

SAMS
**Teach
Yourself**

- 全球销量逾百万册的系列图书
- 连续十余年打造的经典品牌
- 直观、循序渐进的学习教程
- 掌握关键知识的最佳起点
- 秉承Read Less, Do More (精读多练)的教学理念
- 以示例引导读者完成最常见的任务

每章内容针对初学者精心设计, **1**小时轻松阅读学习,
24小时彻底掌握关键知识


涵盖
Java 7和
Android
知识

每章 **案例与练习题** 助你轻松完成常见任务,
通过 **实践** 提高应用技能, 巩固所学知识

Java

入门经典 (第6版)

[美] Rogers Cadenhead 著
梅兴文 郝记生 译

 **人民邮电出版社**
POSTS & TELECOM PRESS

学习如何：

- 设置Java编程环境；
- 在短短几分钟内编写出第一个可运行的程序；
- 控制程序决策和行为；
- 存储和处理信息；
- 构建直观的用户界面；
- 创建交互式Web程序；
- 使用线程创建迅速响应的程序；
- 读写文件和XML数据；
- 掌握面向对象编程的最佳做法；
- 使用JAX-WS创建灵活、可以互操作的Web服务；
- 使用Java创建Android app。

24章阶梯教学

通过阅读本书，读者将学会如何使用免费的NetBeans可视化编辑工具来创建Java应用程序。本书采用直观、循序渐进的方法，引导读者掌握创建桌面和Web应用程序、Web服务，甚至是Android app所需要的所有技巧和技术。本书每章内容都建立在已学的知识之上，即使读者没有任何Java编程经验，也可以通过本书，走向成功的坚实道路。

循序渐进的示例引导读者完成最常见的任务。

问与答、测验和练习帮助读者检验知识的掌握情况。

“注意”、“提示”和“警告”指出捷径和解决方案。

美术编辑 王建国

分类建议：计算机/程序设计/ Java
人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-27181-5



9 787115 271815 >

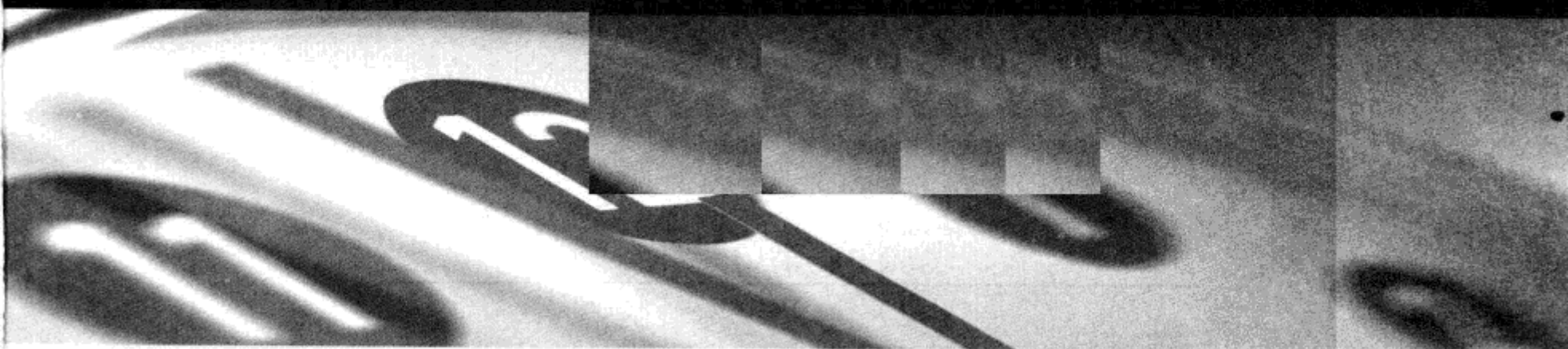
ISBN 978-7-115-27181-5

定价：39.00 元

Java

入门经典 (第6版)

[美] Rogers Cadenhead 著
梅兴文 郝记生 译



人民邮电出版社
北京

图书在版编目 (C I P) 数据

Java入门经典 : 第6版 / (美) 卡登海德
(Cadenhead, R.) 著 ; 梅兴文, 郝记生译. — 北京 : 人
民邮电出版社, 2012.3
ISBN 978-7-115-27181-5

I. ①J… II. ①卡… ②梅… ③郝… III. ①
JAVA语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2011)第258605号

版 权 声 明

Rogers Cadenhead: Sams Teach Yourself Java in 24 Hours (Six Edition)

ISBN: 0672335751

Copyright © 2012 by Sams Publishing.

Authorized translation from the English languages edition published by Sams.

All rights reserved.

本书中文简体字版由美国 Sams 出版公司授权人民邮电出版社出版。未经出版者书面许可, 对本书任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。

Java 入门经典 (第 6 版)

- ◆ 著 [美] Rogers Cadenhead
译 梅兴文 郝记生
责任编辑 傅道坤
 - ◆ 人民邮电出版社出版发行, 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京铭成印刷有限公司印刷
 - ◆ 开本: 787×1092 1/16
印张: 19
字数: 470 千字
印数: 1—3 500 册
- 2012 年 3 月第 1 版
2012 年 3 月北京第 1 次印刷

著作权合同登记号 图字: 01-2011-4670 号

ISBN 978-7-115-27181-5

定价: 39.00 元

读者服务热线: (010)67132705 印装质量热线: (010)67129223

反盗版热线: (010)67171154

广告经营许可证: 京崇工商广字第 0021 号

内容提要

本书通过大量示例程序循序渐进地引导读者快速掌握使用 Java 开发程序的基本技能。

本书总共 24 章，先讲解了 Java 程序的编写流程、工作原理等内容；然后介绍了有关 Java 编程的基本知识，包括变量、条件语句、循环语句、数组和对象等内容；随后介绍了创建图形用户界面、编写交互式 Web 程序、读写文件，以及使用字体、颜色和图形等相关的知识。本书还介绍了如何使用 Java 来开发 Android app。本书每章都提供了示例程序清单，并辅以示例输出和代码分析，以阐述该章介绍的主题。为加深读者对所学内容的理解，每章末尾都提供了常见问题及其答案以及练习和测验。

本书可作为初学者学习 Java 编程技术的教程，也可供其他语言的程序员学习 Java 时参考。

新学网
PDG

关于作者

Rogers Cadenhead 是一名作家、计算机程序员、Web 开发人员，他已经编写了 20 多本与 Internet 相关的图书，其中包括《Sams Teach Yourself Java in 21 Days》。他维护着 Drudge Retort 和其他站点，这些站点的年访问量有 2000 万。本书的官方站点是 www.java24hours.com。



献 词

这一次我要打破传统，不再将本书献给我的家人和朋友，而是献给 James Gosling、Mike Sheridan、Kim Polese、Bill Joy 等人，是你们在 1995 年发明了令人惊叹的 Java 编程语言。当我第一次看到 Java 语言在 Web 页面中运行时，就惊叹不已，现在使用该语言编写的 app 更是运行在全世界数百万台 Android 手机上。Java 语言的出现证明了你们在 Sun 公司卓越工作的价值。

致 谢

感谢Sams出版社的员工，尤其是Mark Tabe、Songlin Qiu、Tonya Simpson、Charlotte Kughen 和Boris Minkin。没有作者可以凭借一己之力来写出一本书。他们出色的工作给了我很多帮助。

谢谢我的妻子 Mary 和儿子 Max、Eli、Sam。尽管他们没有帮我实现当高空飞人杂技演员的梦想，但是在我们这样一个恐高症家庭中，我依然是世界上最自豪的丈夫和父亲。



前言

作为一名计算机图书作者，我花费了大量的时间呆在书店的计算机图书区，在假装阅读最新一期《In Touch Weekly》杂志的同时，观察读者阅读图书的行为。

根据我的观察，读者拿起本书并翻到前言后，在他将书放下前往咖啡厅喝杯咖啡前，给我留下的时间只有大约 12 秒。

因此，这里长话短说：使用 Java 来进行计算机编程比想象的容易。我本不应该这样说，因为数以千计的程序员正是凭借其 Java 技能在软件开发、Web 应用程序编程和移动 app 开发领域获得了高薪职位，对他们来说，最不想让老板知道的是，只要坚持不懈并有一点空闲时间，任何人都能学会当前使用最为广泛的编程语言。通过阅读本书，你可以快速掌握 Java 编程。

任何人都能学会如何编写计算机程序，尽管他们不能对 DVR 进行编程。Java 是最值得学习的编程语言之一，因为这种功能强大的实用技术已被全球大量的程序员采用。

本书是为非程序员、讨厌学习编程的程序员新手，以及经验丰富但想快速掌握 Java 的老程序员编写的。本书使用的是该语言的最新版本——Java 7。

由于 Java 具有“让一切成为可能”的特性，因此成为一种非常流行的编程语言。你可以使用 Java 来创建具有图形用户界面的程序、设计充分利用 Internet 的软件、读取 XML 数据、开发在 Android 手机上运行的游戏，等等。

本书引导读者从零开始学习 Java 编程，它以平实的语言阐述概念，并包含大量要求读者逐步创建的示例程序。读完本书，读者就能编写自己的 Java 程序，对自己使用该语言的能力充满信心，进而更深入地学习它；读者还将获得日益重要的技能，如网络计算、图形用户界面设计和面向对象编程。

就当前而言，这些术语对你来说也许并不重要。事实上，正是它们使得人们对编程充满了恐惧，并认为很难掌握。然而，如果你能使用计算机计算账目的收支平衡，或者能在 Facebook 上创建电子相簿，通过阅读本书就能够编写计算机程序。

现在，如果你还是愿意去喝咖啡而不是学习 Java，请将本书放回书架，并将封面朝向书店中人来人往的通道，以方便别人找到它。

目 录

第 1 章 成为程序员	1	2.8 运行 Java 程序.....	15
1.1 选择编程语言.....	2	2.9 总结	16
1.2 告诉计算机做什么	3	2.10 问与答.....	16
1.3 程序的工作原理.....	5	2.11 测验	17
1.4 为什么程序不能正常工作.....	5	2.11.1 问题	17
1.5 选择 Java 编程工具	5	2.11.2 答案	17
1.6 安装 Java 开发工具	6	2.12 练习	18
1.7 总结	6	第 3 章 Java 之旅	19
1.8 问与答.....	7	3.1 第一站: Oracle	19
1.9 测验.....	7	3.2 去 Java 学校.....	21
1.9.1 问题	7	3.3 在 JavaWorld 用午餐.....	22
1.9.2 答案	7	3.4 在 NASA 仰望天穹.....	24
1.10 练习	8	3.5 回归正题	24
第 2 章 编写第一个程序	9	3.6 到 Java Boutique 去问路	25
2.1 编写程序所需的工具.....	9	3.7 在手机上运行 Java.....	26
2.2 创建 Saluton 程序	10	3.8 总结	27
2.3 开始输入程序.....	10	3.9 问与答.....	27
2.3.1 class 语句	11	3.10 测验	28
2.3.2 main 语句的作用	12	3.10.1 问题	28
2.3.3 大括号	12	3.10.2 答案	28
2.4 在变量中存储信息.....	13	3.11 练习	28
2.5 保存编写好的程序.....	13	第 4 章 理解 Java 程序的工作原理	29
2.6 将程序编译为 class 文件.....	14	4.1 创建应用程序.....	29
2.7 修复错误.....	14	4.2 向应用程序传递参数.....	30

4.3 创建 applet.....	32	6.10 测验.....	56
4.4 总结.....	34	6.10.1 问题.....	56
4.5 问与答.....	34	6.10.2 答案.....	56
4.6 测验.....	34	6.11 练习.....	56
4.6.1 问题.....	34		
4.6.2 答案.....	35	第 7 章 使用条件测试进行判断	57
4.7 练习.....	35	7.1 if 语句.....	58
第 5 章 在程序中存储和修改信息	36	7.1.1 小于和大于的比较.....	58
5.1 语句和表达式.....	36	7.1.2 相等和不等.....	58
5.2 指定变量类型.....	37	7.1.3 使用块语句组织程序.....	59
5.2.1 整数和浮点数.....	37	7.2 if-else 语句.....	60
5.2.2 字符和字符串.....	38	7.3 switch 语句.....	61
5.2.3 其他数值类型的变量.....	39	7.4 条件运算符.....	63
5.2.4 布尔型变量.....	39	7.5 观察时钟.....	63
5.3 给变量命名.....	40	7.6 总结.....	66
5.4 在变量中存储信息.....	40	7.7 问与答.....	66
5.5 运算符.....	41	7.8 测验.....	67
5.5.1 变量的递增与递减.....	42	7.8.1 问题.....	67
5.5.2 运算符优先级.....	43	7.8.2 答案.....	67
5.6 使用表达式.....	44	7.9 练习.....	68
5.7 总结.....	46	第 8 章 使用循环重复执行操作	69
5.8 问与答.....	46	8.1 for 循环.....	69
5.9 测验.....	47	8.2 while 循环.....	71
5.9.1 问题.....	47	8.3 do-while 循环.....	72
5.9.2 答案.....	47	8.4 退出循环.....	73
5.10 练习.....	47	8.5 给循环命名.....	73
第 6 章 使用字符串来交流	48	8.6 测试计算机的运行速度.....	75
6.1 在字符串中存储文本.....	48	8.7 总结.....	76
6.2 在程序中显示字符串.....	49	8.8 问与答.....	76
6.3 在字符串中使用特殊字符.....	50	8.9 测验.....	76
6.4 拼接字符串.....	50	8.9.1 问题.....	77
6.5 将其他变量用于字符串中.....	51	8.9.2 答案.....	77
6.6 字符串的高级处理.....	52	8.10 练习.....	77
6.6.1 比较两个字符串.....	52	第 9 章 使用数组存储信息	78
6.6.2 确定字符串的长度.....	52	9.1 创建数组.....	79
6.6.3 改变字符串的大小写.....	53	9.2 使用数组.....	80
6.6.4 查找字符串.....	53	9.3 多维数组.....	81
6.7 导演及演员名单.....	54	9.4 对数组进行排序.....	82
6.8 总结.....	55	9.5 对字符串中的字符计数.....	83
6.9 问与答.....	55	9.6 总结.....	85

9.7 问与答.....	85	11.9.2 答案.....	110
9.8 测验.....	86	11.10 练习.....	110
9.8.1 问题.....	86		
9.8.2 答案.....	86		
9.9 练习.....	86		
第 10 章 创建第一个对象	87	第 12 章 充分利用现有对象	111
10.1 面向对象编程的工作原理.....	87	12.1 继承的威力.....	111
10.2 对象示例.....	88	12.1.1 继承行为和属性.....	112
10.3 什么是对象.....	89	12.1.2 覆盖方法.....	112
10.4 理解继承.....	90	12.2 建立继承.....	113
10.5 建立继承层次.....	90	12.3 使用现有的对象.....	114
10.6 转换对象和简单变量.....	91	12.4 将相同类型的对象存储到 Vector 中.....	115
10.6.1 简单变量的类型转换.....	92	12.5 创建子类.....	117
10.6.2 对象类型转换.....	92	12.6 总结.....	119
10.6.3 在简单变量和对象之间 进行转换.....	93	12.7 问与答.....	119
10.6.4 自动封装和拆封.....	94	12.8 测验.....	120
10.7 创建对象.....	95	12.8.1 问题.....	120
10.8 总结.....	97	12.8.2 答案.....	120
10.9 问与答.....	97	12.9 练习.....	120
10.10 测验.....	97		
10.10.1 问题.....	97	第 13 章 创建简单的用户界面	121
10.10.2 答案.....	98	13.1 Swing 和抽象窗口工具包.....	121
10.11 练习.....	98	13.2 使用组件.....	122
第 11 章 描述对象	99	13.2.1 窗口和框架.....	122
11.1 创建变量.....	99	13.2.2 按钮.....	125
11.2 创建类变量.....	101	13.2.3 标签和文本框.....	127
11.3 用方法来创建行为.....	102	13.2.4 复选框.....	127
11.3.1 声明方法.....	102	13.2.5 组合框.....	128
11.3.2 参数不同的类似方法.....	103	13.2.6 文本区域.....	129
11.3.3 构造函数.....	103	13.2.7 面板.....	130
11.3.4 类方法.....	104	13.3 创建自己的组件.....	130
11.3.5 方法中变量的作用域.....	105	13.4 总结.....	133
11.4 将一个类放在另一个类中.....	105	13.5 问与答.....	133
11.5 使用关键字 this.....	106	13.6 测验.....	133
11.6 使用类方法和类变量.....	107	13.6.1 问题.....	133
11.7 总结.....	109	13.6.2 答案.....	134
11.8 问与答.....	109	13.7 练习.....	134
11.9 测验.....	109		
11.9.1 问题.....	109	第 14 章 用户界面的布局	135
		14.1 使用布局管理器.....	135
		14.1.1 GridLayout 管理器.....	137
		14.1.2 BorderLayout 管理器.....	137
		14.1.3 BoxLayout 管理器.....	138

14.1.4 使用 Insets 将组件隔开	139	17.2 将 applet 放到 Web 页面中	172
14.2 应用程序的界面布局	139	17.3 创建 applet	173
14.3 总结	143	17.3.1 在 applet 窗口中绘画	174
14.4 问与答	143	17.3.2 测试 SalutonApplet 程序	174
14.5 测验	143	17.4 从 Web 页面传递参数	175
14.5.1 问题	144	17.5 在 applet 中接收参数	176
14.5.2 答案	144	17.6 在 applet 中处理参数	176
14.6 练习	144	17.7 使用 object 标记	178
第 15 章 响应用户输入	145	17.8 总结	178
15.1 让程序监听	145	17.9 问与答	179
15.2 设置要监听的组件	146	17.10 测验	179
15.3 处理用户事件	146	17.10.1 问题	179
15.3.1 复选框和组合框事件	147	17.10.2 答案	179
15.3.2 键盘事件	148	17.11 练习	180
15.3.3 启用和禁用组件	149	第 18 章 处理程序中的错误	181
15.4 完善图形应用程序	150	18.1 异常	181
15.5 总结	157	18.1.1 在 try-catch 块中捕获异常	182
15.6 问与答	157	18.1.2 捕获多种不同的异常	184
15.7 测验	157	18.1.3 出现异常后进行处理	186
15.7.1 问题	158	18.1.4 抛出异常	186
15.7.2 答案	158	18.1.5 忽略异常	188
15.8 练习	158	18.2 抛出和捕获异常	188
第 16 章 创建复杂的用户界面	159	18.3 总结	190
16.1 滚动窗格	159	18.4 问与答	190
16.2 滑块	161	18.5 测验	191
16.3 变更监听器	162	18.5.1 问题	191
16.4 使用图像图标和工具栏	165	18.5.2 答案	191
16.5 总结	168	18.6 练习	191
16.6 问与答	169	第 19 章 创建线程程序	192
16.7 测验	169	19.1 线程	192
16.7.1 问题	169	19.1.1 降低程序的速度	192
16.7.2 答案	169	19.1.2 创建线程	193
16.8 练习	169	19.2 使用线程	196
第 17 章 创建交互式 Web 程序	170	19.2.1 声明类	196
17.1 标准 applet 方法	170	19.2.2 创建变量	197
17.1.1 在 applet 窗口中绘画	171	19.3 从 init() 开始	197
17.1.2 初始化 applet	172	19.4 在创建 URL 时捕获错误	198
17.1.3 启动和停止 applet	172	19.5 在 paint() 方法中处理屏幕更新	198
17.1.4 销毁 applet	172	19.6 启动线程	199

19.6.1 运行线程.....	199	22.2 创建服务实现 Bean.....	230
19.6.2 停止线程.....	200	22.3 发布 Web 服务.....	231
19.7 处理鼠标单击.....	200	22.4 使用 Web 服务描述语言文件...	232
19.8 循环显示链接.....	201	22.5 创建 Web 服务客户端.....	234
19.9 总结.....	203	22.6 总结.....	236
19.10 问与答.....	203	22.7 问与答.....	236
19.11 测验.....	203	22.8 测验.....	237
19.11.1 问题.....	203	22.8.1 问题.....	237
19.11.2 答案.....	204	22.8.2 答案.....	237
19.12 练习.....	204	22.9 练习.....	237
第 20 章 读写文件.....	205	第 23 章 创建 Java2D 图形.....	239
20.1 流.....	205	23.1 使用 Font 类.....	239
20.1.1 文件.....	206	23.2 使用 Color 类.....	240
20.1.2 从流中读取数据.....	207	23.3 创建自定义颜色.....	241
20.1.3 缓冲输入流.....	209	23.4 绘制直线和形状.....	241
20.2 将数据写入流中.....	211	23.4.1 绘制直线.....	242
20.3 读写配置属性.....	212	23.4.2 绘制矩形.....	242
20.4 总结.....	215	23.4.3 绘制椭圆和圆.....	243
20.5 问与答.....	215	23.4.4 绘制弧线.....	243
20.6 测验.....	215	23.5 绘制饼图.....	244
20.6.1 问题.....	215	23.6 总结.....	249
20.6.2 答案.....	216	23.7 问与答.....	250
20.7 练习.....	216	23.8 测验.....	250
第 21 章 读写 XML 数据.....	217	23.8.1 问题.....	250
21.1 创建 XML 文件.....	217	23.8.2 答案.....	251
21.2 读取 XML 文件.....	220	23.9 练习.....	251
21.3 读取 RSS 聚合内容(Syndication Feeds).....	224	第 24 章 编写 Android app.....	252
21.4 总结.....	225	24.1 Android 简介.....	252
21.5 问与答.....	226	24.2 创建 Android app.....	254
21.6 测验.....	226	24.2.1 剖析一个 Android 新项目.....	255
21.6.1 问题.....	226	24.2.2 创建 app.....	256
21.6.2 答案.....	226	24.2.3 安装 Android 模拟器.....	257
21.7 练习.....	227	24.2.4 创建调试配置.....	259
第 22 章 利用 JAX-WS 开发 Web 服务.....	228	24.3 运行 app.....	260
22.1 定义服务端点接口.....	228	24.4 设计真实的 app.....	261
使用注解来简化 Java 代码.....	229	24.4.1 组织资源.....	262
		24.4.2 配置 app 的 Manifest 文件.....	264
		24.4.3 设计用户界面.....	265
		24.4.4 编写 Java 代码.....	267

24.5 总结.....	272	B.3.2 Caféau Lait.....	282
24.6 问与答.....	272	B.3.3 Workbench.....	283
24.7 测验.....	273	B.3.4 Java 7Developer Blog.....	283
24.7.1 问题.....	273	B.3.5 其他 Java 博客.....	283
24.7.2 答案.....	273	B.3.6 InformIT.....	283
24.8 练习.....	273	B.3.7 Stack Overflow.....	283
附录 A 使用 NetBeans IDE.....	275	B.3.8 Java Review Service.....	283
A.1 安装 NetBeans.....	275	B.3.9 JavaWorld 杂志.....	283
A.2 创建新项目.....	276	B.3.10 Developer.com's Java Directory.....	284
A.3 创建新的 Java 类.....	277	附录 C 本书站点.....	285
A.4 运行应用程序.....	278	附录 D 设置 Android 开发环境.....	287
A.5 修复错误.....	279	D.1 起步.....	287
附录 B Java 资源.....	281	D.2 安装 Eclipse.....	287
B.1 可以考虑的其他书.....	281	D.3 安装 Android SDK.....	288
B.2 Oracle 公司的 Java 官方站点.....	282	D.4 安装在 Eclipse 中使用的 Android 插件.....	289
B.3 其他 Java 站点.....	282	D.5 设置你的手机.....	291
B.3.1 本书英文版的配套网站.....	282		



第 1 章

成为程序员

本章介绍如下内容：

- 选择要学习的编程语言；
- 使用程序来让你的计算机运转起来；
- 程序的工作原理；
- 修复程序中的错误；
- 选择 Java 开发工具；
- 为编写程序做好准备。

你可能听到过下面有关计算机编程的说法：计算机编程极其困难，它要求你获得计算机科学专业的学位，需要投入几千美元来购买计算机硬件和软件，需要极强的分析能力，需要有耐心，而且对含咖啡因的饮料有强烈的爱好。

其实，在上述条件中，除喜欢喝含咖啡因的饮料外，其他所有条件都是不正确的。尽管多年以来计算机程序员一直都说编程很难，而且这种说法可以让程序员更容易找到高薪工作，但是，编程工作其实比大多数人想象的要容易。

现在是学习编程的最佳时代。我们可以从 Internet 上免费下载各种编程工具，数以千计的程序员以开源的形式发布他们编写的程序，以方便他人检查并修正程序中的错误，以及对程序进行改进。即使是在经济紧缩时期，许多公司也仍然在招聘程序开发人员。

如今，数以百万计的移动设备采用的都是 Android（其所有应用程序都以 Java 语言编写的一种操作系统）。如果你有一台 Android 手机，每当你在搜索电影、进行导航服务，以及玩“愤怒的小鸟”游戏时，你用到的都是 Java 程序员开发的软件。

本书旨在向两类人群讲授 Java 编程知识：没有任何编程知识的人，有过编程经验但是又讨厌编程（如同伏地魔讨厌英国孤儿院的孩子一样）的人。本书尽可能采用通用语言而不是

行话或者晦涩的首字母缩写，并对新出现的编程术语进行详细解释。

如果这种尝试得以成功，那么读者在读完本书后将获得丰富的编程技能，而这些技能对读者来说曾经是一种挑战。阅读本书后，读者将能够编写程序，更自信地阅读其他编程图书，从而更轻松地学习新的编程语言（请注意，我说的是“编程”语言，而不是西班牙语、法语以及克林贡这种外星语言）。当然，读者也会掌握 Java 这种应用最为广泛的编程语言。

本章将会简单介绍编程的概念，然后讲解计算机的一些设置指令，以为编写 Java 程序做好准备。

1.1 选择编程语言

如果你能够娴熟地使用计算机来准备一个漂亮的简历，或者是计算账目的收支平衡，抑或是将你在假期中拍摄的照片上传到 Facebook 上与他人共享，那么你就能在计算机上编写程序。

学习计算机编程的关键是选择一门合适的语言，而编程语言的选择往往与所要完成的任务有关。每一种编程语言都非尽善尽美。多年以来，人们在学习编程时，使用的都是各种版本的 BASIC 编程语言，原因是 BASIC 针对的就是初学者。

By the Way

注意：

BASIC 语言发明于 20 世纪 60 年代，它的初衷是方便学生和初学者学习，BASIC 中的字母 B 代表初学者（Beginner's）。使用 BASIC 语言的缺点是容易养成马虎的编程习惯。

人们使用 Microsoft Visual Basic 编写了数千个复杂的用于商业目的或个人用途的程序。但是，使用 Visual Basic 的各种版本编写的程序在运行速度上要比其他语言（比如 C# 和 Visual C++）编写的慢。当程序中使用了大量的图形时（比如游戏），这种速度差异更为明显。

本书讲解的 Java 编程语言是由 Oracle 公司开发的（译者注：Java 语言其实是由 Sun 公司开发的，但是 Sun 公司于 2010 年被 Oracle 公司收购，所以作者在这里说“Java 编程语言是由 Oracle 开发的”）。虽然 Java 语言学习起来比 Visual Basic 等语言困难，但基于几个原因，将其作为入门语言仍然是不错的选择。学习 Java 的优点之一是你可以在 Web 和移动手机上使用 Java 语言。你可以使用 Java 语言来为 Android 手机开发应用程序，可以开发网页游戏，还可以将其用在其他热门的软件开发领域。

Java 语言的另外一个重要的优点是：程序要想运行，则程序语句必须要组织有序。如果你不遵守这些规则，则编写的 Java 程序将无法运行。

刚开始编写 Java 程序时，你可能觉得 Java 的这种挑剔行为不是优点。编写完程序后，你可能要修改多处错误才能让程序通过编译，从而对这种程序编写方式产生厌烦心理。

