心类使网络化的计算处理起来更简易。

Java 平台提供 API 和其他工具来解决网络方面的主要问题, 例如, 访问控制、用户身份识别、数据传递过程中的安全性和完整性。它所提供的其他工具还允许用户使用 Java 对象, 甚至使用同一网络中其他计算机上的 Java 对象。详细描述请参阅第 17 章。

为网络和分布式环境编程时, 除了 Java 平台, 程序员有时需要使用特定的工具和 API。对于工业方面的标准规范, 要求 Java 开发者和工业组织一起参与制定。例如, 对象管理组织(object management group, OMG), 通过通用对象请求中介体系结构(common object request broker architecture, CORBA)定义了接口定义语言(interface definition language, IDL), 用来协调不同面向对象语言之间的互相调用。Java 平台不包括帮用户把 Java 程序连接到 IDL 的工具, 当然, 也不可能期望 Java 提供万能的解决方案。

Java 的解释性 Java 程序经过编译形成字节码，然后在虚拟机上解释执行，这是 Java 程序能够独立于平台运行的基础。字节码是用 Java 语言写的具有结构中立性的代码形式。Java 程序运行时，解释程序处理的是字节码而不是源代码。因此，编译过程是至关重要的。在这个过程中，编译器识别和删除错误，优化字节码。

一些计算机制造商还可提供一些更优越的编译器，它可以使用硬件平台的本地代码，将 Java 源代码编译到计算机的装入模块中，从而可以运行在 JVM 以外的其他的硬件平台上。例如，IBM 的高性能的 Java 编译器，可以将 Java 源代码编译成可在 Windows、OS/2、System 390、AS/400 和 AIX 上运行的执行程序。目标代码的运行速度比用字节码编译的程序还快，只是体积相对较大。

Java 的健壮性 Java 平台的设计初衷就是保证系统能正常准确地运行，即使发生意外事件，系统也不会崩溃。尽管在软件界永远不可能保证 100% 的正确性，但是 Java 所拥有的特性确实有力地保证了系统运行时的健壮性。

- 在 JVM 上运行的程序不能直接访问机器本地系统资源，除非通过 JVM 支持的 API。因此，Java 程序不能访问不允许应用程序使用的内存区域，不能错误调用 API，不能执行一些比较低级的操作，因为这些操作常常是用其他语言编制的程序出错的原因。
- JVM 执行自动内存单元管理。Java 语言能很大程度地降低滥用内存单元的机率，而其他的语言一般都明确指出内存的申请和释放必须由程序员来完成。程序员很难任意访问内存单元，可 JVM 却可以实时检查内存中可用的界限。因此，程序从来不会在不该访问内存单元的时候进行访问，或错误调用了他们本不想调用的内存单元。
- Java 平台具有一个强大的管理机制用来处理错误和意外事件。当某个操作不能被完全执行时，用其他编程语言编制的程序往往会过早或突然地终止。在 JVM 中有一个处理错误信息和意外事件的对象叫 Exception(异常)。这个对象记录了所有发生过的意外事件的类型和原因。程序员可为所有类型的意外事件编写自己所需的程序。例如，当程序试图读取一个不存在的文件时，错误处理程序就会捕捉到错误信息并做出适当处理。尽管错误处理模块不是 Java 语言所独有的，但它在 Java 环境中的使用却有更可靠的表现。
- Java 语言是类型安全的，也就是说，不能随意得使用某个信息，所有的操作都应基于该信息所定义的类型。也不能玩弄投机取巧的小伎俩，因为这种做法在其他语言中是许多程序出错的罪魁祸首。例如，不能用你的数据结构覆盖系统中的数据结构，然后再对其进行修改。

Java 的安全性 JVM 可以运行放置在本地机器上的代码(通常为本地应用程序)，也可以运行从网络中下载的代码(通常为一个 applet)。作为默认规定，本地代码为可信代码，远程代码为不可信代码。可以接受默认的安全策略，也可以建立更细致周密的访问控制。例如，可以限定某人只可运行特定的 servlet。

附加的安全策略控制代码的使用：

- JVM 在一个沙盒(sandbox)中运行不可信代码。沙盒为 JVM 分配专门运行有限特权代码的区域。沙盒中的代码只有很少的权限来访问文件系统，而且不允许进行任何损害 JVM 或本地系统完整性的操作。例如，不可信代码不能写入本地文件，不能直接访问本地代码，不可以读取特定的系统属性。
- 从不可信站点下载的字节码(如未知服务器)，在执行前，必须先要进行检验。检验系统是一个安全机制。它必须保证字节码文件的构造是正确的。例如，它必须核查每条指令都有一个合法操作码。
- 如果 applet 具有一个可识别的数字签名密钥，JVM 也能接受 applet。