在接下来的几章中，将介绍 Java 的这些规则以及应避开的陷阱。这些额外工作带来的好处是，程序将更可靠、更有用且没有错误。

Java 是由 Sun 公司的 James Gosling 开发设计的，旨在提供一种更好的计算机编程方式。Gosling 在开发一个项目时，不喜欢编程语言 C++ 的工作方式，因此决定创建一种能够更好地

完成这项工作的新语言。虽然在 Java 是否优于其他编程语言这一点上存在分歧，但 Java 在过去 10 年中取得的成就已经证明了它本身的优势。世界上 30 亿台设备运行的是 Java 程序。从 Java 面世到现在，已经出版了 1000 多种与它相关的图书（这是我们的第 16 本 Java 图书）。

不管 Java 是不是最好的语言，但肯定是当今最值得学习的编程语言。在第 2 章，读者将有机会尝试编写 Java 程序。

学习任何一种编程语言后，再学习其他编程语言将会比较容易。很多语言彼此相似，因此在学习新语言时，不用完全从头开始。例如，很多 C++ 或 Smalltalk 程序员发现学习 Java 相当容易，因为 Java 从这些编程语言中借鉴了许多思想。同样，C# 编程语言也从 Java 借鉴了很多思想，所以 Java 程序员也很容易就能学会 C# 编程语言。

注意：

本章多次提到了 C++，也许读者不明白它指的是什么，以及它是如何发音的。C++ 可以读成“C 加加”，它是由贝尔实验室的 Bjarne Stroustrup 开发的一种编程语言。C++ 是 C 语言的增强版，这就是 C++ 中“++”的含义。那为什么不叫 C+ 呢？本书后面将介绍，++ 是一个计算机编程笑话。

By the Way

1.2 告诉计算机做什么

计算机程序也叫软件，它告诉计算机该做什么。计算机执行的任何操作（从启动到关机）都是由程序控制的。Windows 7 是程序，Call of Duty 游戏是程序，安装的打印机驱动软件是程序，甚至邮件病毒也是程序。

计算机程序由一系列命令组成，程序运行时，计算机按特定顺序处理这些命令。其中的命令称为语句。

如果你的家里雇佣了一名管家，而且你拥有高度紧张的 A 型人格，你需要为管家制定一组详细的命令供其遵循，如下所示。

亲爱的 Jeeves 先生，在我外出向国会请求紧急援助之时，请完成下述工作：

1. 用吸尘器打扫房间；
2. 去商场；
3. 买些酱油、芥末，并尽可能多买些加州寿司卷；
4. 回家。

谢谢！

Bertie Wooster

如果你告诉管家做什么，那么他就会灵活地完成你安排的任务：如果加州寿司卷卖没了，你的管家会给你买波士顿卷。

但是计算机的处理方式不会这么灵活。它会按照你编写的程序严格执行，一次执行一个语句。

下面是一个采用 BASIC 语言编写的计算机程序示例，相当简单。现在我们来看一下这个

程序，请读者不要纠结于该程序每一行所代表的意思。

```
1 PRINT "Shall we play a game?"
2 INPUT A$
```

将该程序翻译为自然语言，就相当于让计算机来执行如下事情：

亲爱的个人计算机，

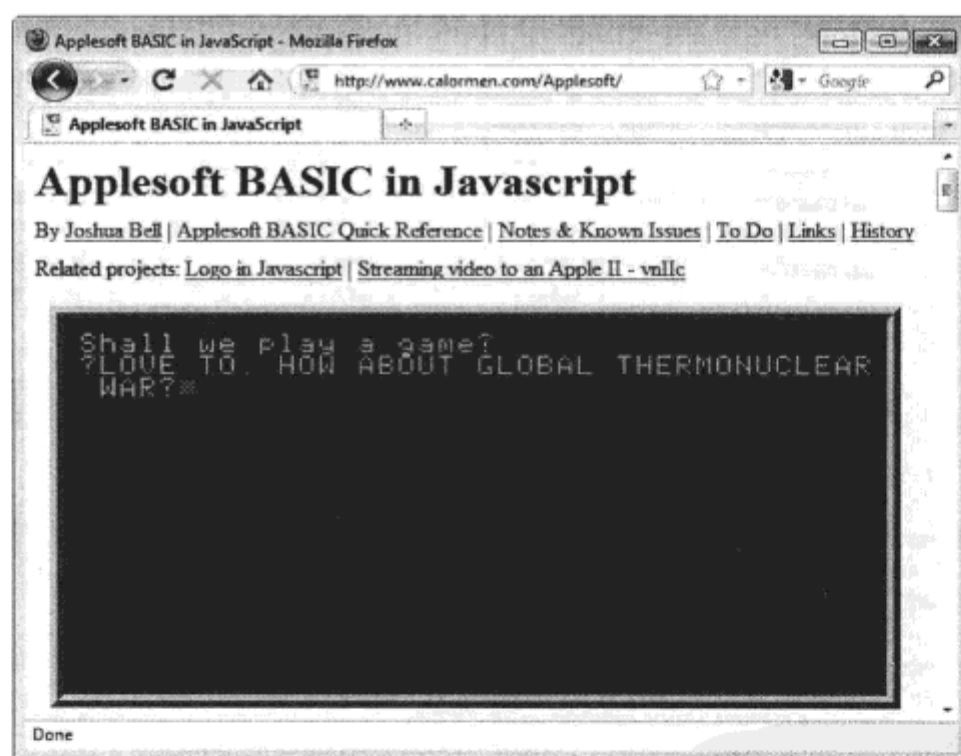
1. 请显示问题 “Shall we play a game?”
2. 给用户一个回答该问题的机会。

爱你的 Snookie Lumps

计算机程序中的每一行都是一条语句。计算机按特定顺序处理程序中的每条语句，就像厨师按菜谱炒菜，或管家 Jeeves 先生按照 Bertie Wooster 的指示行事那样。在 BASIC 语言中，行号用于标识语句的顺序。其他语言（如 Java）不使用行号，而是使用不同的方式告诉计算机如何运行程序。

图 1.1 是一个 BASIC 程序示例，该程序运行的是 Joshua Bell 的 Applesoft BASIC 解释器。该解释器在 Web 浏览器中运行，读者可在 www.calormen.com/Applesoft 找到它。

图 1.1
一个 BASIC 程序
示例



鉴于程序的运行方式，如果程序运行时发生错误，不能责怪计算机。毕竟，计算机只是严格按照你的指令去做。因此程序的运行错误应该归咎于程序员。这真是一个坏消息。

当然，好消息是，对编程人员和计算机而言，都不会造成永久伤害。出版本书的过程中没有人受到伤害；你在学习使用 Java 编程时，也不会损坏计算机。

By the Way

注意：

“Shall we play a game (玩个游戏怎么样)” 摘自 1983 年的电影 War Games。在该影片中，Matthew Broderick 扮演的年轻计算机程序员在全球几乎爆发热核战争时挽救了人类。

1.3 程序的工作原理

大多数计算机程序的编写方式都相同，就像写信一样：在文本编辑器中输入每条语句。有些编程工具自带了编辑器，而有些编程工具则可以与任何文本编辑软件一起使用。

编写好计算机程序后，像保存其他文档一样将其存盘。计算机程序通常有它们自己的扩展名，用于指出所属的文件类型。Java 程序的扩展名为.java，如 Calculator.java。

为了运行保存为文件的程序，你需要某些帮助。所需要的帮助类型取决于你使用的编程语言。有些语言需要解释器来运行程序。解释器是对每条计算机程序的每一行语句进行解释，告诉计算机做什么的程序。大多数版本的 BASIC 都是解释型语言。解释型语言的优点是可以快速进行测试。使用 BASIC 语言编写程序时，可以立即进行测试，找出错误并修正，然后再试。解释型语言的主要缺点是运行速度比其他程序慢。

其他编程语言需要编译器。编译器将计算机程序翻译成计算机能够理解的格式；它还尽可能地对程序进行优化，使其高效运行。编译后的程序不需要解释器就可直接运行，且运行速度比解释型程序快，但测试起来需要更多的时间。在测试前，需要编写并编译程序。发现错误并修复后，必须重新编译，以确定错误是否已消除。

Java 的独特之处在于，它同时需要编译器和解释器，后面编写 Java 程序时，读者将更详细地了解这一点。

注意：

如果使用的文本编辑器是一款字处理程序，能够支持粗体、字号或其他格式化功能，则在编写计算机程序时不要使用这些功能。程序应该是没有任何特殊格式的文本文件。Windows 操作系统自带的字处理程序 Notepad 将所有文件都保存为无格式的文本文件。Linux 系统下的 vi 编辑器也可用于创建无格式的文本文件。

By the Way

1.4 为什么程序不能正常工作

很多编程新手在测试程序时感到气馁，因为到处都是错误。有些是语法错误，表明计算机不能识别程序中的语句；有些是逻辑错误，这种错误仅当测试程序时才能被程序员发现（它们也有可能被忽略掉）。逻辑错误不能被计算机识别，但导致计算机执行意想不到的操作。

开始写程序后，读者将经常遇到错误，这是必然的。编程错误被称为 bug，该术语起源于 100 年前甚至更早，那时指的是技术设备故障。修复错误的过程被称为调试（debugging）。现在有多种方法可以用来描述编程错误，而且这并不是一个巧合。不论是否愿意，在学习编写计算机程序的过程中，都将获得丰富的调试经验。

1.5 选择 Java 编程工具

在开始编写 Java 程序前，首先需要安装 Java 编程软件。当前有多种 Java 编程软件，其

中包括 Java Development Kit、Eclipse、IntelliJ IDEA 和 NetBeans。每当 Oracle 发布 Java 的新版本时，对其提供支持的第一款工具就是 Java Development Kit (JDK)。

为了创建本书中的程序，读者必须使用 JDK 版本 7，或者是可以与它协同工作的其他编程工具。JDK 是一组用来创建 Java 软件的免费的命令行工具。JDK 不具备图形用户界面，如果你之前没有在诸如 DOS 或 Linux 这样的非图形界面环境中工作过，那么当你使用 JDK 时，将会感到束手无策。

Oracle 提供了另外一种免费的工具——集成了开发环境的 NetBeans，而且用它来编写 Java 代码会更加容易。NetBeans 提供了图形用户界面、源代码编辑器、用户界面设计器，以及项目管理系统等功能。它可以与运行在后台的 JDK 形成互补，因此在开始编写 Java 程序时，必须同时安装了这两种工具。

本书中的程序是用 NetBeans 编写的，用户可以下载 NetBeans，并与 JDK 一起安装。当然，用户也可以使用其他 Java 工具，只要他支持 JDK 7。

By the Way

注意：

Oracle 以 Web 页面的形式提供了详尽的 Java 语言文档。在阅读本书时，你没有必要查看该文档，因为本书对每一个主题都进行了详细的讨论。当时当编写程序时，这些页面将会派的上用场。

你可以将整个文档下载下来，当然也可以在需要时，在 Oracle 的网站上在线阅读，这无疑更为方便。用户可以从 <http://download.oracle.com/javase/7/docs/api> 下载最新的 Java 文档。

1.6 安装 Java 开发工具

本书每章都以一个 Java 编程项目收尾，你可以通过该项目来加深对相关主题知识的理解。

如果你的计算机上没有安装 Java 编程工具，也就不能进行 Java 编程。

如果你有诸如 NetBeans 或 JDK 这样的开发工具，你可以用它来开发本书后面 23 章中的示例程序。然而，读者应熟悉如何使用这些工具，因为同时学习如何使用 Java 和复杂的开发工具可能令人畏惧。

如果你还没有 Java 开发工具，则可以考虑使用 NetBeans 7，用户可以从 Oracle 的站点 www.netbeans.org 上免费下载该工具。

有关下载和安装 NetBeans 的方法，请阅读附录 A。

1.7 总结

本章介绍了有关计算机编程的概念：提供一组指令，告诉计算机做什么。读者也可以下载并安装 Java 开发工具，以用来编写本书所有的示例程序。

如果读者仍不清楚程序、编程语言或 Java 等概念，也不要气馁。下一章将详细介绍 Java 程序的创建过程，届时，读者将会明确上述这些概念。

1.8 问与答

问：BASIC、C++、Smalltalk、Java 等语言的名字是什么意思？

答：BASIC 是 Beginner's All Symbolic Instruction Code 的首字母的缩写。C++ 是为了改进 C 语言而创建的编程语言，而 C 语言是 B 编程语言的改进版。Smalltalk 是一种革命性的面向对象的编程语言，开发于 20 世纪 70 年代，而且它的很多理念都被 Java 所采用。

Java 没有采用传统语言的命名方式：使用首字母缩写或其他有意义的名称。Java 是 Java 开发者最喜欢的名称，它击败 WebRunner、Silk、Rube 等名称脱颖而出。当我创建属于我的第一个编程语言时，会将它命名为 Salsa——因为每个人都喜欢它嘛！

问：为什么解释型语言的速度比编译型语言慢？

答：原因与现场翻译的速度比翻译书面演讲稿慢相同。现场翻译时，翻译人员需要思考刚刚讲过的每句话，然而其他翻译人员可将演讲作为一个整体，并采取优化的方式以提高翻译速度。编译型语言比解释型语言快得多，是因为可以采取提高程序的运行效率。

1.9 测验

通过回答下列问题来检测对本章介绍的知识的掌握程度。

1.9.1 问题

1. 人们认为计算机编程很难，下面哪个不是其原因？
 - a. 程序员散布该谣言，以提升就业前景。
 - b. 到处充斥行话和首字母缩写。
 - c. 认为编程很难学习的人可以有资格获得政府的援助。
2. 下面哪种工具以每次一行的方式运行计算机程序？
 - a. 缓慢的工具。
 - b. 解释器。
 - c. 编译器。
3. James Gosling 为什么要发明 Java？
 - a. 他对在一个项目中使用的语言不满意。
 - b. 他的摇滚乐队当时没有任何表演。
 - c. 如果在工作期间不能浏览 YouTube，Internet 将很单调。

1.9.2 答案

1. c. 计算机图书的作者也没有得到政府的援助。

2. b. 编译器事先理解指令，所以程序运行起来更快。
3. a. 他对 C++感到失望。

1.10 练习

如果想更多地了解关于 Java 和计算机编程的信息，可以：

- 访问 Oracle 的 Java 站点，网址为 www.oracle.com/technetwork/topics/newtojava，阅读该公司对 Java 技术的介绍；
- 使用普通语言而不是编程语言编写一组指令，将用户选择的数字加 10，再将结果乘以 5。将这些指令分为多个短句子。

要获悉每章末尾的习题答案，请访问本书的配套网址：www.java24hours.com。



第 2 章

编写第一个程序

本章介绍如下内容：

- 在文本编辑器中输入一个程序；
- 使用 `class` 语句对 Java 程序进行命名；
- 使用括号组织程序；
- 将信息存储到变量中；
- 显示存储在变量中的信息；
- 保存、编译和运行程序；
- 修复错误。

第 1 章已经提到，计算机程序是一组告诉计算机做什么的指令。这些指令使用编程语言输入到计算机中。

在本章，读者将通过将指令输入到文本编辑器的方式来创建第一个 Java 程序。输入完毕之后，可以保存、编译并测试该程序。

2.1 编写程序所需的工具

第 1 章讲到，要创建 Java 程序，你必须要有支持 Java Development Kit (JDK) 的开发工具，比如 NetBeans 集成开发环境 (IDE)。你还需要可以编译和运行 Java 程序的工具，以及用来编写 Java 程序的文本编辑器。

对于大多数编程语言来说，计算机程序都是通过在文本编辑器（也称为源代码编辑器）中输入文本的方式来编写的。有些编程语言自带了文本编辑器。Oracle 的开发工具 NetBeans 就包含了用来编写 Java 程序的编辑器。

Java 程序是简单的文本文件，没有诸如文本居中、粗体等其他特别格式。NetBeans 源代

码编辑器很像一个功能增强了的简单文本编辑器。彩色文本用来识别用户输入的不同元素。NetBeans 本身也带有适当的缩进格式，以及有用的编程文档。

由于 Java 程序是文本文件，因此可以使用任何文本编辑器打开 Java 程序，并对其进行编辑。你也可以使用 NetBeans 编写 Java 程序，然后在 Windows Notepad（记事本）中打开它，并做出相应的修改，然后再在 NetBeans 中打开该程序，这不会造成任何问题。

2.2 创建 Saluton 程序

读者将创建的第一个 Java 程序显示计算机科学界的传统问候语 “Saluton mondo!”。

在 NetBeans 中输入第一个程序之前，先采取如下步骤来创建一个称为 Java24 的新项目。

1. 选择菜单命令 File->New Project。
2. 选择项目分类 “Java” 和项目类型 “Java Application”，然后单击 Next 按钮。
3. 输入 “Java24” 作为该项目的名称。如果之前已经创建了该项目，则会看到错误信息 “Project folder already exists and is not empty”。
4. 取消选中 “Create Main Class” 复选框。
5. 单击 Finish 按钮。

Java24 项目在其文件夹中建立起来。在阅读本书的过程中，你可以为编写的所有 Java 程序使用这个项目。

2.3 开始输入程序

NetBeans 将所有相关的程序分组列入到一个项目中。如果你还没有打开 Java24 项目，可采用如下方式打开。

- 选择菜单命令 File->Open Project。
- 找到并选择 NetBeansProjects 文件夹（如果有必要）。
- 选择 Java24，并单击 Open Project 按钮。

Java24 项目出现在项目（Projects）面板中。

为了添加一个新的 Java 程序到当前的项目中，选择 File->New File，打开 New File Wizard 对话框，如图 2.1 所示。

分类（Categories）面板会列出用户可以创建的所有 Java 程序的类型。在该面板中单击 “Java” 文件夹来查看属于该分类的文件类型。对本项目而言，选择 “Empty Java File” 类型，然后单击 Next 按钮。

在 Class Name 字段，输入 “Saluton”，然后单击 Finish 按钮来创建新的 Java 程序。名为 Saluton.java 的空文件将在源代码编辑器中打开。

在源代码编辑器中，通过输入程序清单 2.1 中的每一行语句，由此开启你的 Java 编程生涯。这些语句称为程序的源代码。

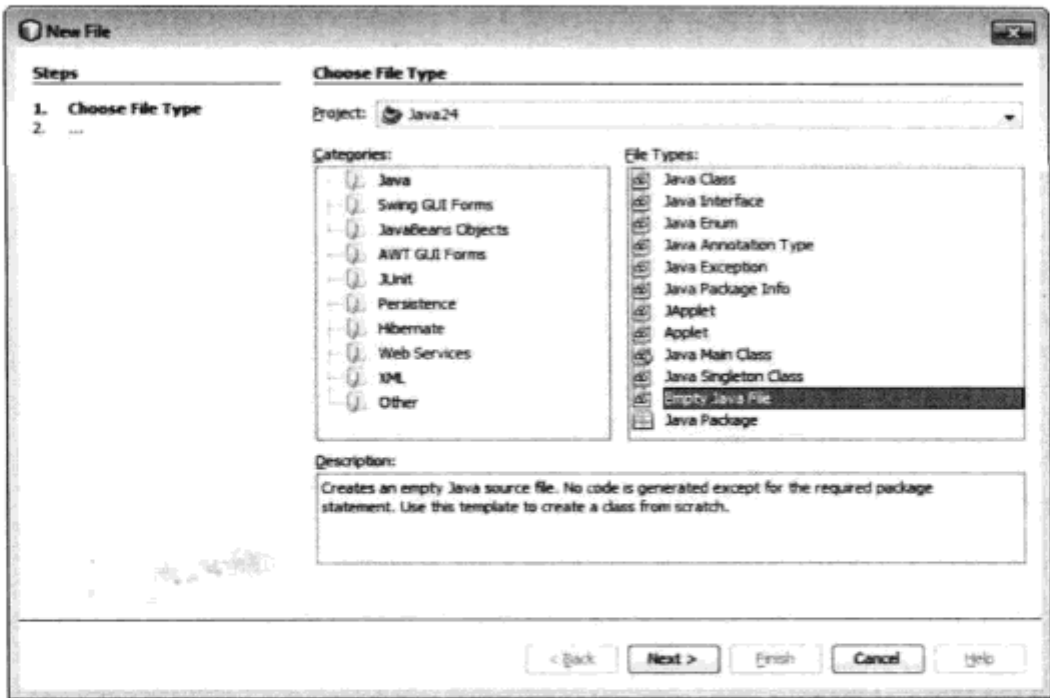


图 2.1
New File Wizard
对话框

警告：
不要输入每行前面的行号和冒号，它们在本书中的目的是方便引用代码。

**Watch
Out!**

程序清单 2.1 Saluton 程序

```
1: public class Saluton {
2:     public static void main(String[] arguments) {
3:         // My first Java program goes here
4:     }
5: }
```

确保大小写与该程序清单中一致，并使用空格键或 Tab 键在第 2~第 4 行前面插入空白。当输入完后，选择 File->Save，然后单击 Save All Files 按钮来保存文件。

当前，Saluton.java 只包含 Java 程序的架构。读者可以创建多个开头与此相同的程序，当然，第 1 行的单词 Saluton 除外。该单词表示程序的名称，而且每个程序都不同。第 3 行是一个英语句子。其他内容对你来说是全新的。

2.3.1 class 语句

程序的第 1 行如下：

```
class Saluton {
```

把这句话翻译成自然语言的意思是：计算机，请将我的 Java 程序命名为 Saluton。

第 1 章讲到，输入到计算机的每条指令都被称为语句。class 语句让你能够给计算机程序指定名称，它还可以确定程序的其他方面，这将在后面介绍。术语 class 的重要性在于 Java 程序也叫 class（类）。

在这个例子中，程序名 Saluton 与文档名 Saluton.java 匹配。Java 程序的名称应与其文件名的第 1 部分相同，而且大小写完全一致。

如果程序名与文件名不匹配，编译有些 Java 程序时可能会出错，而是否出错则取决于如何使用 class 语句来配置程序。

2.3.2 main 语句的作用

该程序的下一行如下：

```
public static void main(String[] arguments) {
```

这行语句告诉计算机：程序的主要部分从这里开始。Java 程序被组织成多个部分，所以需要有种方法指出首先处理哪部分。

大多数 Java 程序的入口都是 main 语句，但 applets 和 servlets 例外，其中，前者是作为网页的一部分而运行的程序，后者是由 Web 服务器运行的程序。接下来的几章中，你编写的大多数程序都将 main 作为起点。

2.3.3 大括号

在 Saluton 程序中，除第 3 行之外，每行都包含一个大括号，要么是“{”，要么是“}”。这些大括号用于将程序中的语句分组（与小括号类似，小括号是用来将句子中的单词分组）。在左大括号“{”和右大括号“}”之间的内容属于同一组。

这些语句组称为块。在程序清单 2.1 中，第 1 行的左大括号“{”与第 5 行的右大括号“}”对应，它们将整个程序作为一个块。可以用这种方式来指示程序的开始和结尾。

块可以包含其他块（就像小括号可以这样使用“((……))”一样）。在 Saluton 程序中，第 2 行和第 4 行大括号指定了另一个块，这个块以 main 语句打头。程序运行时，计算机将把 main 语句块中的内容作为命令进行处理。

下面的语句是该语句块中唯一的内容：

```
// My first Java program goes here
```

该行这是占位行，行首的//告诉计算机忽略本行，在程序中放置它的目的在于方便人们阅读程序的源代码。用于该目的的行被称为注释。

By the Way

注意：

NetBeans 可以显示块的开始位置和结束位置。在 Saluton 程序的源代码中单击其中一个大括号，这个大括号以及与之对应的大括号将显示为黄色。封装在这些黄色大括号中的 Java 语句组成了一个块。

至此，你已经编写完了一个完整的 Java 程序。你可以编译它，但是运行它时什么也不会发生。因为你没有告诉计算机要做什么。main 语句块只包含了一行注释，而它将被忽略。要让 Saluton 程序问候用户，必须在 main 语句块的左大括号和右大括号之间添加一些命令。

2.4 在变量中存储信息

在编写的程序中，你需要将信息临时存储在某个地方，为此可以使用变量。变量是可存储整数、浮点数、真假值、字符及文本行等信息的地方。在变量中存储的信息可以修改，“变量”因此而得名。

在 `Saluton.java` 文件中，用如下语句来替换第 3 行：

```
String greeting = "Saluton mondo!";
```

这条语句告诉计算机：将文本行“`Saluton mondo!`”存储到变量 `greeting` 中。

在 Java 程序中，必须告诉计算机变量存储的信息类型。在这个程序中，`greeting` 变量是一个字符串——可以包含字母、数字、标点符号及其他字符的文本行。在该变量前面添加 `String` 之后。就可以用来存储字符串值。

在程序中输入这条语句时，必须在行尾输入分号（`;`），在 Java 程序中，每条语句都要以分号结束。这就像句尾的句点，计算机使用它们来判断当前语句的结束位置以及下一条语句的开始位置。

对我们人类来说，每一行只输入一个语句的方式，可以使程序更具有可理解性。

显示变量的值

如果此时运行程序，将不会显示任何内容。在 `greeting` 变量中存储文本行的命令在幕后运行。要让计算机显示它在做什么，则需要显示该变量的内容。

在 `Saluton` 程序中，在语句 `String greeting = "Saluton mondo!";` 的后面插入一个空行，然后输入下面的语句：

```
System.out.println(greeting);
```

这条语句告诉计算机显示存储在 `greeting` 变量中的值。语句 `System.out.println` 告诉计算机在系统输出设备上（也就是你的显示器）显示一行。

2.5 保存编写好的程序

程序现在应如程序清单 2.2 所示，当然，可能第 3 行和第 4 行使用的空格略有不同。进行必要的改正，然后存储该文件（通过选择 `File->Save` 或 `Save All Files` 按钮来执行）。

程序清单 2.2 `Saluton` 程序的完整版本

```
1: class Saluton {
2:     public static void main(String[] args) {
3:         String greeting = "Saluton mondo!";
4:         System.out.println(greeting);
5:     }
6: }
```

计算机运行该程序时，将运行 `main` 语句块中的第 3 行和第 4 行。用英文而不是 Java 来编写该程序时，将如程序清单 2.3 所示。

程序清单 2.3 Saluton 程序的分解

```
1: The Saluton program begins here:
2:   The main part of the program begins here:
3:   Store the text "Saluton mondo!" in a String variable named
greeting
4:   Display the contents of the variable greeting
5:   The main part of the program ends here.
6: The Saluton program ends here.
```

2.6 将程序编译为 class 文件

在运行 Java 程序之前，必须先编译它。在编译程序时，输入到计算机中的程序指令被转换为计算机可以更容易理解的一种形式。

在 NetBeans 中，程序在保存时会自动进行编译。如果你输入程序清单 2.2 中的指令，则该程序将顺利通过编译。

该程序在编译之后生成一个新的文件，名为 `Saluton.class`。所有的 Java 程序都将编译为类（class）文件，且其文件后缀名为 `.class`。Java 程序也可以由协同工作的多个类文件组成，但是对于 `Saluton` 这样简单的程序来说，只需要一个类文件。

By the Way

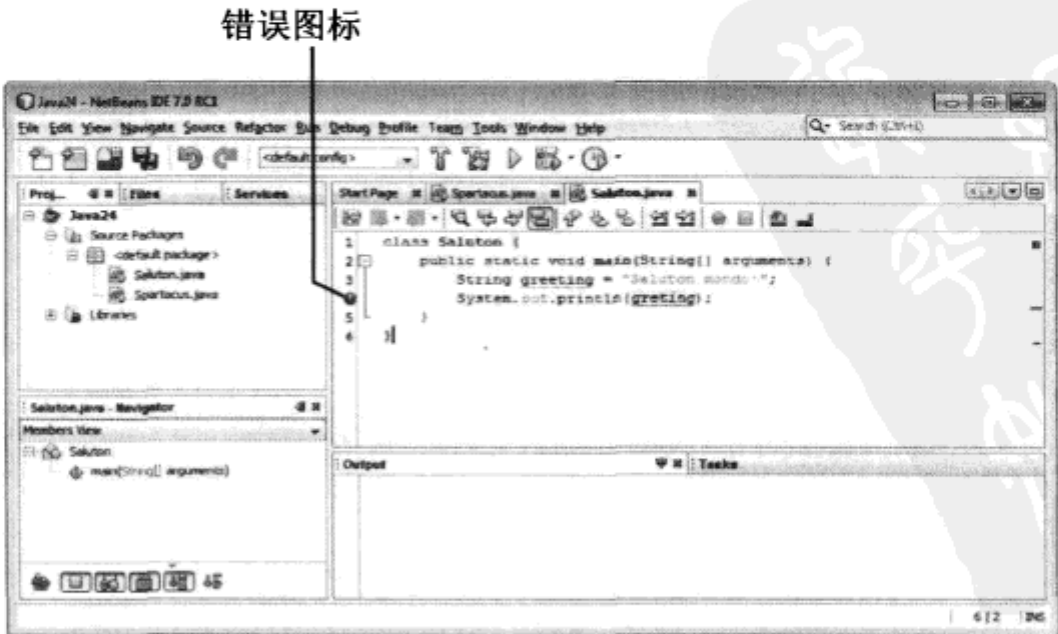
注意：

只有 Java 程序在编译出现错误时，Java 编译器才会有提示。如果你成功地编译完了一个程序，期间没有任何错误，则 Java 编译器不会有任何“动静”。这有点虎头蛇尾。当我在刚开始学习 Java 编程时，我曾经希望在编译取得成功时，能够有所提示。

2.7 修复错误

当在 NetBeans 源代码编辑器中编译程序时，如果出现错误，在编辑面板中对应行的左边会出现一个红色警示图标，如图 2.2 所示。

图 2.2
在源代码编辑器中
定位错误



这个图标显示在触发错误的那一行代码上。可以单击该图标来显示该错误信息，该错误信息会用如下细节来解释这个编译器错误。

- Java 程序的名字。
- 错误的类型。
- 出错的行号。

比如，在编译 Saluton 程序时，可能会看到如下错误信息。

```
cannot find symbol.
symbol   : variable greting
location: class Saluton
```

提示：

本书官方站点 www.java24hours.com 包含了本书所有程序的源文件。如果你能确定 Saluton 程序的错误不是由输入原因造成的，同时又找不到其他原因，则可以去该书的官方站点下载 Saluton.java 程序，然后再运行该程序。

**Did you
Know?**

这里的错误消息是第一行“cannot find symbol”。很多编程新手往往会对这些消息感到迷惑。当读者不明白错误消息的意思时，不要为此枉费时间。只要看一下错误所在的行数，然后寻找最明显的原因即可。

例如，你能确定下面的语句哪里出错了么？

```
System.out.println(greting);
```

这里的错误是变量名输入错误，即变量名应该是 `greeting`，而不是 `greting`。

在创建 Saluton 程序时，如果提示有错误消息，可以检查你的程序是否与程序清单 2.2 中的相同，然后再修改你找到的不同之处。要保证每一个字母的大小写都是正确的，而且也包含了所有的标点符号（比如 {、} 和 ;）。

通常情况下，仔细检查错误消息指示的那一行，足以发现需要修复的所有错误。

2.8 运行 Java 程序

要查看 Saluton 程序的结果是否如你所愿，可使用 Java 虚拟机 (JVM) 运行类文件，JVM 就是运行所有 Java 代码的解释器。在 NetBeans 中，选择菜单命令 `Run->Run File`。在源代码编辑器的下面将会打开输出面板（窗口）。如果没有错误，则该程序会在该面板中显示输出结果，如图 2.3 所示。

如果你看到了在 Java 程序中输入的文本“Saluton Mondo!”，那么你的计算机也就对世界发出了问候——这是计算机编程领域的一个传统，这对于很多程序员来说就像咖啡、短袖衬衫和 Call of Duty（使命召唤）一样重要。

图 2.3

运行第一个 Java 程序



By the Way

注意:

读者可能会问“为什么 Saluton mondo! 是一句传统的问候语言呢”。该短语在世界语中的意思是“Hello world!”。而世界语是由 Ludwig Zamenhof 在 1887 年创建的一种人工语言，用于国际间的交流。我在这里使用世界语，旨在希望更多的计算机人员学习它。

2.9 总结

在本章，读者使用 NetBeans IDE 创建了第一个 Java 程序，并学会了开发 Java 程序所需要的 3 个基本步骤。

1. 使用文本编辑器编写程序。
2. 将程序汇编为类文件。
3. 让 JVM 运行该类文件。

与此同时，读者还学习了一些基本的计算机编程概念，如编译器、解释器、块、语句和变量。随着后续章节的学习，读者将会对这些概念越来越清晰。只要读者在本章成功地运行了 Saluton 程序，就可以进入下一章。

2.10 问与答

问：在 Java 程序的每行中插入适当数量的空格有多重要？

答：这完全不重要。空格无疑会让阅读计算机程序的人受益，但 Java 编译器对空格的数量并不关心。在编写 Saluton 程序时，你也可以不使用空格或 Tab 键进行缩进，而且它也能成功编译。

虽然每行开头的空格数不重要，但在 Java 程序中应采用一致的缩进方式。原因是空格有助于查看程序的组织结构以及语句所属的程序块。

问：Java 程序被称为一个类和一组类。哪种说法是正确的？

答：两者都正确。在接下来几章中，你创建的简单 Java 程序将被编译为扩展名为.class 的单个文件，你可以使用 JVM 来运行它们。Java 程序也可以由一组协同工作的类组成。该主题将在第 10 章详细介绍。

问：既然每条语句都必须以分号结尾，为何注释行 “//My first Java program goes here” 不需要以分号结尾？

答：编译器会完全忽略掉注释行。如果在程序中加入 “//”，则是告诉 Java 编译器忽略该行中 “//” 右边的所有内容。下面的例子演示了如何在语句所在的行添加注释：

```
System.out.println(greeting); // hello, world!
```

问：我在编译器指出有错误的行中没有发现任何错误，我应该怎么办？

答：错误消息中显示的行号并不总是程序中发生错误的地方。检查错误消息指出的行号上面的行，看看能否找到拼写错误或其他 bug。错误通常位于相同的程序块中。

2.11 测验

通过回答下列问题检测对本章介绍的知识的掌握程度。

2.11.1 问题

1. 编译 Java 程序时，你实际上做了什么工作？
 - a. 将其存盘。
 - b. 将其转换为计算机能够理解的格式。
 - c. 将其加入到程序集合中。
2. 什么是变量？
 - a. 是摇摆但是不下降的东西。
 - b. 程序中被编译器忽略的文本。
 - c. 程序中用来存储信息的地方。
3. 修复错误的过程叫什么？
 - a. 解冻。
 - b. 调试。
 - c. 分解。

2.11.2 答案

1. b. 编译是将.java 文件转换为一个或一组.class 文件。
2. c. 变量是用来存储信息的地方，后面将介绍其他存储信息的地方，如数组和常量。

钟摆会摇摆但不会掉下来；注释是编译程序中被编译器忽略的文本。

3. b. 由于在计算机程序中的错误叫 **bug**，修复这些错误就叫调试 (**debugging**)。有些编程工具自带了名为调试器 (**debugger**) 的工具，可帮助修复错误。

2.12 练习

如果想更多地探索本章介绍的主题，可完成下列练习：

- 将英文短语 **Hello world** 翻译成其他语言，翻译时可以使用 **Yahoo!** 网站的 **Babelfish**，其网址为 <http://babelfish.yahoo.com>。编写一个程序，让计算机用法语、意大利语或葡萄牙语向世界问候。
- 在 **Saluton** 程序中故意添加一两个错误。例如，删除行尾的分号或将某行中的 **println** 改成 **println**。保存文件再编译程序，比较由此产生的错误消息。

要获悉这些练习的答案，请访问本书的配套网站 www.java24hours.com。



第 3 章

Java 之旅

本章介绍如下内容：

- Java 的历史；
- 使用 Java 语言的好处；
- 几个 Java 示例；
- 面向对象编程的解释。

在进一步探索 Java 编程之前，有必要更详细地了解 Java 这门编程语言，以及当今的 Java 程序员都在做些什么。尽管 Java 已经不像最初那样局限于 Web 浏览器编程，但是我们仍然可以找出一些如何将 Java 用于 Web 的有趣示例。

本章将访问一些包含 Java 程序的网站，并讨论该语言的历史和发展过程。

要开始这次 Java 之旅，必须有能够处理 Java 程序的 Web 浏览器。

启动你选择的浏览器，穿上舒适的衣服，准备开始 Java 之旅。因为不需要离开房间，所以你不会体验到观光旅游带来的那种简单的快乐，比如粗心大意的出租车司机、异国他乡的美食、风俗人情等。但是，这也有好的一面：不需要被强迫进行安检、不需要护照，而且也没有蒙特祖玛（注：一款游戏的名称）式的报复。

3.1 第一站：Oracle

Java 之旅的第一站始于 Oracle 公司创建的 www.java.com，Oracle 公司开发了 Java 语言（注：应该是 Sun 公司开发了 Java 语言，鉴于 Sun 公司已经被 Oracle 公司收购，因此说 Oracle 公司开发了 Java 语言也并无不可）。

作为 Web 页面的一部分运行的 Java 程序称作 applet。在网页中放置 applet 就像放置其他元素一样：使用标记标语言 HTML 指定在哪里显示程序、它多大以及运行时程序做什么。Java

还以另外两种方式增强了 Web 页面：用 Java 语言编写的桌面程序（desktop program）可以从 Web 浏览器中启动；Web 服务器运行的 Java servlets 可以分发 Web 应用程序。

Oracle 的 Java 部门负责 Java 语言和相关软件的开发。Java.com 上的“The Java in Action”区域显示了 Java 是如何应用在 Web 站点、Android 手机以及其他平台上的。如今有数以百万计的设备运行的是以 Java 语言编写的程序。图 3.1 显示的是一款使用 Java 语言编写的大型多人在线游戏 RuneScape。通过使用任何 Web 浏览器来访问 www.runescape.com 网站，即可免费玩这款游戏。

图 3.1

由 Java 语言编写的在线游戏 Rune Scape



Java.com 网站为读者提供了一个学习 Java 使用方式的地方。Oracle 还为 Java 开发人员提供了一个更为技术性的 Web 站点，其网址为 <http://www.oracle.com/technetwork/java>。读者可以在该站点上找到最新版本的 NetBeans、Java Development Kit，以及其他编程资源。

Java 历史简要回顾

当 Sun 公司创造了 Java 语言时，当时的公司主管之一 Bill Joy 将该语言称为“15 年努力的成果，它是更好、更可靠的计算机编程语言”。Java 的创建过程可能比他说的还要复杂一些。

Java 是在 1990 年由 Sun 公司的工程师 James Gosling 开发的，它可以作为智能设备（如交互式电视、无所不能的烤箱、具备人工智能的 SkyNet 军用卫星等）的编程语言。Gosling 对其使用 C++编写的程序感到失望，他灵机一动，决定躲在办公室开发一种更适合其需求的新语言。

By the Way

注意：

读者可能已经听说过 Java 是 Just Another Vague Acronym（只是另外一种模糊的缩写）的首字母缩写，或许还听说过 Java 是以 Gosling 最喜欢的咖啡命名的。其实，Java 语言的命名没有任何秘密，也不是出于对咖啡这种饮品的热爱。之所以选择以 Java 来命名，其原因与喜剧演员 Jerry Seinfeld 喜欢说 salsa 这个单词一样：它听起来很酷。

Gosling 最初将这种语言命名为 Oak，灵感来自 Gosling 从办公室的窗户向外看到的一棵

橡树。当时交互式电视已成为一个具有数百万美元产值的行业，该语言是 Sun 公司进入该行业的发展策略的一部分。可到今天这些都没成为现实（如今 TiVo、WebTV 等都在向游戏领域进军），但 Gosling 发明的新语言却发生了重大变化。正当 Sun 公司准备放弃 Oak 开发小组，将其成员分散到公司其他部门时，Web 开始流行起来。

正是阴差阳错，使 Gosling 发明的语言适合家用电器的特性也使其适用于 Web。Gosling 的团队设计了能够让程序在 Web 页面安全运行的方法，而且为了与该语言的新用途相匹配，为之选了一个易于记忆的名字：Java。

虽然 Java 还可以做很多其他的事情，但 Web 为其提供了吸引全球注意力的舞台。当 Java 语言成为主流语言时，如果你没听过 Java，那一定是离群索居或在从事长期的轨道任务。

Java 语言总共有如下 8 个主要版本。

- 1995 年秋发布的 Java 1.0：最初的 Java 版本。
- 1997 年春发布的 Java 1.1：对 Java 语言的升级，在对图形用户界面的支持上有了很大改进。
- 1998 年夏发布的 Java 2 1.2 版：对 Java 语言进行了巨大的扩展，使其成为通用的编程语言。
- 2000 年秋发布的 Java 2 1.3 版：增强了多媒体功能。
- 2002 年春发布的 Java 2 1.4 版：一次重大升级，增强了对 Internet、XML 功能和文本处理的支持。
- 2004 年春发布的 Java 2 第 5 版：提供了更强的可靠性和自动数据转换功能。
- 2006 年冬发布的 Java 6：对内置的数据库和 Web 服务的支持进行了升级。
- 2011 年夏发布的 Java 7：Java 的最新版本，对核心语言、内存管理进行了改进，同时增加了 Nimbus 图形用户界面。

3.2 去 Java 学校

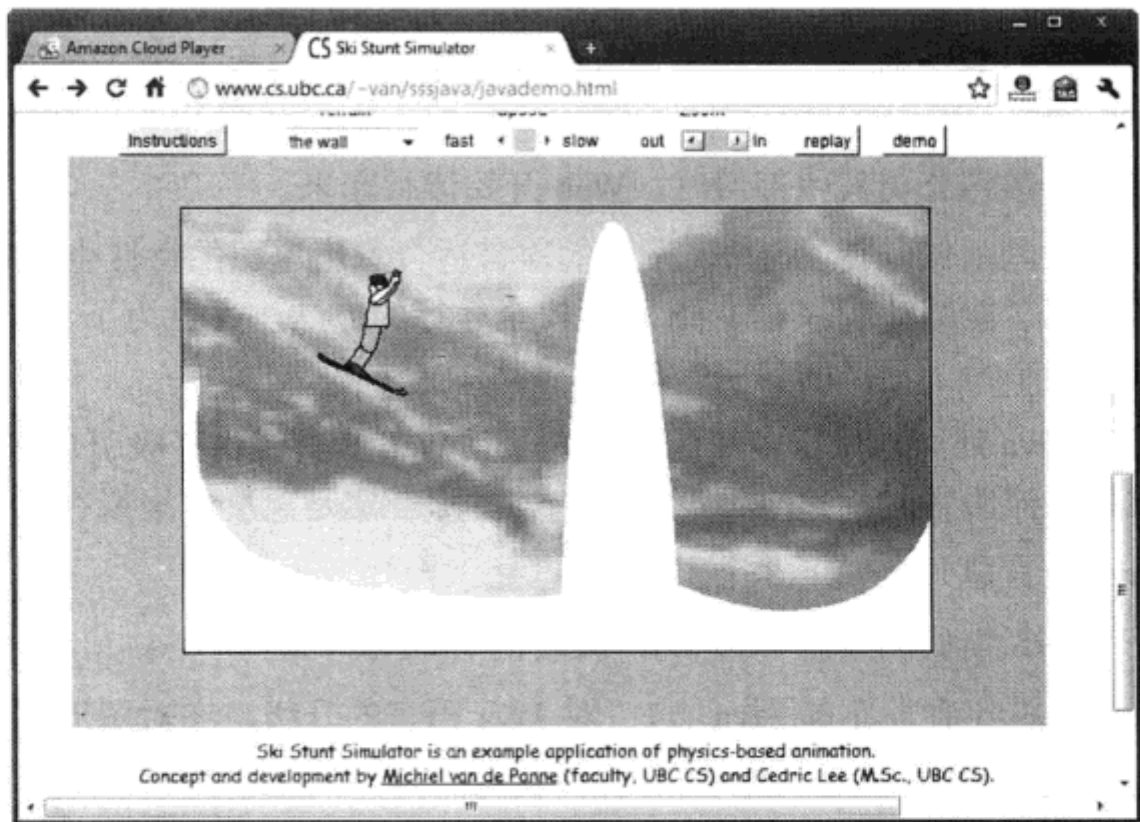
Web 为教育工作者和学生提供了众多的资源。由于与标准 Web 页面相比，Java 程序能够提供更多的交互体验，因此有些程序员使用 Java 编写用于 Internet 的学习程序。

读者可访问 <http://www.cs.ubc.ca/~van/sssjava> 来看这样的一个示例。通过访问该网站，将会看到由 Michiel van de Panne（英国哥伦比亚大学计算机科学专业的教授）开发的跳台滑雪模拟器。该程序使用 Java 来演示当滑雪人员从不同的斜坡进行跳台滑雪时，其基于物理的动画。通过将鼠标向 8 个方向移动，可以控制滑雪人员的动作，而且这 8 个方向都会对滑雪的成功与否造成影响。图 3.2 显示的是在我的虚拟滑雪人员遇到阻挡之前，一次成功的滑雪尝试。

尽管有大量的教育程序可运行在不同的操作系统上，但使这种程序脱颖而出的是其可用性（availability）。模拟器是直接在 Web 页面上运行的，它和大多数桌面软件不同，不需要专门安装，也不局限于特定的操作系统。只要计算机中安装了 JVM，那么就可以运行 Java 程序。

图 3.2

使用 Java 语言编写的跳台滑雪模拟器可以提供交互体验



浏览器载入的 JVM 与第 2 章中运行 Saluton 程序使用的 JVM 相同。浏览器的 JVM 只能运行 Web 页面中的 Java 程序，而不能处理位于其他地方（比如文件夹）的程序。

第一款支持 Java 的浏览器需要有内置的 JVM。如今，浏览器是否支持 Java 则依赖于是否有 Java 插件。Java 插件是一种对浏览器功能进行增强了的 JVM。

Did you know?

提示：

Oracle 将 Java 插件包含在 JDK 和其他产品中，所以你的计算机中可能已经安装了 Java 插件。要检查你的计算机是否安装了 Java，可以访问 www.java.com 网站，该网站上的“Do I Have Java?”链接可以检测到是否安装了 Java。

诸如跳台滑雪这样的 Java 程序不需要针对特定的计算机系统进行编写，由于类似于 Windows 这样的操作系统也称为平台，因此 Java 的这种优势称之为平台独立性。Java 可以在多种计算机系统中运行。最初，Java 的开发者们之所以认为它必须支持多种平台，是因为要将其用于各种家用电器和其他电子设备。

用户可将使用 Java 语言编写的程序运行在各种计算机系统中，而且不需要做任何额外的工作。在正常情况下，使用 Java 编写程序时，不需要为不同的操作系统和设备创建不同的版本。

3.3 在 JavaWorld 用午餐

通过前面的介绍，在读者对 Java 的兴趣应该逐渐浓厚起来。我们现在可在 JavaWorld 的网站吃午餐。JavaWorld 是一本针对 Java 程序员的在线杂志，其网址为 www.javaworld.com。

JavaWorld 提供了“如何 (how-to)”文章、与 Java 开发热点领域相关的新闻以及其他定期专题。以 Web 格式出版的一个优点是，可以演示与文章相关的 Java 程序。图 3.3 所示为一个“诗歌磁板”Java 程序，是在描写如何创建该程序的文章中提供的。

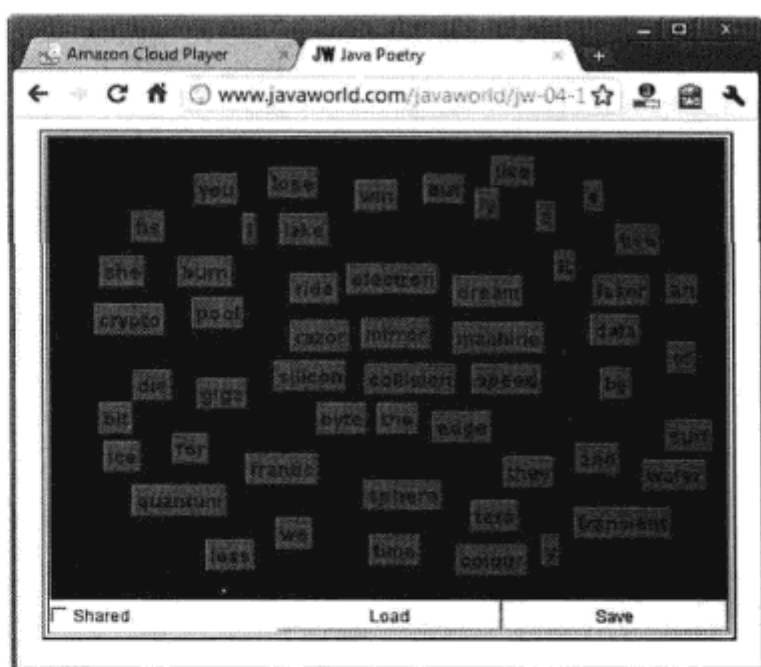


图 3.3

JavaWorld 中一篇介绍如何创建“诗歌磁板”的文章包含程序示例

注意：

JavaWorld 上的内容有时会发生移动，但是在本书写作之时，可通过 www.cadenhead.org/poetry 网站直接访问这篇“诗歌磁板”文章。如果该页面不可用，则可使用站点的搜索引擎来查找关键词“poetry”。

By the Way

JavaWorld 发表了与 Java 语言及其发展相关的文章和评论。自 Java 语言面世以来，一个激烈争论的问题是该语言是否安全。

由于 Java 程序在网页中的运行方式，安全性至关重要。读者在本章尝试的 Java 程序将下载到你的计算机中。当程序下载完毕后，将在计算机上运行。

除非你认识很多人，否则你访问的大部分 Web 页面都是由陌生人发布的。从安全角度看，运行这些人的程序无异于将你的计算机借给别人使用。如果 Java 语言不能防范这种滥用行为，Java 程序将把病毒引入到你的计算机系统，删除文件、播放 Justin Bieber 的歌曲，以及执行其他可怕的事情。Java 提供几种不同类型的安全措施，确保运行在 Web 页面中的 Java 程序是安全的。

对通过 Web 运行的 Java 程序进行下列限制来确保安全：

- 任何程序都不能打开、读写或删除用户系统中的文件；
- 任何程序都不能运行用户系统中的其他程序；
- 程序创建的所有窗口都明确标识为 Java 窗口；
- 除其所属的网站，程序不能连接到其他网站；
- 对所有程序进行验证，确保编译后未被修改。

尽管没有任何保证，但是 Java 语言已经证明可以在 Web 页面中安全使用。

Java 还为在浏览器中运行的程序提供了更灵活的安全策略。可以将某些公司和程序员指定为“可信的开发者”，从而在你的浏览器中运行他们的 Java 程序，而不受正常的限制。

这种信任系统是通过使用具有数字签名的 applet（可以明确地识别 Java 程序的作者）建立的。这些签名是与独立认证机构（如 VeriSign）联合创建的。

如果读者曾经授权一个程序在浏览器（比如 IE 或 Google Chrome）中运行，也就相当于建立了信任与身份验证的系统。

尽管 applet 在今天仍然有用武之地，但是这几年发展起来的其他技术，比如 Flash、Silverlight、HTML5 等已经在基于 Web 页面的程序中崭露头角。Java 在移动应用、服务器程序和桌面软件中的应用却越来越常见。

3.4 在 NASA 仰望天穹

Java 之旅的第一天下午将前往 NASA——一家大量使用 Java 的美国政府机构。其中最流行的是 SkyWatch，这是一个帮助天文学家观察轨道卫星的 applet。通过访问 www.cadenhead.org/nasa，你将自动被转接到 NASA 的 SkyWatch 站点，然后即可将该 applet 载入到浏览器中。

SkyWatch 将 8 个不同卫星（也可以自行添加或移除卫星的数量）的当前位置和轨道叠加到世界地图上。图 3.4 中运行的 applet 显示了 SEASAT-1 卫星从牧夫星座到武仙星座的运动轨迹。

图 3.4

i NASA 的 SkyWatch applet 可以监视轨道卫星的位置和路径，这对金属观鸟者来说是一个福音



这个 applet 可以不断重绘出轨道卫星在运行时的位置。这种实时更新是可以实现的，因为 Java 语言是多线程的。多线程是计算机同时执行多项任务的一种方式；程序的一部分负责一项任务，另一部分负责另一项任务，两部分互不影响。在这种情况下，程序的每一部分称为一个线程。

在诸如 SkyWatch 这样的程序中，每一颗卫星都运行自己的线程。如果使用诸如 Windows 7 这样的操作系统，在同一时刻运行多个程序时，使用的就是这种行为。如果你在一个窗口中玩 Desktop Tower Defense 游戏，同时在另外一个窗口中查看公司销售报表，同时还给朋友打长途电话——恭喜你，你是多线程的。

3.5 回归正题

此时，读者可能会有这样的印象：Java 主要在空间爱好者、诗人和滑雪人员中使用。Java 之旅的下一站将展示一个 Java 的常规使用示例。

将 Web 浏览器指向 JTicker 站点，网站为 www.jticker.com。

JTicker 的发布者，也就是 Stock Applets 公司，开发了在其他网站上可以显示商业新闻头条和股票行情的 Java 程序。图 3.5 所示为其滚动股票行情的一个演示版本。

不像其他股票分析程序那样，需要在每个要访问它的雇员的计算机上安装软件，通过使用 Java，Stock Applets 的雇员可以使用 Web 浏览器来访问这些程序。所有雇员所需要做的就是只需访问该公司的网站。

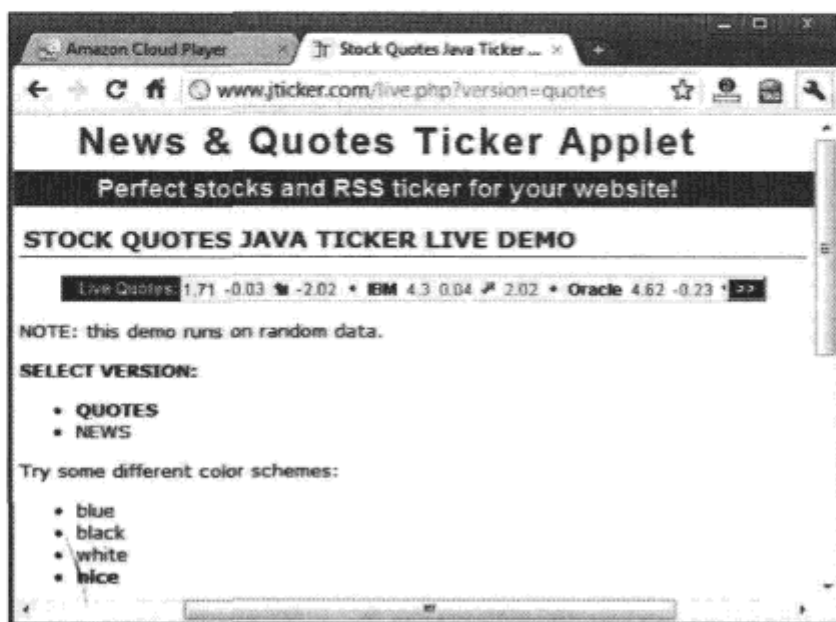


图 3.5

Stock Applets 开发的显示股票行情的 Java 程序

可以从不同的方式看待像这个 applet 的程序。一种方式是认为该程序就像物体——存在于现实中，占据一定的空间，并具备特定的功能。Java 使用的面向对象编程（将在第 10 章介绍）将计算机程序作为一组对象来创建。每个对象处理特定的工作，并且知道如何同其他对象交流。例如，股票行情程序可以有如下对象组成：