Java 的结构中立性和可移植性 通过限制使用依赖于平台的特性，Java 平台提供中性化的结构。例如，C 和 C++ 的基本数据类型的内存存储单元的分配由特定的计算机系统和编译器决定，而对于 Java，基本数据类型的存储器交换是由 Java 平台定义的，并适用于在所有类型的计算机系统上的实施。

可移植性是结构中立性结构的最大优点。字节码是可移植性的，因为它可以运行在所有的 JVM 上。

任何 JVM 都可以运行合法的 Java 程序，而不管该程序是在什么样的 Java 平台开发的，而且所有的 JVM 都会为该程序提交一样的处理结果。

不过，可移植性也有它的代价。结构中立性的代码不能使用任何平台的最新特性。因为在特性、性能和可移植性中，总是存在着不能同时满足性和交替性。

Java 的高性能性 Java 具有非常高的性能，而为实现性能所付的代价也非常小，因为将 Java 程序编译为字节码的付出是微不足道的。

JIT(just-in-time)编译器可以提高 Java 程序的性能。JIT 编译器是一个基于特定平台的程序，被插入在 JVM 中，用来将结构中立性的字节码转化为特定平台的对象代码。当运行 JIT 编译的程序时，它会绕过字节码的解释程序，从而大大提高程序的性能。另外，包含循环语句的程序通过 JIT 编译后，运行速度会更快，而字节码就需要对循环语句的每个循环分别进行解释。

Java 的多线程性 Java 平台具有专门支持多线程程序的类。在第 11 章中，将详细介绍 API 和多线程程序的关键字。如果你以前曾用别的语言编写过多线程程序，那么将会异常欣喜地发现，在别的语言中如此困难的多线程程序的编写、运行和同步问题的解决，在 Java 语言中是如此得简单。

Java 语言的动态性 在 Java 中，根据使用的环境不同，动态一词有不同的解释：

- 在运行时，JVM 动态地将程序所需的类加载进来。只有当类第一次被使用时，JVM 才将其加入。JVM 不需要将程序使用的所有类都加载进来，也不需在程序执行前将所有类都准备出来。
- 当同一个继承层次支持一个以上的类调用方法时，JVM 使用动态绑定来决定使用哪个调用方法。在第 15 章中，将详细讲解动态绑定，它也是一个被许多面向对象的编程语言使用的重要技术。
- Java 软件的开发过程也具有动态的性质。程序员可采用增量的方式构建类集和创建、测试程序，并用不同的方法进行组合。

1.5 Java 技术的发展历史

Java 发展的历史并不是很长。20 世纪 90 年代初，当时 Sun 公司正在从事电子消费品方面的经营。为了节约成本，消费电子产品上的软件，一般都是运行在当时市面上最低廉的芯片上。一旦新的更优惠的芯片出现，厂家立刻会转换到该芯片上，而基于芯片的软件开发也必须马上改变。Sun 公司一直致力于研制一种新的可移植的软件，以便尽快地适用到新的更低廉的芯片上。

为了实现这个目的，Sun 公司研制了一个实验性的便携式电脑叫*7，计划变成一个新型的统一控制家用电器的产品。James Gosling 是这个项目小组的主要负责人。尽管*7 从来没有变成一个零售产品，但是 Sun 公司的开发者却发现，他们研制的计算机语言有更大的其他用途。例如，可以应用于 Web。因为对于 Web，紧密性和简单性是最重要的需求。其他人则预见将其应用于家用电器的内置系统、企业级应用程序和网络计算。

一开始，他们将该语言称作 Oak(橡树)，据说是为了纪念 Gosling 窗外的大橡树。但是当开发者们听说已经有一个语言叫 Oak 了，他们只好再改名，最后敲定为 Java，也许是为了感谢咖啡在他们编程过程中的巨大帮助。一个更有意思的现象是，Java 还演变出了 Java 语言中的其他名称，如“beans”，即咖啡豆，看样子 Java 确实比 Oak 更具有潜力。

Java 编程语言的发布，以空前的力量吸引了整个软件开发界的兴趣。以下是 Java 平台自开发以来的简短历史介绍：

- 1995 年 5 月，Sun 公司正式发布 Java 编程语言。在此之前的两个版本分别为 alpha 和 beta。Java 平台第一次正式发布的开发工具套件和 API 叫做 Java 开发工具(Java Development Kit, JDK)。
- 自从 1995 年以来，Java 语言的使用者以一个可观的速度在增长。当第一个 Java 平台推出后，许多流行的 Web 浏览器厂商迅速为其浏览器添加了内置的 1.0 JVM 来支持 applet。