- 一个报价对象（quote object），它表示一个单独的股票报价；
- 一个组合对象（portfolio object），它用来存储特定股票的一组报价；
- 一个股票对象（ticker object），它显示一个组合（portfolio）；
- 一个 Internet 对象、一个用户对象和众多其他的对象。

在这个模型下，股票行情软件一个集合，这个集合包含完成工作所需的所有对象。

OOP 是一种功能强大的程序创建方法，让编写的程序更有用。看看股票行情软件，如果程序员想将其报价功能加入到其他软件中，报价对象不需要做任何修改就可以用于新程序中。

3.6 到 Java Boutique 去问路

这次 Java 程序的世界之旅一直都是由精通基于 Web 技术的难点和特点的专家引路，稍后读者将自己探索旅程，所以先停下来，前往对 Java 旅行者来说最好的向导那里看看，这就是 Java Boutique 网站，其网址为 <http://www.developer.com/Java>。

Java Boutique 有齐全的 Java 程序目录以及与 Java 语言相关的编程资源。对程序员来说，本站点的最佳使用方式就是阅读那些提供了源代码的程序。这里所谓的“源代码”就是用来创建计算机程序的文本文件的别称。读者在第 2 章开发的 Saluton.java 文件就是一个源代码示例。

Java Boutique 主页上的 Source Code 链接列出了包含源代码的所有程序。

其中一个源代码可用的程序是 Aleksey Udovychenko 的 Absolute 游戏，这是一个空间视

频游戏，你可以控制你的战舰炸出一条道路，然后穿过小行星带（见图 3.6）。该游戏具有滚动的动画、图形、键盘控制和声音等特色。有关该游戏的详情，请访问 <http://javaboutique.internet.com/Absolute>。

图 3.6

Absolute 这款 Java 程序的源代码可以在 Java Boutique 中找到



By the Way

注意：

Gamelan 的 Java Applet 评级服务（Java Applet Ratings Service, JARS）是 www.jars.com 上的一个基于浏览器的 Java 程序和其他资源的目录，它通常会包含与程序配套的源代码。Java 语言之所以被世界各地数以千计的程序员所使用，部分原因是该语言比较简单。

整个 Absolute 程序的源代码只有 700 行出头。考虑到该程序的功能，代码行数已经非常小了。Java 包含了大量的类库，在编写程序时可以进行调用。Udovydchenko 调用了 Image 类来显示小行星这样的图形，还调用了 AudioClip 类来播放激光枪发射和爆炸的声音。

设计 Java 的目标之一就是要比 C++ 易学，James Gosling 当时在他的智能家电项目中使用的就是 C++。Java 的很大一部分都是基于 C++ 的，因此学习过使用 C++ 语言编程的人学习起 Java 来也不困难。然而，C++ 中有些难以学习和难以正确使用的内容已经从 Java 中删除。

对首次学习编程的人来说，Java 要比 C++ 易学。有些语言是为了让经验丰富的程序员能够在程序中充分利用计算机功能而开发的。

这些语言比较简洁，而且包含编程老手很容易理解的其他特性。

Java 并没有使用这些特性，而是使其成为尽可能简单的面向对象编程语言。创建 Java 语言的目的是易学、易调试和易于使用。Java 还包括了大量增强的特性，从而使得能够与其他语言相抗衡。

3.7 在手机上运行 Java

这次旋风式 Java 之旅的最后一站是 Google Android 手机。运行在 Android 上的每一个单

独的程序都是使用 Java 开发的。这些应用程序扩展了手机的功能，并被称之为 app（应用）。其中最流行的一个 app 是称之为“愤怒的小鸟”的游戏，如图 3.7 所示。



图 3.7

愤怒的小鸟和其他 Android app 是使用 Java 语言开发的

如果你对这款游戏还不了解，或者想深入了解这款游戏，请访问 www.angrybirds.com（最好别去了解这款游戏。否则它会占用你一天、一周，甚至是一个月的时间——这要取决于你多么讨厌游戏中的那些猪！）

之所以将 Android 作为 Java 之旅的最后一站，是因为 Android 成为 Java 语言使用最为广阔的一个领域。在掌握了 Java 语言之后，你可以使用 Android 软件开发包（SDK），来开发自己的 app。Android SDK 是一款可以在 Windows、MacOS 和 Linux 上运行的免费编程套件。

如今大约有 250000 个 app 可以用于 Android 手机和运行移动操作系统的其他设备。第 24 章将会对其详细讲解。

3.8 总结

现在，Java 之旅到此结束了，是时候收起你的行李，并准备开始 Java 编程了。

在接下来的 21 章中，你将会掌握 Java 语言的各种基本组件，学习如何使用面向对象编程的方式来创建对象，以完成任务，还将学习如何设计图形用户界面等内容。

3.9 问与答

问：为什么 Java applet 不再流行了？

答：当 Java 语言于 20 世纪 90 年代中期被发明时，大多数人学习该语言的目的是编写 applet。当时 Java 是创建可在 Web 浏览器上运行的交互式程序的唯一方式。

多年以来，各种技术层出不穷。Macromedia Flash、Microsoft Silverlight，以及新发布的 HTML5 Web 标准都提供了在 Web 页面上运行程序的方式。

由于 applet 在载入到浏览器中时较为费时，而且浏览器开发人员在支持新版本 Java 方面总是慢人一拍，applet 逐渐式微。尽管后面又引入了 Java 插件来在浏览器中运行 Java 的当前版本，但是 Java 在那个时候已经远离了它的初衷，开始演变成为一种复杂的通用编程语言。

3.10 测验

如果此刻你的大脑还没休息，请回答下面的问题以测试对本章内容的理解程度。

3.10.1 问题

1. 面向对象编程因何而得名？
 - a. 程序由一组协同工作的对象组成。
 - b. 由于难于掌握，人们经常反对（object）它。
 - c. 它父母给起的名字。
2. 下面哪项不属于 Java 安全性？
 - a. Web 程序不能运行用户计算机上的程序。
 - b. 总是验证程序作者的身份。
 - c. Java 窗口都标识为 Java 窗口。
3. 程序能同时处理多个任务的功能叫什么？
 - a. 分裂性。
 - b. 多文化性。
 - c. 多线程。

3.10.2 答案

1. a. 它也被缩写为 OOP。
2. b. Java 程序员可以使用数字签名和诸如 VeriSign 等身份验证公司，但这不是必需的。
3. c. 这也被称为多任务，但 Java 使用术语“多线程”，因为独立运行的程序部分被称为线程。

3.11 练习

打开行李前，读者可以通过下面的练习进一步探索本章的主题：

- 通过 Java Boutique 站点 (<http://javaboutique.internet.com>) 了解使用 Java 语言开发了哪些纸牌游戏；
- 访问 Oracle 为 Java 用户开发的网站 www.java.com，然后单击“Do I Have Java?”链接。然后根据指令来查看你的计算机是否安装了 Java。如果有必要的话，可以下载并安装 Java 的最新版本。

要获悉每章末尾的练习答案，请访问本书的配套网站 www.java24hours.com。

第 4 章

理解 Java 程序的工作原理

本章介绍如下内容：

- 应用程序的工作原理；
- 组织应用程序；
- 向应用程序传递参数；
- applet 的工作原理；
- 组织 applet；
- 将 applet 放置到 Web 页面中。

在 Java 编程中需要做出的一个重要决策是，程序将在哪里运行。有些程序将在用户的计算机上运行，其他程序将作为 Web 页面的一部分运行。

在本地计算机上运行的 Java 程序被称为应用程序，而在 Web 页面上运行的程序被称为 applet。本章将介绍为何这种差别很重要。

4.1 创建应用程序

你在第 2 章编写的 Saluton 程序就是一个 Java 程序。接下来，你需要创建另外一个应用程序，用来计算一个数的平方根，然后将该值显示出来。

在 NetBeans 中打开 Java24 项目，开始创建一个新应用程序。

1. 选择 File->New File，打开 New File Wizard。
2. 选择“Java”分类，然后选择“Empty Java File”文件类型，然后单击 Next 按钮。
3. 输入类名“Root”，然后单击 Finish 按钮。

NetBeans 将创建 Root.java 应用程序，并在源代码编辑器中打开一个空文件，以便用户输入代码。输入程序清单 4.1 所示的所有代码，注意不要输入行号以及行号后面的冒号。行号的目

的是方便对程序的某些部分进行描述或引用。输入完毕后，单击工具栏中的 Save All button 按钮保存文件。

程序清单 4.1 Root.java 的完整文本

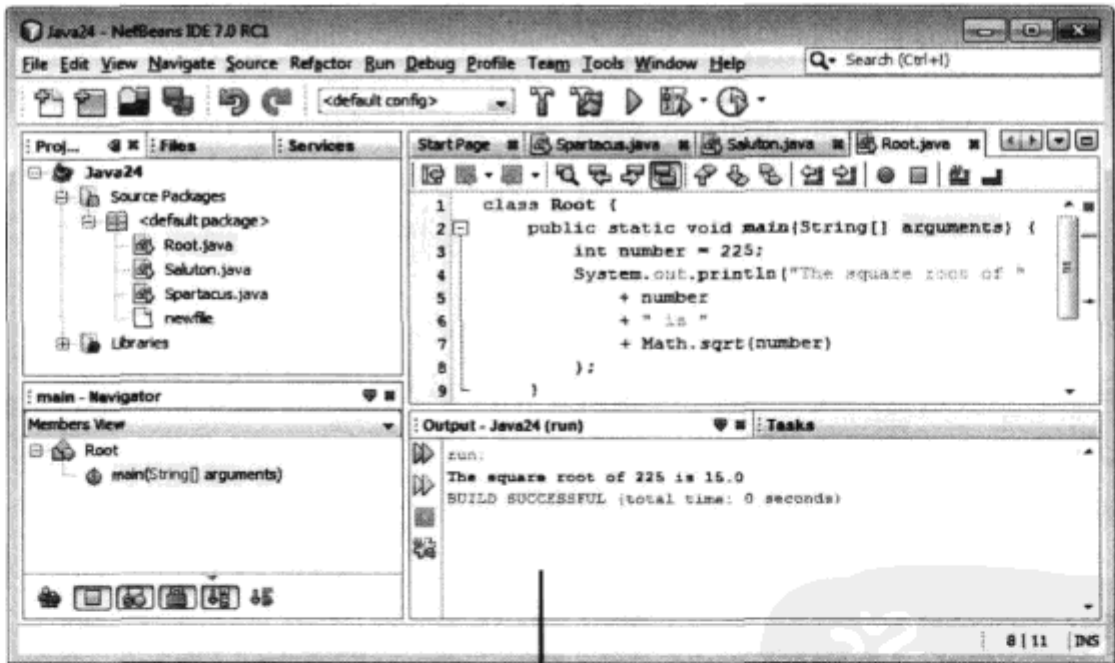
```
1: class Root {
2:     public static void main(String[] arguments) {
3:         int number = 225;
4:         System.out.println("The square root of "
5:             + number
6:             + " is "
7:             + Math.sqrt(number)
8:         );
9:     }
10: }
```

该 Root 程序完成如下任务：

- 第 3 行：在变量 number 中存储整数 225。
- 第 4~8 行：显示该整数及其平方根。第 7 行的 Math.sqrt(number)语句显示平方根。

如果你准确无误地输入了程序清单 4.1 中的代码（包含了每一个标点符号，而且字母大小写没有问题），则可以在 NetBeans 中单击 Run->Run File 菜单命令来运行这个文件。该程序的输出结果将显示在输出面板（窗口）中，如图 4.1 所示。

图 4.1
Root 应用程序的
输出



输出面板

当运行 Java 程序时，JVM 将查找 main()代码块，并开始处理该代码块中的 Java 语句。如果 Java 程序中没有 main()代码块，则 JVM 将提示有错误。

4.2 向应用程序传递参数

你可以使用 java 这个应用程序（会调用 JVM）在命令行运行 Java 程序。当你在 NetBeans 中运行程序时，NetBeans 会在幕后使用这个 java 程序。当以命令方式来运行 Java 程序时，JVM 将装载该应用程序。而且命令可能会包含额外的信息，如下例所示。

```
java TextDisplayer readme.txt /p
```

发送给程序的额外信息称为参数。第一个参数（如果有的话）和应用程序名称之间用一个空格隔开，参数之间也用一个空格隔开。在上面的例子中，参数是 `readme.txt` 和 `/p`。

如果想在参数内部包含一个空格，则必须将该参数使用引号括起来，如下例所示。

```
java TextDisplayer readme.txt /p "Page Title"
```

在该例中，`TextDisplayer` 在运行时，具有 3 个参数：`readme.txt`、`/p` 和 “Page Title”。第 3 个参数中的引号可以防止 `java` 程序将 `Page` 和 `Title` 视为两个独立的参数。

可以向 `Java` 应用程序传递任意数目的参数。然而，要使用这些参数，必须在应用程序中编写处理参数的语句。

为了解参数在应用程序中是如何工作的，在 `Java24` 项目中创建一个新类，步骤如下所示。

1. 选择 `File->New File`。
2. 在 `New File Wizard` 对话框中，分别选择 “`Java`” 分类和 “`Empty Java File`” 文件类型。
3. 将该类命名为 “`BlankFiller`”，然后单击 `Finish` 按钮。

在源代码编辑器中输入程序清单 4.2 中的代码，然后保存。编译该程序，如果出现错误，请进行改正。

程序清单 4.2 `BlankFiller.java` 的完整代码

```
1: class BlankFiller {
2:     public static void main(String[] arguments) {
3:         System.out.println("The " + arguments[0]
4:             + " " + arguments[1] + " fox "
5:             + "jumped over the "
6:             + arguments[2] + " dog."
7:     );
8: }
9: }
```

该程序编译成功，并可以运行。但是如果你使用菜单命令 `Run->Run File` 来运行该程序，将会得到一个看起来很复杂的错误，如下所示。

Output ▼

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at BlankFiller.main(BlankFiller.java:3)
```

之所以会发生该错误，是因为该程序在运行时，期待着接收到 3 个参数。你可以在 `NetBeans` 中通过对项目进行自定义，来指定参数。

1. 选择菜单命令 `Run->Set Project Configuration->Customize`，打开 `Project Properties` 对话框。
2. 在 `Main Class` 文本框中输入 “`BlankFiller`”。
3. 在 `Arguments` 字段中，输入 “`retromingent purple lactose-intolerant`”，然后单击 `OK` 按钮。

由于你自定义了项目，因此在运行该程序时会有所不同。选择菜单命令 `Run->Run Main Project`。应用程序将使用你指定的参数，作为形容词来填充一句话，如下面的输出所示。

Output ▼

The retromingent purple fox jumped over the lactose-intolerant dog.

返回 Project Properties 对话框，然后将自己选择的 3 个形容词指定为参数，而且要确保至少包含 3 个参数。

参数是一种自定义程序行为的简单方法。参数存储在称为数组的变量中。第 9 章将会讲到数组。

在定制程序的行为方面，参数很有用。参数经常用于配置程序，使其以特定的方式运行。Java 将参数存储在数组中，数组是一组相关的变量，这些变量存储的信息类型相同。数组将在第 9 章介绍。

4.3 创建 applet

当 Java 语言在面世之时，它最吸引人的特性就是 applet，即在 Web 页面上运行的 Java 程序。只要 Web 浏览器能够处理 Java 程序，它就可以运行 applet。你可以使用 appletviewer 来测试 applet，appletviewer 是包含在 JDK 中的一个工具，NetBeans 对其提供支持。

applet 的结构与应用程序不同。与应用程序不同，applet 没有 main() 代码块。applet 分为几个代码区域，而且这几个代码区域根据 applet 中所发生的行为被相应的处理。其中两个代码区域是 init() 代码块和 paint() 代码块。init() 是 initialization（初始化）的简写，其作用处理 applet 初次运行时所需要的设置。paint() 用来显示需要显示的内容。

为了查看 Root 应用程序的 applet 版本，新建一个名为 RootApplet.java 的空文件，并输入程序清单 4.3 所示的代码，再将其存盘。

程序清单 4.3 RootApplet.java 程序代码

```
1: import java.awt.*;
2:
3: public class RootApplet extends javax.swing.JApplet {
4:     int number;
5:
6:     public void init() {
7:         number = 225;
8:     }
9:
10:    public void paint(Graphics screen) {
11:        Graphics2D screen2D = (Graphics2D) screen;
12:        screen2D.drawString("The square root of " +
13:            number +
14:            " is " +
15:            Math.sqrt(number), 5, 50);
16:    }
17: }
```

这个程序中的一些语句与 Root 应用程序相同，其主要区别在于组织结构：main() 块被 init() 块和 paint() 块代替。

注意：

本章的示例程序主要用于向读者介绍 Java 程序的组成结构。本章的主要目的是编译程序并看看它们运行的情况。这些程序的其他方面将在随后的章节进行详细介绍。

By the Way

当在 NetBeans 中运行（选择 Run->Run File）这个 applet 程序时，它将被载入到 appletviewer 工具，如图 4.2 所示。

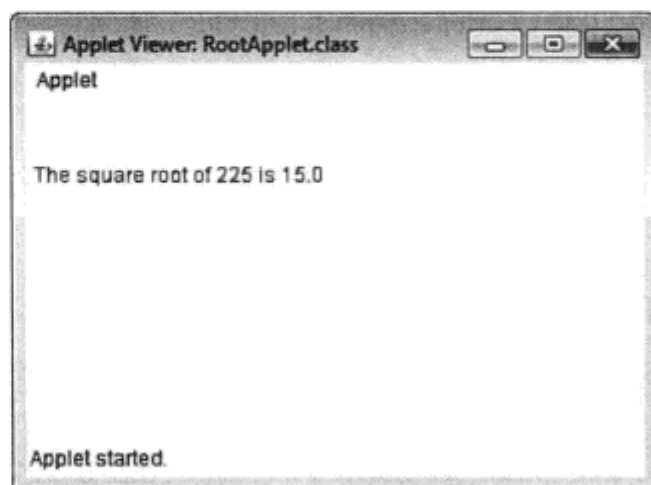


图 4.2

在 appletviewer 中运行的 RootApplet applet

applet 要比应用程序略微复杂，原因是它必须要在 Web 页面上运行，而且要与浏览器中的其他页面元素共存。第 17 章将会对其详细讲解。

appletviewer 工具在测试时非常有用，但是它会给人们留下有关 applet 的错误印象。applet 不能像 Java 程序那样在自己的窗口中运行。相反，需要像文本、照片、图形那样将它放到 Web 页面中，从而与页面的其他部分无缝集成。

图 4.3 所示为在 Web 页面上运行的 RootApplet。applet 窗口是那个白框，显示的是该程序的输出：the square root of 255 is 15。该页面中的标题、文本段，以及那张灯泡照片都是 Web 页面的普通元素。

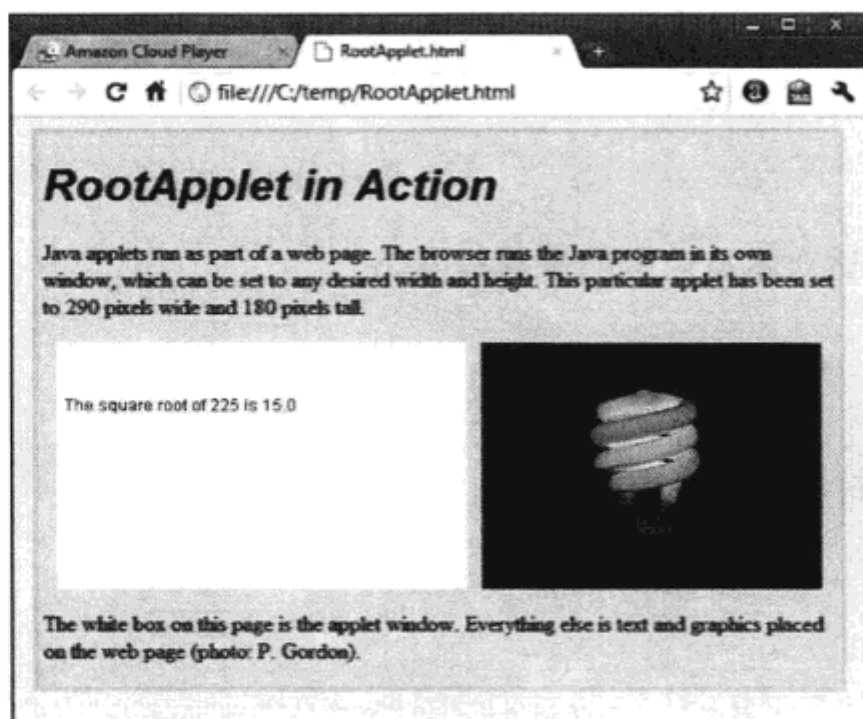


图 4.3

Web 页面中的 RootApplet 载入到 Google Chrome 浏览器中

Java applet 可以像该项目的输出一样，是静态的，但这对 Java 语言来说则是暴殄天物。applet 通常用来显示的是动态内容，比如实时的股票行情、聊天室中的聊天内容，以及视频

游戏等。

4.4 总结

在本章中，读者创建了一个 Java 应用程序和一个 applet，这两种程序在编写和运行方式上有多个重要的差别。

接下来的几章将继续介绍应用程序，让读者成为经验更丰富的 Java 程序员。应用程序的测试比较容易，因为它不需要创建 Web 页面来进行查看；而且应用程序的创建也很容易，同时它的功能也更强大。

4.5 问与答

问：传递给 Java 应用程序的所有参数都必须是字符串吗？

答：应用程序在运行时，Java 将所有参数都转换为字符串进行存储。要使用整型或其他非字符串参数，必须将其进行转换，这将在第 11 章介绍。

问：既然 applet 是在 Web 页面中运行，应用程序可以在任何地方运行，那么 Java Web Start 启动的 Java 程序是什么呢？

答：Java Web Start 是一个从 Web 浏览器启动 Java 应用程序的方法。用户通过单击 Web 页面上的一个链接来运行程序。与先下载，后运行安装向导，然后再启动的桌面程序相比，这种方式无疑更为简便。

尽管应用程序是从浏览器中运行的，但是 Java Web Start 程序仍然是应用程序，而不是 applet。应用程序永远都是最新的，因为它在每次运行时，都是通过 Web 从程序提供者那里下载到的。

Google Web Toolkit (GWT) 是一组用于 Web 编程的开源工具，它可以将 Java 程序转换为 JavaScript，从而使得它在 Web 浏览器中运行时速度更快，更可靠，而且还不需要 JVM 的介入。

4.6 测验

请回答下面的问题，以测试对本章内容的掌握程度。

4.6.1 问题

1. 哪种类型的 Java 程序可以在浏览器内部运行？
 - a. applet。
 - b. 应用程序。
 - c. 两者都不可以。
2. JVM 表示什么意思？
 - a. Journal of Vacation Marketing。



- b. Jacksonville Veterans Memorial。
 - c. Java Virtual Machine。
3. 如果你在给 Java 应用程序传递信息的方式上与他人发生争议，那么争议的焦点是什么？
- a. 关于字符串的争论。
 - b. 关于参数的争论。
 - c. 关于功能的争论。

4.6.2 答案

- 1. a. applet 作为 Web 页面的一部分运行，而应用程序可以在任何地方运行。
- 2. a、b 或 c。这是一个脑筋急转弯。这 3 个选项都可以用 JVM 来表示。但是 Java Virtual Machine 则是读者在接下来的 20 章中需要牢记的。
- 3. b. 应用程序以参数的方式接受信息。

4.7 练习

要应用有关 applet 和应用程序的知识，建议完成下面的练习：

- 根据 Root 应用程序，编写一个可以显示 625 的平方根的 NewRoot 应用程序。
- 根据 Root 应用程序，编写一个 NewRoot 应用程序，使它可以显示作为参数输入的那个值的平方根。

有关完成这两个练习的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 5 章

在程序中存储和修改信息

本章介绍如下内容：

- 创建变量；
- 使用不同类型的变量；
- 在变量中存储值；
- 在数学表达式中使用变量；
- 把一个变量的值赋给另一个变量；
- 递增或递减变量的值。

在第 2 章中，我们使用到了一个变量，它是一个用于存储信息的特殊位置。程序运行时，变量中存储的信息可以发生改变。在你的第一个程序中，变量中存储了一个字符串。字符串只是可在变量中存储的一种信息类型，变量还可以存储字符、整数、浮点数和其他类型的信息。

本章将更详细地介绍如何在 Java 程序中使用变量。

5.1 语句和表达式

计算机程序是一组告诉计算机做什么的指令，每一个指令称为语句。下面就是 Java 程序中的一条语句：

```
int highScore = 450000;
```

在 Java 程序中，可以使用括号将一组语句编组，这一组语句称为块语句（block statement）。请看下面的程序片段：

```

1: public static void main(String[] args) {
2:     int a = 3;
3:     int b = 4;
4:     int c = 8 * 5;
5: }

```

其中第 2~4 行是一条块语句。第 1 行的左大括号表示块语句开始，第 5 行的右大括号表示块语句结束。

有些语句被称为表达式，原因是它包含一个数学表达式，并能得到一个结果。在上面的例子中，第 4 行就是一个表达式，因为它将变量 `c` 的值设置为 8 和 5 的乘积。在接下来的几节中，读者将会用到表达式。

5.2 指定变量类型

计算机在运行程序时，变量是计算机记住信息的主要方法。第 2 章的 `Saluton` 程序使用 `greeting` 变量来存储 “`Saluton mondo!`” 消息。计算机需要记住这段文本，以便稍后显示。

在 Java 程序中，创建变量所使用的语句必须包含下列两项内容：

- 变量名；
- 变量存储的信息类型。

变量也可以包含将要存储的信息的值。

要看不同类型的变量以及如何创建它们，可以运行 `NetBeans`，然后创建一个新的空 Java 文件，其类名为 `Variable`。

开始编写程序，输入下面的代码：

```

class Variable {
    public static void main(String[] args) {
        // Coming soon: variables
    }
}

```

对其进行修改前保存该文件。

5.2.1 整数和浮点数

至此，程序 `Variable` 有了一个 `main()` 块，其中只包含一条语句：注释 `//Coming soon: variables`。删除注释并输入下列语句：

```
int tops;
```

这条语句创建一个名为 `tops` 的变量，但没有给它赋值，因此该变量此时是一个空的存储空间。语句开头的 `int` 指定 `tops` 是一个用于存储整数的变量。可以使用 `int` 类型来存储计算机程序所需的大多数整数，它能够存储 $-2.14 \times 10^9 \sim 2.14 \times 10^9$ 的任何整数。

在语句 `int tops;` 的末尾换行，然后输入下面的语句：


```
float gradePointAverage;
```

这条语句创建一个名为 `gradePointAverage` 的变量，其中 `float` 表示浮点数。浮点数变量用来存储可能包含小数的数值。

`float` 变量类型可以存储高达 38 位的十进制数，而 `double` 变量类型可以存储高达 300 位的十进制数。

By the Way

注意：

可以使用浮点变量来存储像 2.25 这样的绩点成绩 (GPA) (我进入北德克萨斯大学的 GPA 就是 2.25)，还可用于存储像 0 这样的数字。我当年凭 GPA 进入一所好研究院的机会也是 0。不过，我在 1996 年找工作时，仍然被 Sams 出版社慧眼识珠，招募为他们的计算机图书作者。

5.2.2 字符和字符串

到目前为止，你所处理的变量都是数值型的，因此可能认为所有变量都是用来存储数字的。也可以使用变量来存储文本，在变量中可以存储两种类型的文本：字符和字符串。字符是单个字母、数字、标点符号或其他符号。字符串是一组字符。

创建 `Variable` 程序的下一步是创建一个 `char` 变量和一个 `String` 变量。在语句 `float gradePointAverage;` 后面加入下面两条语句：

```
char key = 'C';
String productName = "Larvets";
```

在程序中使用字符值时，必须用单引号将赋给变量的字符值括起来，而对于字符串值必须用双引号括起来。

引号可以将字符或字符串同变量名或其他语句部分区分开来。请看下面的语句：

```
String productName = Larvets;
```

这条语句看起来好像告诉计算机，创建一个名为 `productName` 的字符串变量，并将文本值 `Larvets` 赋给该变量。但是由于没有使用双引号将单词 `Larvets` 括起，计算机认为要将 `productName` 的值设置为变量 `Larvets` 的值。

加入 `char` 和 `String` 语句后，程序将如程序清单 5.1 所示。在更改之后，不要忘记存盘。

程序清单 5.1 Variable 程序

```
1: class Variable {
2:     public static void main(String[] args) {
3:         int tops;
4:         float gradePointAverage;
5:         char key = 'C';
6:         String productName = "Larvets";
7:     }
8: }
```

Variable 程序中的最后两个变量在创建时,使用等号“=”赋予了初始值。在 Java 程序中,可以用这种方法给任何变量赋值,这将在本章后面看到。

该程序可以运行,但是不会产生输出。

注意:

尽管其他变量类型都是小写字母(int、float 和 char),但创建字符串变量时,单词 String 的首字母必须大写。在 Java 程序中,字符串与变量语句中使用的其他信息类型不同,这将在第 6 章介绍。

By the Way

5.2.3 其他数值类型的变量

前面介绍的变量是本书使用的主要变量类型,读者编写的大多数 Java 程序也可能如此。在特殊情况下,还可使用其他几种变量。

还有 3 种其他类型的整型变量。第一种为 byte,取值范围为-128~127 的整数。下面的语句是创建一个名为 escapeKey 的变量,其初始值为 27:

```
byte escapeKey = 27;
```

第二种为 short,可用于存储比 int 类型小的整数,其取值范围为-32768~32767 的整数,如下例所示:

```
short roomNumber = 222;
```

最后一种数值变量类型的是 long,通常用于存储 int 类型无法存储的整数。long 变量可以存储 $-9.22 \times 10^{18} \sim 9.22 \times 10^{18}$ 的整数。

在 Java 中处理很大的数值时,很难一目了然地看到具体的数值,如下面的语句所示:

```
long salary = 264400000;
```

除非你数一下 0 的个数,否则你很难看出它表示的是 264.4 百万美元。Java 7 对此做了改进,在处理很大的数值时,可以在数值中间使用下划线(_),如下所示:

```
long salary = 264_400_000;
```

下划线将会忽略掉,因此变量的值没有发生变化,这只是一种让数值更容易阅读的方式。

警告:

如果 NetBeans 的 IDE 不能识别 Java 7,则 Java 7 中提供的所有改进(包括数值中的下划线),在源代码编辑器中,都将标识为错误。对 IDE 进行设置,使其识别 Java 7 的方法将在第 7 章介绍。

Watch Out!

5.2.4 布尔型变量

Java 有一种特殊类型的变量叫布尔变量,它只能存储 true 或 false。乍一看,布尔型变量好像没多大用途,除非要编写大量基于计算机的真假测验。然而,编写程序时,很多情况下都需要使用布尔型变量。下面是一些可使用布尔型变量解决的问题。

- 用户是否按下了某个键？
- 游戏结束了吗？
- 银行账户透支了吗？
- 这些裤子让我看起来更胖了么？
- 兔子能吃 Trix 吗？

下面这条语句创建了一个名为 `gameOver` 的布尔型变量：

```
boolean gameOver = false;
```

这个变量的初始值为 `false`，这样的语句可在游戏程序中指示游戏还没有结束。然后，当导致游戏结束的事件发生时，将变量 `gameOver` 设置为 `true`。

虽然在程序中，布尔型变量的两个可能取值 `true` 或 `false` 看起来像字符串，但不应将它们用双引号括起。第7章将更详细地介绍布尔型变量。

By the Way

注意：

布尔数字是用 George Boole (1815~1864) 的名字命名的。Boole 是一名数学家，在成年前靠自学成才。他发明了布尔代数学，这是计算机编程、数字电路和逻辑学的重要组成部分。可以想象，他在孩童时就能够做很好的真假测试。

5.3 给变量命名

在 Java 中，变量名可以以字母、下划线字符 (`_`)、美元符号 (`$`) 打头。名称的其余部分可以是任何字母或数字。在遵守这些规则的情况下，你可以取任何喜欢的变量名，但应该采用统一变量命名规则。本节将介绍常用的变量命名方法。

在变量名的问题上，Java 是区分大小写的，所以变量名在整个程序中必须大小写一致。例如，如果变量 `gameOver` 在程序的其他地方写成 `GameOver`，编译程序时使用 `GameOver` 的地方将导致错误。

变量名应该描述变量的用途，第一个字母应该小写，如果变量名有多个单词组成，则将其余单词的首字母大写。例如，如果要创建一个整型变量，用于存储游戏中创记录的最高分，可使用下面的语句：

```
int allTimeHighScore;
```

在变量名中不能使用标点符号和空格，因此下面的变量名都是非法的：

```
int all-TimeHigh Score;  
int all Time High Score;
```

如果在程序中使用这些变量名，NetBeans 将在源代码编辑器中的相应行处用红色警告图标来显示，表示这是一个错误。

5.4 在变量中存储信息

在 Java 程序中，可以在创建变量时就给它赋值，也可以以后再给变量赋值。

要在创建变量时就给它赋初值，可以使用等号(=)。下面的例子创建双精度浮点变量 `pi`，且其初始值为 3.14：

```
double pi = 3.14;
```

所有数值型变量都可以用类似的方法赋值。如果创建的是字符或字符串变量，必须像前面介绍的那样用引号将赋给变量的值括起来。

如果两个变量的类型相同，也可以将一个变量的值赋给另一个变量，请看下面的例子：

```
int mileage = 300;
int totalMileage = mileage;
```

首先，创建了一个整型变量 `mileage`，并且其初始值为 300。接下来，创建了另一个整型变量 `totalMileage`，并将变量 `mileage` 的值赋给它。这两个变量的初始值相同，都是 300。在接下来的几章中，将介绍如何将一个变量的值转换为另一个变量的类型。

警告：

如果没有给变量赋初始值，则在另外一条语句中使用它之前，必须先赋值。如果没有赋值，则在程序编译时将报错，并显示如下错误消息“变量可能还没有初始化”。

**Watch
Out!**

前面已经学到，Java 有多个相似的数值变量，来存储不同大小的数值。`int` 和 `long` 都存储整数，但是 `long` 存储的数值范围更大。`float` 和 `double` 都存储浮点数值，但是 `double` 类型存储的范围更大。

你可以在数值后面追加一个字母来指示数值的类型，如下面的语句所示：

```
float pi = 3.14F;
```

数值 3.14 后面的 F 表示这是一个 `float` 浮点数值。如果忽略了这个字母，Java 会认为 3.14 是一个双精度浮点数值。字母 L 表示长精度整数，D 表示双精度浮点数值。

Java 的另外一种命名约定是，值不变的变量名全部大写。这些变量被称为常量，下面创建了 3 个常量。

```
final int TOUCHDOWN = 6;
final int FIELDGOAL = 3;
final int PAT = 1;
```

由于常量的值不变，读者可能会问，为何使用常量，只需使用赋给常量的值即可。使用常量的优点之一是，程序更容易理解。

在上面的 3 个语句中，常量的名字是大写的。尽管这在 Java 中不是必需的，但是编程人员已经将其作为区分常量和变量的一种标准约定。

5.5 运算符

通过使用 +、-、*、/ 和 % 等运算符，就可以在语句中使用数学表达式。在 Java 程序中使用这些运算符，就可以处理数值运算。

在 Java 中使用 + 运算符的加法表达式语句如下所示：

```
double weight = 205;
weight = weight + 10;
```

上面第2条语句使用+运算符将 weight 变量的值设置为 weight 变量的初始值与 10 的和。使用-运算符的减法表达式如下：

```
weight = weight - 15;
```

该表达式将 weight 变量的值设置为 weight 变量的初始值减去 15 后的结果。

使用/运算符的除法表达式如下：

```
weight = weight / 3;
```

该表达式将 weight 变量的值设置为其初始值除以 3 后的结果（取整）。

要从除法表达式中找到余数，可以使用%运算符（也称为取模运算符）。下面两条语句可以求得 245 除以 3 之后的余数。

```
int remainder = 245 % 3;
```

乘法表达式使用的是*运算符。下面这条复杂的语句使用了乘法表达式。

```
int total = 500 + (score * 12);
```

表达式 `score * 12` 中，变量 score 与 12 相乘。这条完整的语句是将变量 score 与 12 相乘，然后再加 500，最后将结果赋给 total 变量。如果 score 的值为 20，则 total 变量的最终值是 740，也即 $500 + (20 * 12)$ 。

5.5.1 变量的递增与递减

在程序中，常见的任务是将变量的值加 1 或减 1。其中加 1 的操作称为变量递增，减 1 的操作称为变量递减。这些任务可以使用相应的运算符来完成。

使用++运算符可以将变量加 1，如下面的语句所示：

```
x++;
```

该语句将存储在 x 变量中的值进行加 1 操作。

使用--运算符可以将变量减 1，如下所示：

```
y--;
```

该语句将变量 y 中的值减 1。

也可以将递增运算符和递减运算符放置在变量名的前面，如下面的语句所示。

```
++x;  
--y;
```

将运算符放置在变量前面，称之为运算符前置；放到后面，则称之为运算符后置。

By the Way

注意：

有点困惑？其实，当你回想一下在小学时学到的前缀的概念，就会发现前置与后置的概念很简单。与 sub-或 un-这样放在单词前面的前缀相同，前置运算符是放置在变量名的前面，而后置运算符则是放置在变量名的后面。

在表达式中使用递增、递减运算符时，一定要注意运算符前置和运算符后置的区别，这

很重要。

考虑下面的语句：

```
int x = 3;
int answer = x++ * 10;
```

在这两条语句处理完毕之后，`answer` 变量的值是多少呢？你可能认为应该是 40。当然，如果是对 3 加 1，这样等于 4，然后再乘以 10，这样就是 40。

然而，`answer` 变量实际的值却是 30，因为这里使用的后置运算符，而不是前置运算符。

当表达式中的变量使用了后置运算符时，只有当整个表达式执行完毕之后，变量的值才会发生改变。语句 `int answer = x++ * 10` 的执行顺序与下面的两条语句相同。

```
int answer = x * 10;
x++;
```

如果使用的是前置运算符，则顺序相反。也就是说，如果表达式的变量中使用的是前置运算符，则该变量的值会在表达式执行完毕之前就发生改变。

来看下面的语句：

```
int x = 3;
int answer = ++x * 10;
```

此时 `answer` 变量的值将等于 40。前置运算符会在表达式执行之前，先将变量 `x` 加 1。语句 `int answer = ++x * 10` 的执行顺序与下面的语句相同：

```
x++;
int answer = x * 10;
```

读者很容易就会对++运算符和--运算符产生恼怒心理，因为它们不像本书中出现的其他概念那样，容易让人理解。

注意：

第 1 章介绍 C++ 编程语言的名称时，指出这是一个玩笑的结果，以后读者将明白。知道运算符++，读者就能够明白为什么 C++ 使用两个+而不是一个。也许可笑：由于 C++ 在 C 语言的基础上增加了很多新特性和功能，因此将其命名为 C++。

在阅读本书章后，读者也能够讲这个笑话，且全世界 99% 的人都不会理解。

By the Way

在编写 Java 程序时，也可以不使用递增和递减运算符，通过如下方式来使用+和-运算符，同样也可以实现同样的结果。

```
x = x + 1;
y = y - 1;
```

递增和递减是很有趣的便捷方式，但也会让表达式看起来更长。

5.5.2 运算符优先级

当在一个表达式中使用多个运算符时，需要知道计算机处理这些运算符的先后顺序。来看下面的语句：

```
int y = 10;  
x = y * 3 + 5;
```

在计算机处理这些语句时，除非你事先知道其处理的先后顺序，否则将很难确定 x 变量的值到底是多少。取决于 $y*3$ 先执行，还是 $3+5$ 先执行， x 的结果可能是 35，也可能是 80。

在计算表达式时，将会按照如下顺序来执行。

1. 先执行递增和递减操作。
2. 然后执行乘、除以及取模运算。
3. 然后执行加、减操作。
4. 然后是“比较”操作。
5. 最后使用等号(=)来设置变量的值。

由于乘法运算先于加法运算发生，因此可以反过头来看前面的例子，然后得出答案： y 先与 3 相乘，其结果是 30，然后再加 5，这样 x 变量的值最终为 35。

“比较”操作将在第 7 章讲解。其他运算顺序已经在本章讲解了，因此现在可以求出下面语句的结果了。

```
int x = 5;  
int number = x++ * 6 + 4 * 10 / 2;
```

上述语句执行完毕之后， $number$ 变量的值是 50。

计算机是如何得出这个结果来的呢？计算机首先处理递增运算符，因此 $x++$ 表达式将变量 x 的值设为 6。注意， $++$ 运算符位于变量 x 的后面，是后置运算符，也就是说，在计算该表达式时，使用的是变量 x 的初始值。

由于使用的是变量 x 在递增之前的初始值，因此表达式如下所示：

```
int number = 5 * 6 + 4 * 10 / 2;
```

现在，计算机开始从左向右处理乘除运算。即 5 先乘以 6，然后是 4 乘以 10，这个结果再除以 2，这样表达式成为下面的样子：

```
int number = 30 + 20;
```

该表达式将变量 $number$ 的值设置为 50。

如果想以不同的顺序来对表达式求解，可以使用小括号把表达式的一部分括起来，它们就可以先被处理了。例如， $x = 5 * 3 + 2$ ；在执行完毕时， x 的值是 17，原因就是先执行乘法运算，然后再执行加法运算。然而，如果将该表达式修改为下面的形式：

```
x = 5 * (3 + 2);
```

此时，括号内的表达式将先被执行，这样 x 的最终结果就是 25。读者可以根据需要在语句中使用括号。

5.6 使用表达式

当你小时候在学校遇到令人讨厌的数学题时，是否抱怨过高次乘方，发誓今后再也不使

用这样的知识？对不起，这里可能破坏你的誓言，但所有的老师都是正确的——这些数学知识将在计算机编程中得到应用。这真是个坏消息。

但好消息是计算机将帮你做这些数学题。计算机程序中经常使用表达式，可以使用它们完成下面这样的任务：

- 修改变量的值；
- 在程序中计数；
- 在程序中使用数学公式。

编写计算机程序并使用表达式时，你又回到了过去的数学课中。表达式可以使用加、减、乘、除和求模。

要看一看运行中的表达式，回到 NetBeans 并创建一个新的 Java 文件，其类名为 `PlaneWeight`。这个程序可以记录某一个人在进入到太阳系时，其体重的增减。输入程序清单 5.2 中的所有代码。下面依次讲解该程序的每一部分。

程序清单 5.2 `PlaneWeight` 程序

```

1: class PlanetWeight {
2:     public static void main(String[] args) {
3:         System.out.print("Your weight on Earth is ");
4:         double weight = 205;
5:         System.out.println(weight);
6:
7:         System.out.print("Your weight on Mercury is ");
8:         double mercury = weight * .378;
9:         System.out.println(mercury);
10:
11:        System.out.print("Your weight on the Moon is ");
12:        double moon = weight * .166;
13:        System.out.println(moon);
14:
15:        System.out.print("Your weight on Jupiter is ");
16:        double jupiter = weight * 2.364;
17:        System.out.println(jupiter);
18:    }
19: }
```

输入完毕后，保存文件。然后通过菜单命令 `Run->Run File` 来运行该程序。输出结果如图 5.1 所示。

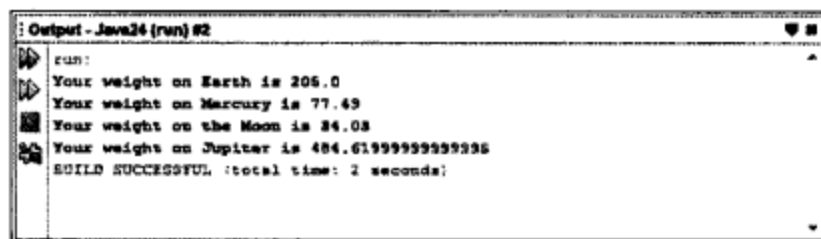


图 5.1
`PlaneWeight` 程序的输出

和你创建的其他程序一样，`PlaneWeight` 程序也使用了 `main()` 块语句来进行相应的处理。该语句可以分成下面 4 部分。

1. 第 3~5 行：将人在地球上的初始体重设置为 205。
2. 第 7~9 行：计算人在水星上的体重，其值减少。

3. 第 11~13 行：计算人在月球上的体重，其值减少。

4. 第 15~17 行：计算人在木星上的体重，其值增加。

第 4 行创建了 `weight` 变量，并使用 `int` 将其指定为整型变量。该变量的初始值为 205，在整个程序中用于监控人的体重。

下一行类似于程序中的其他语句：

```
System.out.println(weight);
```

命令 `System.out.println()` 显示其括号中包含的字符串。在第 3 行，`System.out.print()` 命令显示文本 “Your weight on Earth is”。程序中还有多个 `System.out.print()` 和 `System.out.println()` 语句。

它们之间的区别在于，`print()` 在显示完文本后，不会自动换行，而 `println()` 则会自动换行。

5.7 总结

学习了变量和表达式后，读者将能够在程序中向计算机发出更多的指令。有了本章学习到的知识，读者可以编写程序，完成众多类似计算器的任务，轻松地处理复杂的数学等式。

数字只是一种可存储到变量中的信息，还可以存储字符、字符串以及特殊值 `true` 和 `false`（被称为布尔型变量）。下一章将介绍字符串变量以及如何使用它们。

5.8 问与答

问：在 Java 程序中，一行和一条语句是一回事吗？

答：不是。虽然本书的程序都将每条语句放在一行内，但旨在让程序更容易理解，并非必须如此。

Java 编译器在编译程序时不考虑行、空格或其他格式问题，编译器只要求每条语句以分号结束。下面的这一行代码对 Java 来说也无问题：

```
int x = 12; x = x + 1;
```

可以在一行中放置多条语句，但这样做让阅读源代码的人难以理解。鉴于这种原因，通常不建议这样做。

问：为什么变量名的首字母要小写，如 `gameOver`？

答：这是一种命令约定，而且对你的编程很有帮助。首先，它可以让变量更容易识别，从而与 Java 程序中的其他元素区分开来。另外，通过采用统一的变量命名规则，当你在程序的多个地方使用同一个变量时，更容易修复出现的错误。本书使用的命名规则已被大多数 Java 程序员采用多年。

问：我可以在 Java 程序中将二进制值赋值给整型变量么？

答：在 Java 7 中可以这样做。在为整型变量赋二进制值时，需要在数值前面添加字符 `0b`。由于 1101 是数字 13 的二进制值，因此下面的语句是将 13 赋值给整型变量 `z`。

```
int z = 0b0000_1101;
```

下划线的目的只是为了增加语句的可读性，在编译阶段会被 Java 编译器忽略。

如果你的项目使用的不是 Java 7，则 NetBeans 会把该特性当作错误来对待。第 7 章将会对其详细讲解。

5.9 测验

回答下列问题，以测试对变量、表达式以及本章介绍的其他知识的理解程度。

5.9.1 问题

1. 位于左大括号和右大括号之间的一组语句称为什么？
 - a. 块语句。
 - b. 组件 (groupware)。
 - c. 用括号括起的语句。
2. 布尔型变量用于存储 true 或 false。
 - a. 正确。
 - b. 错误。
 - c. 不，谢谢，我吃过了。
3. 变量名不能以哪些字符打头？
 - a. 美元符号 (\$)。
 - b. 双斜线 (//)。
 - c. 字母。

5.9.2 答案

1. a. 括起来的语句称为块语句或块。
2. a. 布尔型变量只能存储 true 或 false。
3. b. 变量名可以以字母、美元符号 (\$) 或下划线 (_) 打头。如果变量名以两个斜杠打头，该代码行后面的内容将被忽略，因为双斜杠表示注释行。

5.10 练习

为充分复习本章的主题，请完成下面的练习。

- 扩展 PlaneWeight 程序，记录人在金星上的体重 (大约是人在地球上的体重的 90.7%)，以及人在天王星上的体重 (大约是地球上的 88.9%)。
- 创建一个简短的 Java 程序，它使用整型变量 x 和 y，并显示 x 和 y 的平方和。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 6 章

使用字符串来交流

本章介绍如下内容：

- 使用字符串来存储文本；
- 在程序中显示字符串；
- 在字符串中包含特殊的字符；
- 拼接字符串；
- 在字符串中包含变量；
- 比较字符串；
- 判断字符串的长度。

在电影《钢琴》中，Holly Hunter 扮演的 Ada 是一位年轻的苏格兰妇女，她从 6 岁起就是哑巴，只能通过弹钢琴来表达自己的。

同 Ada 一样，计算机程序能够安静地工作，从来不停下来聊天或演奏钢琴。然而如果《钢琴》告诉了我们什么，那就是交流和食物、水和住所一样，是人类的基本需求（这部电影还告诉我们，演员 Harvey Keitel 对自己的肢体表达能力很自信，但这不是本书讨论的内容）。

Java 程序将字符串作为与用户交流的主要方式。字符串是一组文本，可以包含字母、数字、标点符号及其他字符。本章将介绍如何在 Java 程序中使用字符串。

6.1 在字符串中存储文本

字符串用来存储文本并显示给用户。字符串中最基本的元素是字符。字符可以是一个字母、数字、标点符号，或者是其他符号。

在 Java 程序中，字符是可存储到变量中的信息类型之一，字符型变量是在语句中使用 `char` 来创建的，如下所示：

```
char keyPressed;
```

这条语句创建了一个名为 `keyPressed` 的变量，可用于存储字符。当创建字符型变量时，可以设置其初始值，如下所示：

```
char quitKey = '@';
```

字符值必须用单引号括起来。

字符串是一组字符，可以使用 `String` 和变量名来创建存储字符串值的变量，如下所示：

```
String fullName = "Ada McGrath Stewart";
```

这条语句创建了一个名为 `fullName` 的字符串变量，并在其中存储了文本“Ada McGrath Stewart”，这是 Hunter 的钢琴家的全名。在 Java 语句中，字符串用双引号括起，但双引号不是字符串的一部分。

不同于你前面使用过的其他类型变量：`int`、`float`、`char`、`boolean`，表示字符串类型的 `String` 的首字母必须大写。

这样做的原因是，字符串类型与 Java 中的其他类型有些不同。字符串是一种称为对象的特殊信息，而所有对象类型名的首字母都必须大写。有关对象的知识将在第 10 章介绍。在本章需要注意的是，字符串与其他变量类型不同，由于这种差别，在语句中指定字符串类型时，`String` 的首字母必须大写。

6.2 在程序中显示字符串

在 Java 程序中，显示字符串的最基本方法是使用 `System.out.println()` 语句。该语句可在括号中接收字符串和其他变量，并将它们显示在系统输出设备中，即计算机监视器。下面是一个例子。

```
System.out.println("Silence affects everyone in the end.");
```

上述语句将显示下列文本：

```
Silence affects everyone in the end.
```

在屏幕上显示文本通常称为打印，这就是 `println()` 代表的意思：打印该行。你可以使用 `System.out.println()` 语句显示用双引号括起的文本和变量（稍后你将会看到）。将所有要显示的内容都放在括号内。

另一种显示文本的方法是调用 `System.out.print()`，该语句显示括号中的字符串和其他变量，但不同于 `System.out.println()`，它让接下来的语句在同一行显示文本。

可以连续使用 `System.out.print()` 多次，将内容显示在同一行，如下例所示：

```
System.out.print("She ");
System.out.print("never ");
System.out.print("said ");
System.out.print("another ");
System.out.println("word.");
```

这些语句输出的文本如下：

```
She never said another word.
```

6.3 在字符串中使用特殊字符

创建或显示字符串时，其文本必须用双引号括起。这些双引号不会显示出来，这就提出了一个很好的问题：如果要显示双引号该怎么办呢？

为显示双引号，Java 创建了一个特殊编码"\"，可放到字符串中。在字符串中遇到该编码时，将其替换为双引号。例如，请看下面的例子：

```
System.out.println("Jane Campion directed \"The Piano\" in 1993.");
```

这段代码显示如下内容：

```
Jane Campion directed "The Piano" in 1993.
```

可以采用这种方式在字符串中插入多个特殊字符，下面列表出了这些特殊字符，注意到每个都是以反斜线（\）打头。

特殊字符	显示
\'	单引号
\"	双引号
\\	反斜线
\t	制表符
\b	回退符
\r	回车符
\f	走纸符
\n	换行符

换行符的作用是在下一行行首显示换行符后面的文本，请看下面的例子：

```
System.out.println("Music by\nMichael Nyman");
```

这条语句将显示如下内容：

```
Music by
Michael Nyman
```

6.4 拼接字符串

使用 System.out.println() 语句以及用其他方式处理字符串时，可以使用加号（+）将两个字符串拼接起来。这里用到的加号和用来对数值进行求和的加号相同。

将运算符+用于字符串，其含义与原来不同：不是执行数学运算，而是将两个字符串拼接起来。这导致字符串显示在一起，或使用两个小字符串组合成一个长字符串。

这种行为用拼接（concatenation）来描述，因为它的意思是将两样东西连接起来。

注意：

读者在学习编程技巧时，可能会在其他书中看到 concatenation 这个术语。但是，本书在讲解字符串和字符串结合时，用的是 pasting 这个术语。

By the Way

下面的语句使用+运算符来显示一个长字符串：

```
System.out.println("\n'The Piano' is as peculiar and haunting as any" +  
    " film I've seen.\n\t— Roger Ebert, Chicago Sun-Times");
```

这里不是将整个字符串放在单独一行（如果这样，以后查看程序时将更难理解），而是使用运算符+将文本文件分成两行。执行这条语句时，输出结果如下：

```
'The Piano' is as peculiar and haunting as any film I've seen."  
— Roger Ebert, Chicago Sun-Times
```

在该字符串中使用了几个特殊字符：\n、\'、\n 和\t。为更好地熟悉这些字符，请将输出和生成输出的 System.out.println() 语句进行比较。

6.5 将其他变量用于字符串中

虽然可以使用+运算符将两个字符串拼接起来，但更常见的是使用它将字符串和变量拼接起来。请看下面的例子：

```
int length = 121;  
char rating = 'R';  
System.out.println("Running time: " + length + " minutes");  
System.out.println("Rated " + rating);
```

这段代码的输出如下：

```
Running time: 121 minutes  
Rated R
```

这个例子说明了将+运算符用于字符串的独特之处：导致不是字符串的变量作为字符串显示出来。length 是一个整型变量，其值为 121，它显示在字符串“Running time:”和“minutes”之间。System.out.println() 语句用于显示一个字符串加上一个整数再加上一个字符串。这条语句之所以能够正常运行，是因为至少开头的部分是字符串。Java 语言通过提供这种功能使信息更容易显示。

读者可能想做的一件事情是，将字符串拼接多次，如下例所示：