- 1997 年, Sun 公司推出了 1.1 版本。这个版本在 Java 开发语言中增添了许多特性, 尤其突出的是, 对于类集的补充和修改, 最后形成了 Java API。随后, Sun 又发布了该版本的 JavaBean 标准规范。几个月之后, 众多的流行 Web 浏览器开发商迅速升级了他们内置的 JVM, 以便与 1.1 版本兼容。
- 从 1997~1999 年的三年之间, Sun 公司多次升级了 JDK 的版本。1998 年, Sun 将 Java 基础类 Java Foundation Classes, JFC)开发成了一个独立的软件包。它随即被用到了 JDK 1.1.6 及以后更新的版本。
- 1998 年前期, Sun 公司推出了 Java Servlet 开发工具包(Java Servlet Development Kit, JSDK)上的 servlet API。与此同时, 市面上迅速出现了支持 servlet 的 Web 服务器。一些服务器, 如 IBM 的 WebSphere 还推出了能同时支持更早规范的 Web 服务器。
- 早在 1998 年下半年之前, Java 2 平台以已为人们热切期望, 并被预先称为 JDK 的 1.2 版本。当 Java 2 平台推出后, Java 开发工具和 API 被改名为 Java 2 SDK, 标准版本。JFC 被完全集成到了 Java 2 中。Java 2 平台在以下方面进行了突出改善: 数据库访问、分布式计算、与 CORBA 的互用性和 Web 浏览器上的 Java 2 插件 JVM。
- 大约 1998 年末, EJB 的第一个标准规范完成。随之, 一些企业开始采用支持 EJB 的 EJS。
- 1999 年 6 月, Sun 公司推出了 3 个版本的 Java 2 平台。第 1 个是 Java 2 标准版(Java 2 Standard Edition, J2SE), 也就是本书常提到的 SDK。第 2 个版本是 Java 2 企业版(Java 2 Enterprise Edition, J2EE), 它涵盖了企业级 Java Bean (Enterprise Java Bean, EJB)的内容, 并规范了 Web 应用程序的概念。Web 应用程序是基于 Web 的解决方案, 由 Java 程序、HTML 网页和其他可被配置到 N 层企业环境的资源构成。第 3 个版本是 Java 微版本, 主要适用于 Web 电话、智能卡和其他便携式电子产品。
- 1999 年 8 月, Sun 公司发布了 JSP 1.0 的标准规范和与之配套的 Java 服务器 Web 开发套件, 作为这项规范的引用实施。

一个有意思的现象是, 尽管 Sun 公司已经推出了最新的版本 Java 2, 但它还在继续进行一些在 1.1 版本上的开发。看样子, 它要一直支持 1.1 版本, 直到所有的使用厂商完全升级到 Java 2。

许多厂商都已开始使用 Java 应用程序开发工具和相关产品。能从许多地方购买或下载到 JavaBean 的资料。

1.6 Java 工 具

Java 平台包括一个开发 Java 程序的工具集。SDK 工具只运行在命令行的窗口中和提供基本的功能。不过, 这些对于开发 Java 程序来说已经足够了, 而且可以利用它来实践整本书的内容。

通过互联网来查找有关开发方面的资料是一个非常好的方法, 在不同的网站上, 可以找到技术支持、实例和演示、科技文章、常见问题解答、在线教程和升级等方面的资料。

下面介绍一些 Java 平台最常用的工具。本书列举的所有的命令范例可以运行在 Java 平台的 Windows 版本上。虽然本书的大多数命令范例是跨平台的, 但是命令中的选项和语法会因文件系统或主机操作系统的命令行处理器的不同而不同。所有工具的最完整的描述, 可在 Java 2 平台的安装文档中找到。

- `javac` 是 Java 编译器。`javac` 将 Java 应用程序、applet 和 servlet 编译成字节码。使用 `javac` 时, 必须导入扩展名为 `.java` 的源代码文件。编译后的字节码文件扩展名为 `.class`。
Java 平台为源代码文件和类设定了严格的命名规则。如果某个类可以像应用程序一样被运行, 或被与其不在同一包中的其他类调用, 那么这个类的名称必须与它的源代码文件名一样。即使 Java 程序运行的操作系统不需要文件名区分大小写, 所有的 Java 文件名称仍必须区分大小写, 源代码文件名与其类名必须严格一样, 必须区分大小写。文件编译时, 必须指明源文件的绝对