```
String searchKeywords = "";  
searchKeywords = searchKeywords + "drama ";  
searchKeywords = searchKeywords + "romance ";  
searchKeywords = searchKeywords + "New Zealand";
```

这段代码导致变量 searchKeywords 被设置为“drama romance New Zealand”。第 1 行创建变量 searchKeywords 并将其设置为空字符串，因为双引号之间为空。第 2 行将变量 searchKeywords 设置为其当前值加上字符串 drama；接下来的两行用相同的方式再加上 romance 和 New Zealand。

可以看到，在变量后面拼接文本时，变量名将出现两次。Java 提供了一种快捷方式来简化该过程，这就是+=运算符。+=运算符将=和+运算符的功能融为一体。对于字符串，它用于在当前字符串后面加上其他字符串。上述 SearchKeywords 示例可以使用+=运算符简化为如下所示：

```
String searchKeywords = "";
searchKeywords += "drama ";
searchKeywords += "romance ";
searchKeywords += "New Zealand";
```

这段代码的效果与前面相同：将 searchKeywords 设置为 “drama romance New Zealand”。

6.6 字符串的高级处理

还有多种其他方式可用于查看字符串变量和修改其值。之所以有这些高级功能，是因为字符串在 Java 语言中是对象。通过处理字符串对象获得的知识，也适用于其他对象。

6.6.1 比较两个字符串

在程序中经常要比较两个字符串是否相等，为此可在带有两个字符串的语句之中使用 equals()，如下所示：

```
String favorite = "piano";
String guess = "ukulele";
System.out.println("Is Ada's favorite instrument a " + guess + "?");
System.out.println("Answer: " + favorite.equals(guess));
```

这里使用了两个字符串变量，一个是变量 favorite，用于存储 Ada 最喜欢的乐器名称：钢琴；另一个变量是 guess，用于存储对其最喜欢的乐器的猜测，该猜测是 Ada 最喜欢尤克里里琴。

第 3 行显示文本 “Is Ada’s favorite instrument a”、变量 guess 的值和问号。第 4 行显示文本 “Answer:” 以及下面的新内容：

```
favorite.equals(guess)
```

语句中的这部分称为方法，方法是在 Java 程序中完成任务的一种方式，这里的方法要完成的任务是，比较字符串 favorite 和字符串 guess 的值是否相等。如果这两个字符串的值相等，就显示 true，否则显示 false。下面是该示例的输出结果：

Output ▼

```
Is Ada's favorite instrument a ukulele?
Answer: false
```

6.6.2 确定字符串的长度

有时确定字符串的长度很有用，为此可使用方法 length()。该方法的工作原理与 equals() 相似，但只涉及一个字符串变量。请看下面的例子：

```
String cinematographer = "Stuart Dryburgh";
int nameLength = cinematographer.length();
```


该示例将整型变量 `nameLength` 的值设置为 15，方法 `cinematographer.length()` 计算字符串变量 `cinematographer` 包含的字符数，并将结果赋给整型变量 `nameLength`。

6.6.3 改变字符串的大小写

计算机很不灵活，不能识别明显相同的东西。虽然人很容易识别出文本 `Harvey keitel` 和 `HARVEY KEITEL` 是一回事，但大多数计算机不这么认为。例如，在本章前面介绍的 `equals()` 方法将果断地认为 `Harvey keitel` 不等于 `HARVEY KEITEL`。

为了克服这些障碍，Java 提供了将字符串变量全部转换为大写的方法 (`toUpperCase()`) 和全部转换为小写的方法 (`toLowerCase()`)。下面的例子演示了如何使用方法 `toUpperCase()`：

```
String baines = "Harvey Keitel";
String change = baines.toUpperCase();
```

这段代码将字符串变量 `change` 设置为字符串变量 `baines` 的大写形式，即 `HARVEY KEITEL`。`toLowerCase()` 方法的用法相同，但返回的是字符串的小写。

注意，`toUpperCase()` 方法不改变调用它时使用的字符串变量的大小写。在上述示例中，变量 `baines` 的值仍为 `Harvey keitel`。

6.6.4 查找字符串

处理字符串时，另一项常见的任务是，确定在一个字符串中能否找到另一个字符串。要在字符串中查找，可使用方法 `indexOf()`，并将要查找的字符串放在括号中。如果没有找到指定的字符串，`indexOf()` 返回 -1；如果找到，`indexOf()` 返回一个整数，指出该字符串的起始位置。字符串中字符位置从 0 开始编号，即第一个字符的位置为 0。在字符串 `The Piano` 中，文本 `Piano` 的起始位置为 4。

警告：

`indexOf()` 方法是区分大小写的，这也就意味着只有当目标字符串和搜索字符串的大小写完全相同时，才算查找成功。否则，`indexOf()` 返回 -1。

**Watch
Out!**

`indexOf()` 方法的一种用法是，在电影《钢琴》的剧本中，搜索 `Ada` 盛气凌人的丈夫跟她的女儿 `Flora` 的谈话：“`You are greatly shamed and you have shamed those trunks`”。

如果《钢琴》的剧本存储在变量 `script` 中，可以使用下面的语句从中搜索前面引用的部分：

```
int position = script.indexOf("you have shamed those trunks");
```

如果在 `script` 中找到文本 “`you have shamed those trunks`”，变量 `position` 将等于该文本在 `script` 中的起始位置，否则将等于 -1。

6.7 导演及演员名单

在电影《钢琴》中，Ada McGrath Stewart 从苏格兰搬到新西兰，来到一片陌生的土地并嫁给了一个不懂得欣赏钢琴的人。对于本章介绍的有些主题，读者可能与她一样有种失落感。

接下来，为了加深读者对前面介绍的字符串处理功能的理解，将编写一个 Java 程序，显示一部电影的导演和演员名单。你应该能够猜到该电影的名字。

返回 NetBeans 中的 Java24 项目，然后创建一个名为 Credits 的新 Java 类，在源代码编辑器中输入程序清单 6.1 中的所有文本，输入完毕之后存盘。

程序清单 6.1 Credits 程序

```
1: class Credits {
2:     public static void main(String[] args) {
3:         // set up film information
4:         String title = "The Piano";
5:         int year = 1993;
6:         String director = "Jane Campion";
7:         String role1 = "Ada";
8:         String actor1 = "Holly Hunter";
9:         String role2 = "Baines";
10:        String actor2 = "Harvey Keitel";
11:        String role3 = "Stewart";
12:        String actor3 = "Sam Neill";
13:        String role4 = "Flora";
14:        String actor4 = "Anna Paquin";
15:        // display information
16:        System.out.println(title + " (" + year + ")\n" +
17:            "A " + director + " film.\n\n" +
18:            role1 + "\t" + actor1 + "\n" +
19:            role2 + "\t" + actor2 + "\n" +
20:            role3 + "\t" + actor3 + "\n" +
21:            role4 + "\t" + actor4);
22:    }
23: }
```

编译程序前先浏览一遍程序，看是否能够明白各条语句的功能。对该程序的详细分析如下。

- 第 1 行将该 Java 程序命名为 Credits。
- 第 2 行是 main() 块语句的开头，程序的所有功能都是在该块语句中完成的。
- 第 4~14 行创建用于存储导演和演员以及影片信息的变量。其中一个变量 year，它是一个整型变量，其他变量都是字符串变量。
- 第 16~21 行是长语句 System.out.println()。在第 16 行和第 21 行的括号之间的信息都将显示到屏幕上。换行符\n 的作用将其后面的文本在下一行的行首显示。制表符\t 的作用是在输出信息中插入制表符。其他要显示的内容要么是文本，要么是字符串变量。
- 第 22 行结束 main() 块语句。
- 第 23 行结束整个程序。

如果提示有错误，可以修改程序中的输入错误，然后重新保存。NetBeans 将自动编译程

序。当运行程序时，将会看到类似于图 6.1 所示的输出窗口。

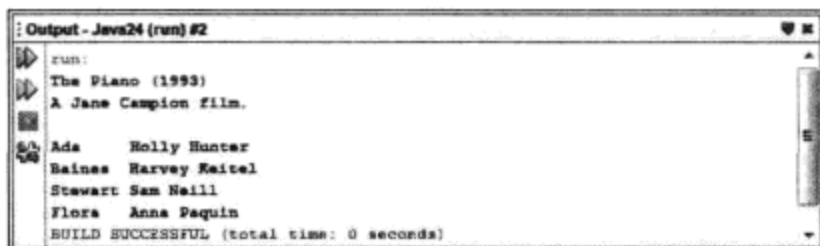


图 6.1

Credits 程序的输出

注意：

如果你对本章提到的电影《钢琴》、电影中的主人公，以及该电影的导演 Jane Campion 感兴趣，可以访问 Magnus Hjelstuen 的非官方网站，地址为 www.cadenhead.org/piano。

By the Way

6.8 总结

如果你的 Credits 程序运行正常，输出结果与图 6.1 相同，你的信心也应该增加。通过本书前 6 章的学习，你现在可以编写一个比较长的 Java 程序，而且也可以处理一些比较复杂的问题了。字符串是你在每次编程时都会用到的东西。

在影片《钢琴》的开头，Ada 的丈夫不允许其毛利人劳工将她的钢琴搬回家，她失去了钢琴。你很幸运，你在 Java 程序中使用字符串的技能不会其他任何人夺走，你可以通过多种方式使用使用字符串来与用户交流。

6.9 问与答

问：如何将字符串变量的值设置为空？

答：一对双引号之间没有任何文本就表示空字符串。下面的代码创建一个名为 `adaSays` 的字符串变量，并将其值设置为空：

```
String adaSays = "";
```

问：使用 `toUpperCase()` 方法好像不能将字符串中的字母全部转换为大写，我哪里操作不正确？

答：调用字符串对象的 `toUpperCase()` 方法时，实际上它并未修改该字符串对象，而是创建一个字母全部大写的新字符串，请看下面的语句：

```
String firstName = "Nessie";
String changeName = firstName.toUpperCase();
System.out.println("First Name: " + firstName);
```

这些语句的输出结果为 “First Name: Nessie”，因为变量 `firstName` 包含的是原来的字符串。如果将最后一条语句改为显示变量 `changeName`，输出结果将为 “First Name: NESSIE”。

当字符串在 Java 中创建之后，它们的值不会发生改变。

问：在 Java 中，所有方法都像 `equals()` 那样返回 `true` 或 `false` 吗？

答：方法被调用后，可以有不同的方式来进行响应。如果方法像 `equals()` 那样发回一个值，则被称为返回一个值。方法 `equals()` 方法返回一个布尔值，其他方法可能返回字符串、

整数、其他类型的值，也可能什么都不返回（使用 `void` 来表示）。

6.10 测验

回答下列问题，以测试对字符串理解和掌握的程度。

6.10.1 问题

1. 我的朋友要执行拼接操作，需要向权威部门报告吗？
 - a. 不，仅在冬季这才是非法的。
 - b. 是的，但要等到我将故事卖给 TMZ.com 再说。
 - c. 不，他所做的只是在程序中将两个字符串连接起来。
2. 为什么 `String` 的首字母要大写，而 `int` 等类型名的首字母不需要大写？
 - a. `String` 是一个完整的单词，而 `int` 不是。
 - b. 和 Java 中所有的对象一样，`String` 的首字母必须大写。
 - c. Oracle 的质量控制做得很糟糕。
3. 下列那项在字符串中添加一个单引号？
 - a. `<quote>`。
 - b. `\'`。
 - c. `'`。

6.10.2 答案

1. c. 拼接（Concatenation）指的是将两个字符串连接起来，它使用运算符`+`或`+=`。
2. b. 在 Java 中，所有的对象类型名的首字母都要大写，因此变量名的首字母都是小写，这样就不容易将变量和对象搞混。
3. b. 在字符串中插入特殊字符时，总是以单个反斜杠开头。

6.11 练习

通过下列练习来复习本章介绍的主题：

- 编写一个名为 `Favorite` 的小型 Java 程序，将本章中“比较两个字符串”一节的代码放在 `main()` 块语句中。测试该程序，确认其输出就像正文中描述的那样，Ada 最喜欢的乐器不是尤克里里琴。测试完毕后，将变量 `guess` 的初始值从 `ukelele` 改为 `piano`，再看看如果如何。
- 修改程序 `Credits`，将导演和全部演员的名字都用大写字母显示。

有关完成这些练习需要编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 7 章

使用条件测试进行判断

本章介绍如下内容：

- 使用 if 语句进行基本的条件测试；
- 测试一个值是大于还是小于另一个值；
- 测试两个值是否相等；
- 使用与 if 语句对应的 else 语句；
- 组合多个条件测试；
- 使用 switch 语句进行复杂的条件测试；
- 使用三元运算符创建复杂测试。

编写计算机程序时，你提供给计算机的是一系列称为语句的指令，这些指令被严格地执行。你可以让计算机计算令人讨厌的数学公式，它将为你的得出结果；也可以让计算机显示一些信息，它将忠实地完成。

然而，有时需要让计算机做出选择。例如，编写计算账目收支平衡的程序时，可能想在账户透支时让计算机给显示警告消息，而且只有当账户透支时，计算机才显示该消息；如果没透支也显示这样的消息，计算机程序就不准确，也让人感到不安。

在 Java 程序中，完成这项工作的方法是使用条件语句。而且仅当满足特定的条件时，条件语句才导致特定的操作。本章将介绍如何使用各种条件语句：if、else 和 switch。

在 Java 程序中，决策都是使用条件语句来完成的。在本章，读者将在 Java 程序中使用条件语句 if、else、switch、case、break 检查各种条件，还将使用多种条件运算符，如 ==、!=、<、> 和 ?，以及布尔型变量。

7.1 if 语句

在 Java 程序中，对条件进行测试所使用的最基本的方法是使用 if 语句。if 语句测试某个条件为 true 还是 false，并在条件为 true 时执行特定的操作。

使用 if 和条件来进行测试，如下面的语句所示：

```
if (account < 0) {  
    System.out.println("Account overdrawn; you need a bailout");  
}
```

if 语句使用小于运算符<来判断 account 变量是否小于 0。如果是，则运行 if 语句中的代码块，然后显示文本。

只有条件为真时，if 语句中的代码块才运行。在上面的例子中，如果变量 account 的值大于或等于 0，println 语句将不执行。注意，在 if 语句中测试的条件要用括号括起，如(account < 0)。

小于运算符 (<) 是可以在条件语句中使用的多种运算符之一。

7.1.1 小于和大于的比较

在前一节，像数学课中那样使用了运算符 (<)：表示小于。还有大于运算符 (>)，下面的语句演示了如何使用该运算符：

```
int elephantWeight = 900;  
int elephantTotal = 13;  
int cleaningExpense = 200;  
  
if (elephantWeight > 780) {  
    System.out.println("Elephant too fat for tightrope act");  
}  
  
if (elephantTotal > 12) {  
    cleaningExpense = cleaningExpense + 150;  
}
```

第一条 if 语句测试变量 elephantWeight 的值是否大于 780，第二条 if 语句测试变量 elephantTotal 的值是否大于 12。

如果上述语句位于这样的程序中，即变量 elephantWeight 等于 600，而变量 elephantTotal 等于 10，则 if 语句块中的语句将不会执行。

有时可能需要判断一个变量是否小于或等于某个值，为此需要使用运算符<=。下面是一个例子：

```
if (account <= 0) {  
    System.out.println("You are flat broke");  
}
```

还有一个>=运算符可以用于“大于或等于”测试。

7.1.2 相等和不等

在程序中要检测的另一种条件是否相等。变量是否等于特定的值？一个变量是否与另一

个变量相等？这些问题可以使用运算符`==`来回答，如下面的语句所示：

```
if (answer == rightAnswer) {
    studentGrade = studentGrade + 10;
}

if (studentGrade == 100) {
    System.out.println("Show off!");
}
```

警告：

用于检测是否相等的运算符由两个等号组成，即`==`。很容易将其同运算符`=`混淆，后者用于给变量赋值。在条件语句中，总是使用两个等号。

**Watch
Out!**

读者也可以测试不相等，即一个值是否等于另一个值，为此可使用运算符`!=`，如下例所示：

```
if (answer != rightAnswer) {
    score = score - 5;
}
```

除字符串变量之外（因为字符串是对象），可以将运算符`==`和`!=`用于任何类型的变量。

7.1.3 使用块语句组织程序

到目前为止，所有的 if 语句后面都有一个代码块，该代码块位于`{}`之间（这两个字符的技术术语应该是“大括号”）。

在本书前面，读者看到过如何使用块语句来标识 Java 程序中 `main()` 语句块的起始和结束。程序运行时，`main()` 语句块中的每条语句都被执行。

if 语句不需要块语句，它只占用单独的一行，如下面的例子所示：

```
if (account <= 0) System.out.println("No more money");
```

只有当 if 条件中的条件为 `true` 时，if 条件后的语句才能执行。

程序清单 7.1 的 Java 程序使用块语句来表示 `main()` 块。该块语句从第 2 行的左大括号“`{`”开始，到第 13 行的右大括号“`}`”结束。在 NetBeans 中创建一个空的 Java 文件，命名为 `Game`，然后输入程序清单 7.1 中的所有文本。

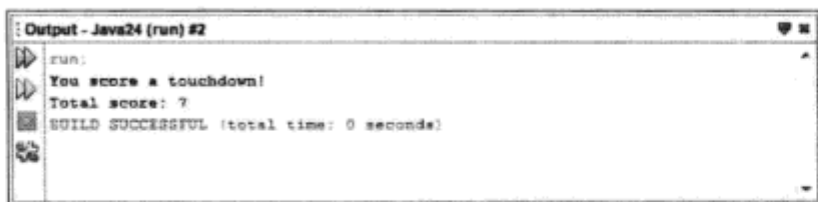
程序清单 7.1 Game 程序

```
1: class Game {
2:     public static void main(String[] arguments) {
3:         int total = 0;
4:         int score = 7;
5:         if (score == 7) {
6:             System.out.println("You score a touchdown!");
7:         }
8:         if (score == 3) {
9:             System.out.println("You kick a field goal!");
10:        }
11:        total = total + score;
12:        System.out.println("Total score: " + total);
13:    }
14: }
```

运行该程序时，得到的输出如图 7.1 所示。

图 7.1

Game 程序的输出
结果



也可以在 if 语句中使用块语句来让计算机在条件为 true 时执行多个操作。下面是一条包含块语句的 if 语句：

```
int playerScore = 12000;
int playerLives = 3;
int difficultyLevel = 10;

if (playerScore > 9999) {
    playerLives++;
    System.out.println("Extra life!");
    difficultyLevel = difficultyLevel + 5;
}
```

大括号用于将所有属于 if 语句的语句编组，如果变量 playerScore 大于 9999，将做 3 件事情：

- 将变量 playerLives 的值加 1（因为使用了递增运算符++）；
- 显示文本“Extra life!”；
- 将变量 difficultyLevel 的值加 5。

如果变量 playerScore 不大于 9999，将什么都不做，if 语句块中的 3 条语句都将被忽略。

7.2 if-else 语句

有时可能要在条件为 true 时做某些事情，而在条件为 false 时做另一些事情，为此可以结合使用 if 语句和 else 语句，如下例所示：

```
int answer = 17;
int correctAnswer = 13;

if (answer == correctAnswer) {
    score += 10;
    System.out.println("That's right. You get 10 points");
} else {
    score -= 5;
    System.out.println("Sorry, that's wrong. You lose 5 points");
}
```

不同于 if 语句，else 语句不包含条件。else 语句与前面离它最近的 if 语句相匹配。也可以使用 else 语句将多条 if 语句连接起来，如下例所示：

```
if (grade == 'A') {
    System.out.println("You got an A. Great job!");
} else if (grade == 'B') {
    System.out.println("You got a B. Good work!");
} else if (grade == 'C') {
    System.out.println("You got a C. What went wrong?");
} else {
    System.out.println("You got an F. You'll do well in Congress!");
}
```

通过这种方式将几个不同的 if 和 else 语句放在一起，可以处理很多条件。在上面的示例中，对 A 类学生、B 类学生、C 类学生和未来的议员分别发送不同的消息。

7.3 switch 语句

if 和 else 语句非常适合于只有两种情况的情形，但有时候需要考虑两种以上的情况。

从前面有关成绩的示例可知，可以结合使用 if 和 else 语句处理多种情况。

另一种方法是使用 switch 语句。使用 switch 可以测试多个不同的条件并做出相对的响应。在下面的例子中，使用 switch 语句对前面有关成绩的示例进行了重写：

```
switch (grade) {
    case 'A':
        System.out.println("You got an A. Great job!");
        break;
    case 'B':
        System.out.println("You got a B. Good work!");
        break;
    case 'C':
        System.out.println("You got a C. What went wrong?");
        break;
    default:
        System.out.println("You got an F. You'll do well in Congress!");
}
```

第 1 行的 switch 语句指定了要检测的变量，在本例中是 grade，然后使用 “{” 和 “}” 形成了一个块语句。

switch 语句中的每条 case 语句检查变量是否等于某个值，在 case 语句中使用的值可以是字符、整数或字符串。在这里，这些 case 语句中使用的值分别是字符 ‘A’、‘B’ 和 ‘C’。每条 case 语句后跟一条或两条语句。当某条 case 语句与 switch 语句中变量的值匹配时，计算机将处理其后面的语句，直到遇到 break 语句。

例如，如果变量 grade 的值为 B，将显示文本 “You got a B. Good work!”。接下来是 break 语句，因此不会执行 switch 语句的其他部分。break 语句告诉计算机退出 switch 语句。

default 语句用于处理所有 case 语句都不满足的情况。在这个例子中，如果变量 grade 不等于 ‘A’、‘B’ 或 ‘C’，将进入 default 语句。在程序中，并非在每个 switch 语句块中都需要使用 default 语句。如果没有 default 语句，且所有 case 语句条件都不满足，将什么也不做。

在 Java 7 中，switch-case 语句中的测试变量可以使用字符串。程序清单 7.2 中的 Commodity 类使用该语句来购买或销售商品（这里没有指明商品到底是什么）。该商品在购买时的价格为 balance-20 美元，卖出时的价格为 balance+15 美元。

switch-case 语句对名为 command 的字符串值进行测试，如果等于 “BUY”，则运行一个语句块；如果等于 “SELL”，则运行另一个语句块。

程序清单 7.2 Commodity 程序

```
1: public class Commodity {
2:     public static void main(String arguments) {
3:         String command = "BUY";
```

```

4:         int balance = 550;
5:         int quantity = 42;
6:
7:         switch (command) {
8:             case "BUY":
9:                 quantity += 5;
10:                balance -= 20;
11:                break;
12:            case "SELL":
13:                quantity -= 5;
14:                balance += 15;
15:        }
16:        System.out.println("Balance: " + balance + "\n"
17:            + "Quantity: " + quantity);
18:    }
19: }

```

在该程序的第 3 行，`command` 字符串被设置为“BUY”。当测试 `switch` 语句时，将运行第 9～第 11 行的 `case` 语句块。商品的 `quantity` 变量增加 5，`balance` 变量降低 20。

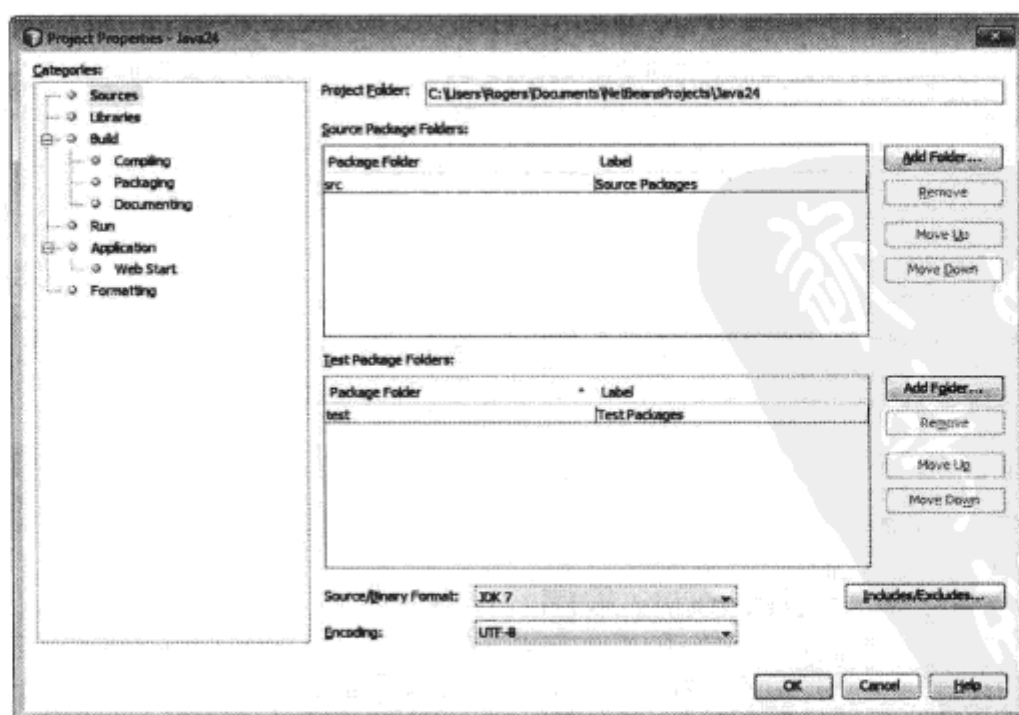
编写完该程序，并在编译和运行之前，你可能会遇到一个错误。因为此时 NetBeans 可能还无法使用 Java 7 中新引入的特性。如果在 `switch` 语句中使用了字符串作为测试条件，源代码编辑器面板中第 7 行代码左边将显示一个红色的警告图标。而且错误消息可能是“strings in switch are not supported”，这表示需要对 NetBeans 进行配置后，才可以对其提供支持。

在 NetBeans 中，Java 7 中的特性需要以每个项目为基础进行启用，其启用步骤如下所示。

1. 在项目（Projects）面板中，右键单击 Java24 项目（或者是你命名的其他项目），然后在弹出菜单中单击属性（Properties），打开项目属性对话框。
2. 在分类（Categories）面板中，单击 Source（如果还没有选定的话），打开源代码属性对话框（见图 7.2）。
3. 在 Source/Binary Format 下拉菜单中，选择 JDK 7 然后单击 OK 按钮。

图 7.2

设置 NetBeans 编辑器，使其支持 Java 7



经过这样设置之后，项目中的所有程序都可以在 Java 7 下运行。

当运行 Commodity 程序时，将产生如下输出：

```
Balance: 530
Quantity: 47
```

7.4 条件运算符

在 Java 中，最复杂的条件语句是三元运算符（?）。

当你想根据条件测试的结果进行赋值或显示时，可以使用三元运算符。例如，在视频游戏中，可能需要根据变量 skillLevel 是否大于 5 来设置变量 numberOfEnemies 的值。为此，一种方法是使用 if-else 语句：

```
if (skillLevel > 5) {
    numberOfEnemies = 10;
} else {
    numberOfEnemies = 5;
}
```

另一种快捷方法是使用三元运算符（?）。三元运算符由 5 部分组成：

- 要测试的条件，用括号括起来，如(skillLevel > 5)；
- 问号（?）；
- 用于判断条件是否为 true 的值；
- 冒号（:）；
- 用于判断条件是否为 false 的值。

要使用三元运算符基于 skillLevel 来设置 numberOfEnemies 的值，可以使用如下语句：

```
int numberOfEnemies = (skillLevel > 5) ? 10 : 5;
```

也可以使用三元运算符来确定要显示的信息。假如要编写这样一个程序：根据变量 gender 的值显示文本“Mr.”或“Ms.”。为此，可以使用三元运算符，如下所示：

```
System.out.print( (gender.equals("male")) ? "Mr." : "Ms." );
```

三元运算符很有用，但也是 Java 中最难以掌握的条件运算符。学习 Java 时，你不会遇到只能使用三元运算符，而不能使用 if-else 语句的情况。

7.5 观察时钟

接下来的程序会为读者展示编程中会用到的每一个条件测试。在这个程序中，将使用 Java 内置的计时功能，它跟踪当前的日期和时间，并将该信息用一句话显示出来。

运行 NetBeans（或其他可以创建 Java 程序的软件），新建一个文件，将其命名为 Clock。这个程序很长，但大部分都是很长的条件语句。在源代码编辑器中输入程序清单 7.3 中的所有文本，完成后将文件保存。

程序清单 7.3 Clock 程序

```
1: import java.util.*;
2:
3: class Clock {
4:     public static void main(String[] arguments) {
5:         // get current time and date
6:         Calendar now = Calendar.getInstance();
7:         int hour = now.get(Calendar.HOUR_OF_DAY);
8:         int minute = now.get(Calendar.MINUTE);
9:         int month = now.get(Calendar.MONTH) + 1;
10:        int day = now.get(Calendar.DAY_OF_MONTH);
11:        int year = now.get(Calendar.YEAR);
12:
13:        // display greeting
14:        if (hour < 12) {
15:            System.out.println("Good morning.\n");
16:        } else if (hour < 17) {
17:            System.out.println("Good afternoon.\n");
18:        } else {
19:            System.out.println("Good evening.\n");
20:        }
21:
22:        // begin time message by showing the minutes
23:        System.out.print("It's");
24:        if (minute != 0) {
25:            System.out.print(" " + minute + " ");
26:            System.out.print( (minute != 1) ? "minutes" :
27:                "minute");
28:            System.out.print(" past");
29:        }
30:
31:        // display the hour
32:        System.out.print(" ");
33:        System.out.print( (hour > 12) ? (hour - 12) : hour );
34:        System.out.print(" o'clock on ");
35:
36:        // display the name of the month
37:        switch (month) {
38:            case 1:
39:                System.out.print("January");
40:                break;
41:            case 2:
42:                System.out.print("February");
43:                break;
44:            case 3:
45:                System.out.print("March");
46:                break;
47:            case 4:
48:                System.out.print("April");
49:                break;
50:            case 5:
51:                System.out.print("May");
52:                break;
53:            case 6:
54:                System.out.print("June");
55:                break;
56:            case 7:
57:                System.out.print("July");
58:                break;
59:            case 8:
```

```

60:         System.out.print("August");
61:         break;
62:     case 9:
63:         System.out.print("September");
64:         break;
65:     case 10:
66:         System.out.print("October");
67:         break;
68:     case 11:
69:         System.out.print("November");
70:         break;
71:     case 12:
72:         System.out.print("December");
73:     }
74:
75:     // display the date and year
76:     System.out.println(" " + day + ", " + year + ".");
77: }
78: }

```

在程序通过编译之后，仔细阅读程序，来理解程序中条件测试是如何进行的。

除了第 1 行和第 6~11 行外，Clock 程序涉及的知识在本章都介绍过。在创建一系列变量用于存储当前日期和时间后，使用一系列 if 和 switch 语句来决定显示什么信息。

该程序包含几条 System.out.println() 语句和 System.out.print() 语句，用于显示字符串。

第 6~11 行使用了名为 now 的 Calendar 变量，类型名 Calendar 的首字母必须大写，这是因为 Calendar 是一种对象。

第 10 章将介绍如何创建和使用对象。这里的重点是第 6~11 行发生了什么，而不是这是如何发生的。

Clock 程序由下面几部分组成。

- 第 1 行让程序能够使用类 java.util.Calendar，它用于跟踪当前的日期和时间。
- 第 3~4 行开始 Clock 程序及其 main() 语句块。
- 第 6 行创建一个名为 now 的 Calendar 对象，该对象包含系统的当前日期和时间。每次运行该程序时，对象 now 的值都不同（当然，除非宇宙的物理定律发生变化或时间停止不前）。
- 第 7~11 行创建变量 hour、month、day 和 year，这些变量的值来自 Calendar 对象，后者是存储所有这些信息的仓库。
- 第 14~20 行显示三个问候语之一：“Good morning.”、“Good afternoon.”和“Good evening.”。显示的问候语取决于变量 hour 的值。
- 第 23~29 行显示当前的分钟值及其他一些文本。首先，第 23 行显示文本“It’s”，如果 minute 的值为 0，则由于第 24 行的 if 语句，第 25~28 行将不执行。第 24 行的语句是必须的，因为在程序中告诉人们现在是几小时零分没有意义。第 25 行显示变量 minute 的当前值，在第 26~27 行使用三元运算符显示文本“minutes”或“minute”，这取决于变量 minute 是否等于 1。最后，在第 28 行显示文本“past”。
- 第 32~34 行使用另外一个三元运算符显示当前的 hour 值。第 33 行的三元运算符条

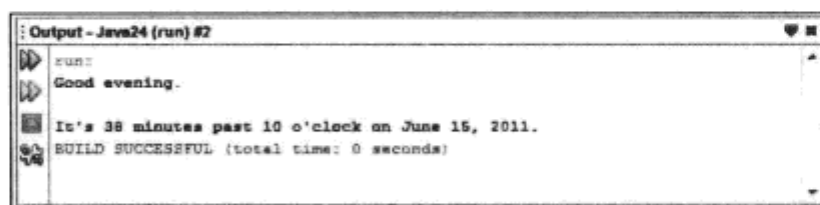
件语句用于在 `hour` 的值大于 12 时以不同的方式显示 `hour` 的值，从而避免计算机显示现在是 15 点这样的情况发生。

- 第 37~73 行几乎占据了程序的一半，这是一个大型的 `switch` 语句，它根据变量 `month` 的值来显示不同的月份名称。
- 第 76 行显示当前的日期和年份。
- 第 77~78 行结束 `main()` 语句块，然后结束整个 `Clock` 程序。

运行该程序时，输出的具体内容随当前日期和时间而异。该程序的输出示例如图 7.3 所示。

图 7.3

Clock 程序的输出



运行该程序多次，看看它如何跟踪时钟变化。

7.6 总结

能够使用条件语句后，你的 Java 编程水平将得到极大的提高。即使程序运行时外界信息发生变化，程序也能够对信息进行评测，并据此做出不同的反应。它们可以根据特定的条件，在两个或多个选项之间做出选择。

编写计算机程序时，必须将任务分成一组需要执行的步骤和需要做出的决策。通过在程序中使用 `if` 语句和其他条件语句，还可能提高逻辑思维能力，这对生活中的其他方面也有帮助：

- 如果他在 11 月份赢得总统选举，我将谋求参与内阁，否则我将移民加拿大；
- 如果我对相亲感到很满意，就在一家豪华餐厅用晚餐，否则将去吃比萨；
- 如果我在试用期犯了错误，则只有费城老鹰队会挑选我。

7.7 问与答

问：`if` 语句好像是最有用的语句之一，有没有这种可能：在程序中只使用 `if` 语句，而不用使用其他语句？

答：这是可能的，不使用 `else` 和 `switch`，很多程序员也从不使用三元运算符（`?`）。然而，在程序中使用 `else` 和 `switch` 是有益的，这样程序更易理解。将一系列 `if` 语句放在一起将使程序难以理解。

问：在 `Clock` 程序中，获取当前月份时，为什么要给 `Calendar.MONTH` 加 1？

答：这是必须的，因为 `Calendar` 类采用怪异的方式表示月份。它不像你预期的那样，使用 1~12 来表示月份，而是用 0 表示 1 月份，用 11 表示 12 月份。加 1 使表示月份的数字更

容易理解。

问：在本章中，将 if 语句同一条语句结合使用时，没有使用左大括号“和右大括号”。不是必须使用这对大括号吗？

答：不是的。大括号用于 if 语句旨在标识根据条件测试结果执行的程序部分。使用大括号是个好习惯，这样可以避免修改程序时犯常见的错误。如果在 if 条件后面添加第二条语句但没有添加大括号，程序运行时将出现错误。

问：是不是在每条 case 语句后面的语句部分都需要使用 break 语句？

答：不是必须这样做。但如果没有在一组语句末尾使用 break，则不管测试出的 case 值是多少，switch 语句块中所有后续语句都将被执行。然而，大多数情况下，你会希望在每一组语句的后面使用 break 语句。

7.8 测验

回答下列问题，以检测对条件语句的掌握程度。

7.8.1 问题

1. 条件测试的结果为 true 或 false，这让你想起了哪种变量？
 - a. 没有。不要没完没了地提问。
 - b. long 变量。
 - c. 布尔型变量。
2. 在 switch 语句块中，哪条语句用于处理其他所有情况？
 - a. default。
 - b. otherwise。
 - c. onTheOtherHand。
3. 什么是条件？
 - a. 洗发后修理零乱并纠缠在一起的头发。
 - b. 在程序中测试条件是 true 还是 false。
 - c. 向邻近的教徒坦白罪过的地方。

7.8.2 答案

1. c. 布尔型变量只能为 true 或 false，它与条件测试类似。
2. a. 如果没有其他 case 语句与 switch 变量匹配，default 语句将被执行。
3. b. 其他两项说的是梳子和忏悔。

7.9 练习

为提高对 Java 条件的认识，通过下面的练习复习本章的主题。

- 在 Clock 程序某行的 `break` 语句前面添加 “//”，使其成为注释，然后编译并运行它，看看将发生什么情况。对更多的 `break` 语句进行相应处理，再看看发生的情况。
- 创建一个小程序，它在一个名为 `grade` 的变量中存储用户从 1~100 之间选择的值。在条件语句中使用 `grade` 变量向 A、B、C、D 和 F 类学生显示不同的消息。先用 `if` 语句完成这项工作，然后再用 `switch` 语句完成。

要查看完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 8 章

使用循环重复执行操作

本章介绍如下内容：

- 使用 for 循环；
- 使用 while 循环；
- 使用 do-while 循环；
- 提早退出循环；
- 为循环命名。

对学生来说，最讨厌的惩罚是在纸上或黑板上一遍又一遍地写东西。在电影 *The Simpsons*（辛普森一家）中，Bart Simpson 得经常到黑板上写 “The art teacher is fat, not pregnant”，这种惩罚对孩子也许有效，但对计算机肯定不管用，因为计算机可以轻松重复一项任务。

计算机程序最适合重复地做相同的事情，因为有循环。循环是在程序中重复执行一条语句或一组语句。有些循环执行固定的次数，有些循环则可以无限期的执行。

Java 中有 3 种循环语句：for、do 和 while。这 3 个循环语句的工作方式相似，但是了解其各自的工作机制仍然大有裨益。通过选择合适的循环语句，可以简化程序的循环部分。

8.1 for 循环

在编程时，你会发现循环可以在多种情况下使用。你可以使用循环重复做某些事情很多次，例如防病毒程序打开每封邮件时检查是否有病毒。也可以使用循环让计算机在某一个简短的周内内什么都不做，比如每隔一分钟显示一次当前时间的动态时钟。

循环语句让计算机程序多次返回到同一个地方，就像飞机特技在空中表演转圈时那样。

Java 中最复杂的循环语句是 for。for 循环经常用于重复执行程序某部分特定次数。下面是一个例子：


```
for (int dex = 0; dex < 1000; dex++) {
    if (dex % 12 == 0) {
        System.out.println("#: " + dex);
    }
}
```

这个循环显示 0~999 之间可被 12 整除的数字。

每个 for 循环都使用一个变量来确定循环何时开始、何时结束。这个变量通常称为计数器，在上述循环中计数器为变量 dex。

这个示例演示了 for 语句的 3 部分。

- 初始化部分：在第一部分，变量 dex 被赋予初始值 0。
- 条件部分：在第二部分，有类似于 if 语句中的条件测试，这里的测试为 dex<1000。
- 修改部分：第三部分是一条语句，它使用递增运算符修改变量 dex 的值。

在初始化部分，可以设置计数器变量，也可以在 for 语句中创建该变量，就像上例中的整型变量 dex 那样。当然也可以在程序的其他地方创建该变量。不管是哪种情况，在 for 语句的这部分都必须给该变量赋初值。循环开始时，变量应该就具备了初始值。

条件部分包含测试，要继续循环，该测试的结果必须为 true。一旦测试结果为 false，循环将结束。在这个例子中，当变量 dex 的值大于或等于 1000 时，循环将结束。

for 语句的最后一部分包含一条修改计数器变量值的语句。每次循环时，这条语句都将执行。计数器变量必须以某种方式改变其值，否则循环永远都不会结束。例如，在这个例子中，在修改部分，使用运算符++将变量 dex 的值加 1。如果变量 dex 的值不变，它将始终为初始值 0，这样条件 dex<1000 将永远为 true。

for 语句中的语句块在每次循环时也都被执行。

在上述例子中，for 语句中的语句块如下：

```
if (dex % 12 == 0) {
    System.out.println("#: " + dex);
}
```

这条语句将执行 1000 次。该循环首先将变量 dex 设置为 0，然后每次循环时将变量 dex 的值加 1，当变量 dex 的值大于或等于 1000 时循环结束。

By the Way

注意：

一个与循环相关的不常见的术语是迭代，指的是执行单词循环，用于控制循环的计数器变量就是迭代。

在 if 语句中已经看到，如果 for 循环只包含一条语句，可以不使用大括号，如下例所示：

```
for (int p = 0; p < 500; p++)
    System.out.println("I will not sell miracle cures");
```

这个循环显示文本 “I will not sell miracle cures” 500 次。虽然循环只有一条语句时可以不使用大括号，但为使程序易于理解，也可以使用括号。

本章创建的第一个程序显示 1~200 的整数与 9 的乘积：即 9×1 、 9×2 、 9×3 等，直到 9×200 。在 NetBeans 中，创建一个新的 Java 空文件，并命名为 Nines，然后输入程序清单 8.1 中的文本。当对文件进行保存时，将存储为 Nines.java。

打开字处理器并新建一个文件，输入程序清单 8.1 所示的内容。输入完成后，将文件保存为 Nines.java。

程序清单 8.1 Nines.java 程序

```
1: class Nines {
2:     public static void main(String[] arguments) {
3:         for (int dex = 1; dex <= 200; dex++) {
4:             int multiple = 9 * dex;
5:             System.out.print(multiple + " ");
6:         }
7:     }
8: }
```

程序 Nines 的第 3 行包含一条 for 语句，该语句有 3 部分。

- 初始化部分 `int dex=1`：它创建一个整型变量 `dex`，并将其初始值设置为 1。
- 条件部分 `dex <= 200`：它必须为真才执行循环，否则将结束循环。
- 修改部分 `dex ++`：在每次循环中将变量 `dex` 的值加 1。

在 NetBeans 中选择命令菜单 `Run->Run File`，运行该程序，将产生如下输出：

Output ▼

```
9 18 27 36 45 54 63 72 81 90 99 108 117 126 135 144 153 162 171
180 189 198 207 216 225 234 243 252 261 270 279 288 297 306 315
324 333 342 351 360 369 378 387 396 405 414 423 432 441 450 459
468 477 486 495 504 513 522 531 540 549 558 567 576 585 594 603
612 621 630 639 648 657 666 675 684 693 702 711 720 729 738 747
756 765 774 783 792 801 810 819 828 837 846 855 864 873 882 891
900 909 918 927 936 945 954 963 972 981 990 999 1008 1017 1026
1035 1044 1053 1062 1071 1080 1089 1098 1107 1116 1125 1134 1143
1152 1161 1170 1179 1188 1197 1206 1215 1224 1233 1242 1251 1260
1269 1278 1287 1296 1305 1314 1323 1332 1341 1350 1359 1368 1377
1386 1395 1404 1413 1422 1431 1440 1449 1458 1467 1476 1485 1494
1503 1512 1521 1530 1539 1548 1557 1566 1575 1584 1593 1602 1611
1620 1629 1638 1647 1656 1665 1674 1683 1692 1701 1710 1719 1728
1737 1746 1755 1764 1773 1782 1791 1800
```

由于 NetBeans 中的输出窗口不会对文本换行，所有所有的数字将在一行中显示。为了使文本能够换行，右键单击输出面板，然后从弹出的菜单中选择“Wrap text”。

8.2 while 循环

while 循环不像 for 循环那样有多个不同的组成部分，它所所要的只是一个条件测试，由 while 语句来完成。下面是一个 while 循环语句的例子：

```
while (gameLives > 0) {
    // the statements inside the loop go here
}
```

该循环将不断重复，直到变量 `gameLives` 小于等于 0。

`while` 语句在循环一开始，即执行循环中的任何语句之前，就测试条件。因此，如果程序首次运行到 `while` 语句时，测试条件为 `false`，循环体中的语句将根本不会执行。

如果 `while` 条件为 `true`，将执行循环一次，然后再测试 `while` 条件。如果在循环体内不改变测试条件，循环将无休止地执行下去。

下面的语句使用 `while` 循环显示同一行文本多次：

```
int limit = 5;
int count = 1;
while (count < limit) {
    System.out.println("Pork is not a verb");
    count++;
}
```

`while` 循环会在循环语句之前设置的一个或多个变量。在这个例子中，创建了两个整型变量：`limit` 和 `count`，其中 `limit` 的值为 5，`count` 的值为 1。

该 `while` 循环显示文本 “Pork is not a verb” 4 次，如果将变量 `count` 的初始值改为 6，将不会显示这行文本。

8.3 do-while 循环

`do-while` 循环的功能类似于 `while` 循环，但测试条件的位置不同，下面是一个 `do-while` 循环的例子：

```
do {
    // the statements inside the loop go here
} while (gameLives > 0);
```

与前面的 `while` 循环类似，该循环不断执行，直到变量 `gameLives` 不再大于 0。`do-while` 循环的不同之处在于，条件测试是在循环体语句之后而不是之前执行的。

当程序首次运行到 `do` 循环时，`do` 和 `while` 之间的语句被自动执行，然后再测试 `while` 条件以决定循环是否继续循环。如果 `while` 条件为 `true`，循环将再次执行；如果 `while` 条件为 `false`，循环结束。在 `do` 和 `while` 语句之间，必须有改变条件的语句，否则循环将一直进行下去。`do-while` 循环体内的语句至少会执行一次。

下列语句使用 `do-while` 循环显示相同的文本行多次：

```
int limit = 5;
int count = 1;
do {
    System.out.println("I will not Xerox my butt");
    count++;
} while (count < limit);
```

与 `while` 循环类似，`do-while` 循环会在循环语句之前使用已经设置的一个或多个变量。

该循环显示文本 “I will not Xerox my butt” 4 次。如果将变量 `count` 的初始值设置为 6，尽管 `count` 大于 `limit`，文本仍将显示 1 次。

在 `do-while` 循环中，即使循环条件为 `false`，循环体内的语句也会被执行一次。

8.4 退出循环

退出循环的正常途径是测试条件为 `false`，3 种类型的 Java 循环（`for`、`while` 和 `do-while`）都是如此。然而有时可能想立即结束循环，即使此时测试条件为 `true`，为此可以使用 `break` 语句，如下面的代码所示：

```
int index = 0;
while (index <= 1000) {
    index = index + 5;
    if (index == 400) {
        break;
    }
}
```

`break` 语句将结束包含该语句的循环体。

在本例中，只有当 `index` 变量大于 1000 时，`while` 循环体才停止执行。然而，可以使用一个特殊的语句提前终止循环的执行：如果 `index` 等于 400，将执行 `break` 语句，从而立即结束循环。

另外一个可在循环中使用的特殊语句是 `continue`，它导致退出当前循环，并进入下一次循环，请看下面的循环：

```
int index = 0;
while (index <= 1000) {
    index = index + 5;
    if (index == 400)
        continue;
    System.out.println("The index is " + index);
}
```

在该循环中，如果变量 `index` 的值不等于 400，语句将正常执行。在本例中，`continue` 语句会令循环从 `while` 语句开始执行，而不是执行 `System.out.println()` 语句。由于 `continue` 语句的存在，循环从来不会显示下面的文本：

```
The index is 400
```

在 3 种循环中，都可以使用 `break` 和 `continue` 语句。

使用 `break` 语句可以在程序中创建一个永远运行的循环，如下例所示：

```
while (true) {
    if (quitKeyPressed == true) {
        break;
    }
}
```

8.5 给循环命名

与 Java 程序中的其他语句类似，可将一个循环放在另一个循环中。下面的示例将一个 `for`

循环放在一个 `while` 循环中：

```
int points = 0;
int target = 100;
while (target <= 100) {
    for (int i = 0; i < target; i++) {
        if (points > 50)
            break;
        points = points + i;
    }
}
```

在这个例子中，如果变量 `points` 大于 50，`break` 语句将导致退出 `for` 循环。然而 `while` 循环永远不会结束。因为变量 `target` 永远不会大于 100。

在有些情况下，你可能想退出两个循环。为此，需要给外层循环（这里是 `while` 循环）命名。要给循环命名，将名称放在循环起始位置的前一行，并在名称后加冒号（:）。

循环有名称后，就可以在 `break` 或 `continue` 语句中使用名称来指出它们将作用于哪个循环。下面的例子与前一个例子相同，但有一点不同：如果变量 `points` 大于 50，将结束两个循环：

```
int points = 0;
int target = 100;
targetLoop:
while (target <= 100) {
    for (int i = 0; i < target; i++) {
        if (points > 50)
            break targetLoop;
        points = points + i;
    }
}
```

当在 `break` 或 `continue` 语句中使用循环名称时，不要加上名称后面的冒号。

复杂的 for 循环

`for` 循环可以相当复杂，可以在初始化部分、条件测试部分，以及循环体部分有多个变量。`for` 循环的各个部分可以使用分号（;）进行隔离，而且在初始化部分可以设置多个变量，循环体中也可以有多条语句，如下面的例子所示。

```
int i, j;
for (i = 0, j = 0; i * j < 1000; i++, j += 2) {
    System.out.println(i + " * " + j + " = " + (i * j));
}
```

在 `for` 循环的同一个部分中，变量之间用逗号隔开，如 `i=0, j=0`。这个循环要显示一系列 `i` 乘以 `j` 的等式。在每次循环中，变量 `i` 加 1，变量 `j` 加 2。一旦 `i` 与 `j` 的乘积大于等于 1000，循环将结束。

`for` 循环的组成部分也可为空，一个这样的例子是，在程序的其他地方已经创建计数器变

量并设置了其初始值，如下例所示：

```
for ( ; displayCount < endValue; displayCount++) {
    // loop statements would be here
}
```

8.6 测试计算机的运行速度

这里将会用到一个运行计算机标准检查程序（Benchmark）的 Java 程序，它可以测试计算机的软件或硬件的运行速度。Benchmark 程序使用循环语句来重复执行下面的数学表达式：

```
double x = Math.sqrt(index);
```

该语句调用 `Math.sqrt()` 方法来查找数值的平方根。第 11 章将会讲解该方法的工作机制。

这里创建的 Benchmark 程序可以查看计算机在一分钟之内计算平方根的次数。

在 NetBeans 中创建一个新的 Java 空文件，然后命名为 Benchmark。输入程序清单 8.2 中的文本，然后保存文件。

程序清单 8.2 Benchmark.java 的完整源代码

```
1: class Benchmark {
2:     public static void main(String[] arguments) {
3:         long startTime = System.currentTimeMillis();
4:         long endTime = startTime + 60000;
5:         long index = 0;
6:         while (true) {
7:             double x = Math.sqrt(index);
8:             long now = System.currentTimeMillis();
9:             if (now > endTime) {
10:                 break;
11:             }
12:             index++;
13:         }
14:         System.out.println(index + " loops in one minute.");
15:     }
16: }
```

该程序在运行时将会进行如下操作。

- 第 1~2 行：声明 Benchmark 类，开始程序的 main（）块。
- 第 3 行：创建变量 startTime，并用当前时间（毫秒）为其赋初始值。该初始值可以使用 Java 的 System 类中的 currentTimeMillis（）方法来获取。
- 第 4 行：创建 endTime 变量，其初始值为 startTime 加 60000。由于 1 分钟等于 60000 毫秒，因此 endTime 与 startTime 正好间隔 1 分钟。
- 第 5 行：创建一个长整型变量 index，且其初始值为 0。
- 第 6 行：while 语句使用 true 作为测试条件，开始循环。由于测试条件始终为 true，

因此循环将一直运行（也就是说，只有其他事情将其终止时，才停止循环）。

- 第 7 行：计算 `index` 的平方根，然后将其存储到 `x` 变量中。
- 第 8 行：创建变量 `now`，并使用 `currentTimeMillis()` 方法将当前时间赋值给它。
- 第 9~11 行：如果 `now` 大于 `endTime`，这表明循环已经运行了 1 分钟，此时 `break` 语句会结束 `while` 循环。反之，继续执行循环。
- 第 12 行：每循环一次，`index` 变量加 1。
- 第 14 行：程序在循环体外显示它进行平方根计算的次数。

图 8.1 所示为该程序的输出结果。

图 8.1

Benchmark 程序的输出



Benchmark 程序非常适合用于测试读者的计算机是否比我的快。在测试时，我的计算机总共执行了 45 亿次计算，如果你的机器的执行次数比这个值要高，不要只是安慰我，多买点我的书，这样我好对自己的计算机进行升级。

8.7 总结

循环是大多数编程语言的基础。通过循环语句对将要显示的图形依次进行控制，即可创建动画效果。如果没有循环，则在 Java 以及其他编程语言中都无法完成这样的任务。

8.8 问与答

问：本章在多个地方都使用了术语“初始化”，这是什么意思？

答：意思是给变量设置初始值。创建一个变量并将一个字符串赋给它时，你就是在对变量进行初始化。

问：如果循环永不结束，如何让程序停止运行？

答：在程序中，如果有循环永不结束，通常采用其他方式使其停止。例如，在游戏中，循环可能不断运行，直到玩家丧失性命为止。

编写程序时一种常见的 bug 是无限循环：由于编程错误，循环永不停止。如果运行 Java 程序时发生了无限循环错误，可以按下输出面板中左侧出现的红色警告图标。

8.9 测验

下列问题用于测试读者对循环的理解程度。为贯彻该主题的精神，不断回答这些问题，直到充分理解为止。

8.9.1 问题

1. 应使用什么分隔开 `for` 语句的各个部分？
 - a. 逗号。
 - b. 分号。
 - c. 不当班的警察。
2. 哪条语句导致程序回到循环的开头，并从那里继续运行？
 - a. `conitnue`。
 - b. `next`。
 - c. `skip`。
3. Java 中的哪一个循环语句至少可以运行一次？
 - a. `for`。
 - b. `while`。
 - c. `do-while`。

8.9.2 答案

1. b. 逗号用于分隔同一部分内的内容，而分号用于分隔不同的部分。
2. a. `break` 语句用于彻底结束循环，而 `continue` 语句用于进入下一轮循环。
3. c. 只有在执行过一次循环之后，才测试 `do-while` 语句中的条件。

8.10 练习

如果学习有关循环的知识后，你的头脑还清醒，请完成下面的练习以复习本章的主题：

- 修改 `Benchmark` 程序，对乘法或除法这些简单数学运算的执行进行测试。
- 使用循环编写一个小程序，找出前 400 个能被 13 整除的数。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 9 章

使用数组存储信息

本章介绍如下内容：

- 创建数组；
- 设置数组的大小；
- 为数组元素赋值；
- 修改数组中的信息；
- 创建多维数组；
- 数组排序。

在计算机的发展历程之中，圣诞老人从中获得的便利要比所有人都多。几个世纪以来，人们要求他收集并处理大量的信息。年事已高的圣诞老人还必须记录下列信息：

- 淘气的孩子；
- 好孩子；
- 需要的礼物；
- 带无通路烟囱的家；
- 希望圣诞老人（而不是圣诞老人的老婆）满足她们愿望的妇女；
- 先对其坐骑拍照留念，然后再提需求的国家。

在北极，计算机是最实用的东西，它非常适合用于对信息进行存储、分类和研究。

在计算机程序中存储信息的最基本方式是，将它放在变量中。淘气孩子的名单就是一组类似的信息。可以使用数组来记录这种名单。

数组是一组类型相关的相关变量，可以在数组中存储任何类型的信息，就像在变量中存储信息一样。与单个变量相比，数组可记录更复杂的信息，但创建和使用数组同变量一样简单。

9.1 创建数组

数组是名称相同的一组变量，读者可能熟悉术语“数组”，读者应该很熟悉数组——想象一个销售人员展示的一系列产品，或者是具有一系列令人眼花的奖品的游戏。与变量一样，创建数组也要指出存放在数组中的变量类型以及数组名，不同之处在于多加了一对方括号：“[”和“]”。

像变量一样，可以创建存放任何类型信息的数组。例如，下面的语句创建一个字符串数组：

```
String[] naughtyChild;
```

下面是另外两个例子：

```
int[] reindeerWeight;
boolean[] hostileAirTravelNations;
```

注意：

在创建数组时，Java 在方括号的位置方面比较灵活，可以将方括号放在变量名后面，而不是放在变量类型的后面，如下所示：

```
String niceChild[];
```

为了使程序中的数组更容易理解，应在程序中统一使用一种格式，而不是两种格式都用。在本书的示例中，都是将方括号放在变量类型的后面。

By the Way

前面的例子创建了数组，但是其中没有存储任何值。为此，可以使用包含变量类型名的 `new` 语句，或者将初始值放在大括号“{”和“}”之间。当使用 `new` 关键字时，还必须指定数组包含多少项，数组中的项被称为元素。下面的语句创建一个数组，并为其将存储的值预留空间：

```
int[] elfSeniority = new int[250];
```

该例子创建了一个名为 `elfSeniority` 的整型数组，该数组包含 250 个元素，这些元素用于存储圣诞老人的每个淘气孩子到北极的月份。如果圣诞老人运营一个联盟商店，记录这个信息将很重要。

使用 `new` 语句创建数组时，必须指定元素的个数。而且数组中的每个元素都将赋给初始值，初始值取决于数组的类型。对于所有数值型数组，初始值为 0，字符型数组的初始值为 ‘\0’，布尔型数组的初始值为 `false`，字符串数组和所有其他对象数组的初始值为 `null`。

对于不是非常大的数组，可以在创建数组时指定初始值。下面的例子创建一个字符串数组并指定初始值：

```
String[] reindeerNames = { "Dasher", "Dancer", "Prancer", "Vixen",
    "Comet", "Cupid", "Donder", "Blitzen" };
```

要存储到数组元素中的信息放在“{”和“}”之间，之间用逗号隔开。数组中元素的个数也就是用逗号隔开的元素数。数组中的每个元素必须有相同的类型，上述例子用字符串定

义每种驯鹿的名称。

在数组创建之后，就不能增大其空间，进而增加其他的元素。即使又想起了一种最著名的驯鹿的名称，也不能将 Rudolph 作为第 9 个元素加入到数组 `reindeerNames` 中，Java 编译器不允许这样做。

9.2 使用数组

在程序中使用数组就像使用变量一样，只是需要在靠近数组名的方括号内指定元素编号。你可以在允许使用变量的任何地方使用数组元素。下面的语句使用的都是本章前面定义的数组：

```
elfSeniority[193] += 1;
niceChild[9428] = "Eli";
if (hostileAirTravelNations[currentNation] == true) {
    sendGiftByMail();
}
```

数组的第 1 个元素的编号为 0，而不是 1。这意味着最大的元素编号比你想象的小 1。请看下面的语句：

```
String[] topGifts = new String[10];
```

这条语句创建一个字符串数组，其元素编号为 0~9，如果在程序的其他地方使用 `topGifts[10]`，将会得到一个与 `ArrayIndexOutOfBoundsException` 相关的错误消息。

在 Java 程序中，异常是错误的另外一个名称。这里的异常是一个“数组越界”错误，这表示程序试图使用一个预定义边界之外的数组元素。第 18 章将详细讲解异常。

如果你想检测数组的上界，以避免在引用数组元素时超越上界，可以使用 `length` 变量，它与数组紧密相关。`length` 是一个整型变量，包含数组能容纳的元素数。下面的例子创建一个数组，然后报告其长度：

```
String[] reindeerNames = { "Dasher", "Dancer", "Prancer", "Vixen",
    "Comet", "Cupid", "Donder", "Blitzen", "Rudolph" };
System.out.println("There are " + reindeerNames.length + " reindeer.");
```

在这个例子中，`reindeerNames.length` 的值为 9，也就是说可以指定的最大元素编号为 8。

在 Java 中有两种处理文本的主要方式：字符串和字符数组。使用字符串时，一种有用的技巧是将字符串中的每个字符放在字符数组的一个元素中。为此，可使用字符串的 `toCharArray()` 方法，它生成一个字符数组，该数组包含的元素数与字符串长度相同。

本章的第一个程序使用了本节介绍的两种方法。程序 `SpaceRemover` 将显示一个字符串，并将其中所有的空格字符（' '）替换为句点字符（'.'）。

在 NetBeans 中打开 Java24 项目，然后选择 File->New File，创建一个新的空 Java 文件，将其命名为 `SpaceRemover`。然后在源代码编辑器窗口中输入程序清单 9.1 中的文本，并保存。

程序清单 9.1 SpaceRemover.java 程序

```

1: class SpaceRemover {
2:     public static void main(String[] args) {
3:         String mostFamous = "Rudolph the Red-Nosed Reindeer";
4:         char[] mfl = mostFamous.toCharArray();
5:         for (int dex = 0; dex < mfl.length; dex++) {
6:             char current = mfl[dex];
7:             if (current != ' ') {
8:                 System.out.print(current);
9:             } else {
10:                 System.out.print('.');
11:             }
12:         }
13:         System.out.println();
14:     }
15: }

```

在 NetBeans 中选择 Run->Run File 命令，运行该程序来查看其输出，如图 9.1 所示。

应用程序 SpaceRemover 将文本“Rudolph the Red - Nosed Reindeer”存储在两个地方：一个是字符串变量 mostFamous，另一个是字符数组 mfl。数组是在第 4 行通过调用 mostFamous 的 toCharArray() 方法创建的，该方法将文本中的每个字符存储到数组的一个元素中，字符 R 存储在元素 0 中，字符 u 在元素 1 中，字符 d 在元素 2 中，依此类推，最后将字符 r 存储到元素 29 中。

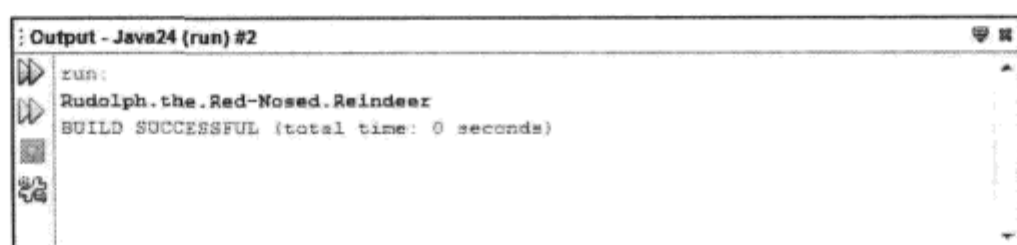


图 9.1

SpaceRemover
程序的输出

第 5~12 行的 for 循环检查数组 mfl 中的每个字符，如果字符不是空格，就直接显示它，如果是空格，就显示句点字符 (.)。

9.3 多维数组

目前为止，本章介绍的都是一维数组，可以使用 0 到数组元素数之间的数字检索元素。有些类型的信息需要使用多维数组来存储，如 (x, y) 坐标系的点，其中一维用于存储 x 坐标，另一维用于存储 y 坐标。

要创建二维数组，必须在创建和使用数组时多加一对方括号。请看下面的例子：

```

boolean[][] selectedPoint = new boolean[50][50];
selectedPoint[4][13] = true;
selectedPoint[7][6] = true;
selectedPoint[11][22] = true;

```

这个例子创建了一个名为 selectedPoint 的布尔数组，该数组的第一维有 50 个元素，第二维也有 50 个元素，因此总共有 2500 (50×50) 个元素。该数组创建后，每个元素的默认值为 false。接下来，将其中的 3 个元素设置为 true，它们在数组中的位置分别是 (4, 13)、(7, 6) 和 (11, 22)。

根据需要，数组可以有任意多维数，但别忘了，如果数组的维数很多，将占用大量的内

存。创建一个 50×50 的数组相当于创建 2500 个变量。

9.4 对数组进行排序

将一系列类似的数据组织为数组后，你可以重新安排它们的序列。下面的语句交换整型数组 `numbers` 中两个元素的值：

```
int temp = numbers[7];
numbers[5] = numbers[6];
numbers[6] = temp;
```

这些语句导致 `number[5]` 和 `number[6]` 的值相互交换，整型变量 `temp` 在交换时用作临时存储空间。“排序”是指将一组相关内容按指定顺序排列，一个例子是将数字从小到大排列。

圣诞老人可以按照姓氏对接收礼物的人进行重新排序。例如，Willie Aames 和 Hank Aaron 会先于 Dweezil Zappa 和 Jim Zorn 收到礼物。

在 Java 中对数组排序很容易，因为 `Arrays` 类提供了这种功能。`Arrays` 类位于 `java.util` 中，它可以对任何类型（包括字符串）的数组进行排序。

要在程序中使用 `Arrays` 类，可执行下列步骤：

1. 使用 `import java.util.*` 语句将所有 `java.util` 类导入到程序中；
2. 创建数组；
3. 使用 `Arrays` 类的方法 `sort()` 来对数组重新排序。

使用 `Arrays` 类的 `sort()` 方法对数组进行排序后，其中的值将按数字升序排列。字符和字符串将按字母顺序排列。

为了对其验证，创建一个新的空 Java 文件，名 `Name`，然后在源代码编辑器中输入程序清单 9.2 中的所有文本。这是一个按照名字进行排序的段程序。

程序清单 9.2 `Name.java` 的完整源代码

```
1: import java.util.*;
2:
3: class Name {
4:     public static void main(String[] args) {
5:         String names[] = { "Lauren", "Audrina", "Heidi", "Whitney",
6:             "Stephanie", "Spencer", "Lisa", "Brody", "Frankie",
7:             "Holly", "Jordan", "Brian", "Jason" };
8:         System.out.println("The original order:");
9:         for (int i = 0; i < names.length; i++) {
10:             System.out.print(i + ": " + names[i] + " ");
11:         }
12:         Arrays.sort(names);
13:         System.out.println("\nThe new order:");
14:         for (int i = 0; i < names.length; i++) {
15:             System.out.print(i + ": " + names[i] + " ");
16:         }
17:         System.out.println();
18:     }
19: }
```

当运行该 Java 程序时，它首先按照原来的顺序显示这 13 个名字，按照名字进行排序后，然后再重新显示名字，其输出结果如下。

Output ▼

```
The original order:
0: Lauren 1: Audrina 2: Heidi 3: Whitney 4: Stephanie 5: Spencer
6: Lisa 7: Brody 8: Frankie 9: Holly 10: Jordan 11: Brian
12: Jason
The new order:
0: Audrina 1: Brian 2: Brody 3: Frankie 4: Heidi 5: Holly
6: Jason 7: Jordan 8: Lauren 9: Lisa 10: Spencer 11: Stephanie 12:
Whitney
```

如果使用字符串或基本类型（如整数和浮点）变量时，通过 Arrays 类的方法只可按升序进行排序。如果要按其他顺序排列或希望排序效率比 Arrays 类高，可以自己编写代码来实现。

9.5 对字符串中的字符计数

在英文中，经常出现的字母依次是 E、R、S、T、L、N、M 和 O，如果你之前看过辛迪加游戏“财富转轮”，就会意识到这一点。

注意：

如果读者不熟悉“财富转轮”这个节目，这里介绍一下：财富转轮游戏有 3 个参赛者，他们猜测组成一个短语、名字或引文的字母。如果猜中一个字母且是辅音字母，就可以通过转动大轮子来确定赢多少钱。为增加娱乐性，参加者与在观众席前排就坐的朋友一起来玩这个游戏，猜中字母时，就发给他们随机数量的钱，并给胜利者一个新的猜测机会。

By the Way

这里将要创建的程序用来统计字母在不同短语或表达中出现的频率，并使用数组来存放每个字母出现的字数。当程序运行时，它将显示每个字母在短语中出现的次数。

在 NetBeans 中创建一个新的 Java 空文件，将其命名为 Wheel.java，然后输入程序清单 9.3 中的所有文本，最输入完毕之后保存。你可以在第 17 行和第 18 行之间随意输入其他短语，只要格式与第 17 行相同即可。

程序清单 9.3 Wheel.java 的完整源代码

```
1: class Wheel {
2:     public static void main(String[] args) {
3:         String phrase[] = {
4:             "A STITCH IN TIME SAVES NINE",
5:             "DON'T EAT YELLOW SNOW",
6:             "JUST DO IT",
7:             "EVERY GOOD BOY DOES FINE",
8:             "I WANT MY MTV",
9:             "I LIKE IKE",
10:            "PLAY IT AGAIN, SAM",
11:            "FROSTY THE SNOWMAN",
12:            "ONE MORE FOR THE ROAD",
13:            "HOME FIELD ADVANTAGE",
14:            "VALENTINE'S DAY MASSACRE",
15:            "GROVER CLEVELAND OHIO",
```

```

16:         "SPAGHETTI WESTERN",
17:         "AQUA TEEN HUNGER FORCE",
18:         "IT'S A WONDERFUL LIFE"
19:     };
20:     int[] letterCount = new int[26];
21:     for (int count = 0; count < phrase.length; count++) {
22:         String current = phrase[count];
23:         char[] letters = current.toCharArray();
24:         for (int count2 = 0; count2 < letters.length; count2++) {
25:             char lett = letters[count2];
26:             if ( (lett >= 'A') & (lett <= 'Z') ) {
27:                 letterCount[lett - 'A']++;
28:             }
29:         }
30:     }
31:     for (char count = 'A'; count <= 'Z'; count++) {
32:         System.out.print(count + ": " +
33:             letterCount[count - 'A'] +
34:             " ");
35:     }
36:     System.out.println();
37: }
38: }

```

如果你没有添加自己的短语，该程序的输出如程序清单 9.4 所示。

程序清单 9.4 Wheel 程序的输出结果

```

A: 22 B: 3 C: 5 D: 13 E: 28 F: 6 G: 5 H: 8 I: 18
J: 1 K: 0 L: 13 M: 10 N: 19 O: 27 P: 3 Q: 0 R: 13
S: 15 T: 19 U: 4 V: 7 W: 9 X: 0 Y: 10 Z: 0

```

Wheel 程序中将会发生如下事情。

- 第 3~19 行：短语存储在字符串数组 `phrase` 中。
- 第 20 行：创建整型数组 `letterCount`，它包含 26 个元素。该数组用来存储每个字母（依次为 A~Z）出现的次数。元素 `letterCount[0]` 存储字母 A 出现的次数，元素 `letterCount[1]` 存储字母 B 出现的次数，依此类推，最后，元素 `letterCount[25]` 存储字母 Z 出现的次数。
- 第 21 行：开始一个 for 循环，该循环遍历数组 `phrase` 中的所有短语。该 for 语句使用了变量 `phrase.length`，以便达到最后一个短语时结束循环。
- 第 22 行：创建一个名为 `current` 的字符串变量，并将数组 `phrase` 中当前元素的值赋给它。
- 第 23 行：创建一个字符型数组，用于存储当前短语中的所有字符。
- 第 24 行：开始一个 for 循环，该循环遍历当前短语中的所有字符。这里使用了变量 `letters.length`，以确保达到最后一个字符时结束循环。
- 第 25 行：创建字符变量 `lett` 并将当前字符赋给它。字符除文本值外，还有对应的数值。由于数组中的元素可以编号，因此可以使用每个字符对应的数值来确定其元素号。
- 第 26~28 行：使用 if 语句排除所有非字母字符，如标点符号和空格。依据当前存储在变量 `lett` 中的字符对应的数值，决定将数组 `letterCount` 中的哪个元素加 1。字母对

应的数值从 65（代表 ‘A’）到 90（代表 ‘Z’）。由于数组 `letterCount` 的元素编号为 0~25，因此为确定将哪个数组元素加 1，将变量 `lett` 与 ‘A’ 相减。

- 第 31 行：使用 `for` 循环从字母 ‘A’ 遍历到字母 ‘Z’。
- 第 32~34 行：显示当前字母、冒号，以及该字母在数组 `phrase` 中存储的短语中出现的次数。

注意：

与字符 ‘A’ 到 ‘Z’ 相关联的数值是用于 ASCII 字符集中的值。ASCII 字符集是 Unicode 的一部分，后者是 Java 语言支持的完整字符集。Unicode 字符集支持全世界的各种书面语言中使用的 60000 多个字符。而 ASCII 只有 256 个字符。

By the Way

该程序演示了如何使用两个嵌套的 `for` 循环，以每次一个字符的方式遍历一组短语。Java 给每个字符提供了一个相关联的数值，这个值比数组中的字符更易使用。

9.6 总结

通过使用数组，可以在程序中存储和处理复杂的信息。数组也适合存储列表信息，并可使用第 8 章介绍的循环语句轻松地进行存取。

说实话，圣诞老人的信息处理需求可能超过了数组的处理能力。每年都有孩子出生，他们要求的礼物越来越复杂和昂贵。

使用变量不方便时，可以在程序中使用数组来存储信息，即使没有创建列表或对其进行复核，也可以使用数组。

9.7 问与答

问：字母的数值范围 65（‘A’）到 90（‘Z’）是基本 Java 语言的一部分吗？如果是，1~64 留作什么用处？

答：数值 1~64 对应于数字、标点符号和其他可打印字符（如回车、换行符和退格）。在 Java 程序中，可以使用与可打印字符及一些不可打印字符相关联的数值。Java 使用 Unicode 字符集，其中前 127 个字符来自 ASCII 字符集，读者在其他编程语言可能使用过。

问：为什么有些错误称为异常？

答：该术语的含义是程序正常运行时没有问题，而异常表明必须应对特殊情况。异常是 Java 程序发出的警告消息，在 Java 语言中，术语“错误（error）”有时只用于描述在运行程序的解释器中发生的错误状态。第 18 章将更详细地介绍这两个术语。

问：在多维数组中，可以使用变量 `length` 来测量除第一维外的其他维的长度吗？

答：可以测量数组任何维的长度，对于第一维，可使用数组名和 `length`，如 `x.length`；对于其他维，可以使用该维的第 1 个元素和 `length`。请看使用下面的语句创建的数组 `data`：

```
int[][][] data = new int[12][13][14];
```

该数组第一维的长度可使用 `data.length` 来确定；对于第二维，可以通过 `data[0].length` 来测量；对于第三维，可以使用 `data[0][0].length` 来测量。

9.8 测验

如果读者的脑袋是数组，可以通过回答下列关于数组的问题来测量其长度。

9.8.1 问题

1. 数组最适合用于存放什么类型的信息？
 - a. 列表。
 - b. 一对相关的信息。
 - c. 琐碎的东西。
2. 什么变量用于检查数组的上界？
 - a. `top`。
 - b. `length`。
 - c. `limit`。
3. 包括 Rudolph 在内，圣诞老人有多少只驯鹿？
 - a. 8。
 - b. 9。
 - c. 10。

9.8.2 答案

1. a. 列表包含相同类型的信息，如字符串、数字等，适合用数组存储。
2. b. 变量 `length` 包含数组的元素数。
3. b. Clement Clark Moore 在其著作的圣诞诗歌《A Visit from St. Nicholas》中提到，圣诞老人有 8 只小驯鹿，所以加上 Rudolph 后是 9 只。

9.9 练习

要获得更多可供以后使用的经验，可通过下面的练习拓展有关本章主题的知识：

- 创建一个程序，它使用多维数组存储学生的成绩。第一维是学生编号，第二维是每个学生的成绩。显示全部学生的平均成绩以及每个学生的平均成绩。
- 编写一个程序，将能被 13 整除的前 400 个数存储到数组中。

要查看完成这些练习编写出的 Java 程序，请访问本书的配套网站 www.java.24hours.com。

第 10 章

创建第一个对象

本章介绍如下内容：

- 创建对象；
- 使用属性描述对象；
- 确定对象的行为；
- 合并对象；
- 从其他对象继承；
- 转换对象和其他类型的信息。

在本书中，面向对象编程（OOP）是比较难理解的专业术语之一。这个复杂的术语以优雅的方式描述了计算机程序是什么以及他是如何工作的。

在面向对象编程之前，计算机程序是使用本书前面介绍过的最简单的定义来描述的：它是文件中的一组指令，这些指令按某种可靠的顺序执行。

如果将程序视为对象的集合，便可以确定程序要完成的任务，然后将这些任务指派给最适合完成它们的对象。

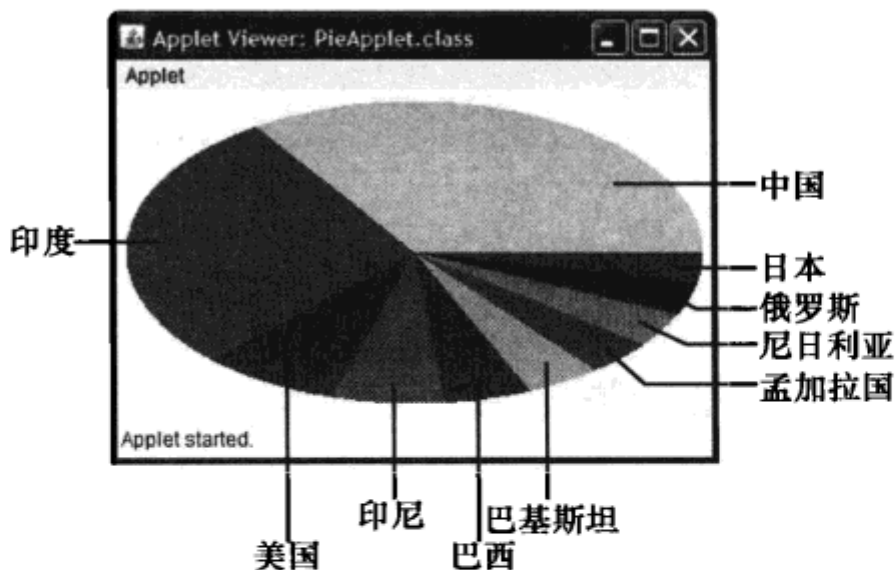
10.1 面向对象编程的工作原理

用 Java 创建的程序可被视为一个对象，就像真实世界中存在的物体一样。对象独立于其他对象而存在，以特定方式同其他对象交互，可以与其他对象合并成更大的事务。如果将计算机程序视为一组彼此交互的对象，设计出的程序将更可靠，更容易理解，更容易在其他项目中重用。

在第 23 章，读者将创建一个显示饼图的 Java 程序，饼图是一个圆，使用不同颜色的扇形区表示数据（见图 10.1）。饼图是一个由更小的对象（具有不同颜色的扇形区域、指出每个扇形区代表什么的图例以及标题）组成的对象。

图 10.1

显示饼图的 Java
程序



每个对象都有区别于其他对象的特征。饼图是圆的，而柱状图则使用一系列矩形来表示数据。如果你用划分饼图的方式划分计算机程序，也就是在进行面向对象编程（OOP）。在面向对象编程中，对象包含两项内容：属性和行为。属性描述对象并使其不同于其他对象，而行为指的是对象能做什么。

在 Java 中，创建对象时使用类作为模板，“类”是对象的母版，它可以决定对象应有哪些属性和行为。读者对术语“类”应该不陌生，因为每个 Java 程序都被称为类。使用 Java 创建的每个程序都是类，你可以将它用作创建新对象的模板。例如，任何使用字符串的 Java 程序都使用了根据 String 类创建的对象。String 类包含属性和行为，前者决定了 String 对象是什么样的，而后者控制 String 对象能做什么。

在面向对象编程中，计算机程序是一组对象，这些对象协同工作以完成某项任务。有些简单的程序看似由一个对象（类文件）组成，但即使是这样的程序也使用了其他对象来完成其工作。

10.2 对象示例

在显示饼图的程序中，PieChart 对象可能包含以下内容：

- 计算饼图中每个扇形区应多大的行为；
- 绘制饼图的行为；
- 存储饼图标题的属性。

对读者来说，让 PieChart 来绘制它自己可能会有点奇怪，因为现实世界中图形无法绘制自身。但在面向对象编程中，对象尽可能独立地完成工作。这种特征使对象更有用，因为可以将其用于其他程序，而不用教它如何做。如果 PieChart 对象不知道如何绘制自己，则每当在其他程序中使用 PieChart 对象时，你都必须创建绘制它的行为。

面向对象编程的另一个例子是自动拨号器。在电影《War Games》中，Matthew Broderick 扮演的角色使用它寻找可入侵的计算机。

**By the
Way**

注意：

自动拨号器是一个软件，它使用调制解调器依次拨打一系列电话号码。这种程序旨在找到做出应答的计算机，以便以后拨打这些电话号码，看看是哪里的电话。

当前，使用自动拨号器肯定会引起本地电话公司的关注，甚至带来法律纠纷。但在 1980 年代，几乎不用出门就能做到。David Lightman (Broderick 扮演的角色) 使用自动拨号器搜索一家视频游戏公司的私有计算机系统，以便在该公司发布前就能玩其新游戏。结果 Lightman 找到了一台秘密的政府计算机，在该计算机上能够玩从国际象棋到《Global Therm-nuclear War》在内的所有游戏。

和其他任何计算机程序一样，自动拨号器可以视为一组协同工作的对象。它可能有以下几部分组成。

- **Modem 对象**：它具有自己的属性（比如速度）以及行为。比如，该对象能够让调制解调器拨出一个号码，并检测到另外一个计算机系统响应了该呼叫。
- **Monitor 对象**：对所呼叫的号码和成功的呼叫进行记录。

每个对象都独立于其他对象存在。

设计完全独立的 Modem 对象的一个优点是，可以将其用在需要调制解调器功能的其他程序中。

使用自包含对象的另一个原因是更容易调试。计算机程序的规模越来越大，如果你在调试 Modem 等对象时，你知道它不依赖于其他任何东西，则只需将精力放在确保 Modem 对象完成其预期工作，并存储完成其工作所需的信息即可。

将 Java 等面向对象编程语言作为学习的第一种编程语言是有好处的，因为这样不必改正原有的编程习惯。

10.3 什么是对象

对象是通过将对象类作为模板来创建的。下面的语句将创建一个类。

```
public class Modem {
}
```

根据这个类创建的对象毫无用处，因为它没有任何属性和行为。只有为其添加了属性和行为后，这个类才具有可用性。可以像下面这样为这个类添加属性和行为。

```
public class Modem {
    int speed;

    public void displaySpeed() {
        System.out.println("Speed: " + speed);
    }
}
```

这个 Modem 类看起来应该很像本书前面编写的程序。Modem 类以 class 语句打头，只是 class 左边还有关键字 public。关键字 public 表示这个类是公有的，换句话说，任何程序都可以使用 Modem 对象。

Modem 类的第一部分创建一个整型变量 speed，该变量是对象的一个属性。

Modem 类的第二部分是一个名为 displaySpeed() 的方法。该方法是对对象的行为的一部分，它包含一条语句：System.out.println()，用于显示调制解调器的速度值。

对象的变量通常称为实例变量或成员变量。

如果要在程序中使用 Modem 对象，可以使用如下语句来创建它。

```
Modem device = new Modem();
```

这条语句创建一个名为 device 的 Modem 对象。在创建了对象之后，可以设置其变量并调用其方法。要设置 device 对象的 speed 变量的值，可使用下面的语句：

```
device.speed = 28800;
```

通过调用 displaySpeed() 方法可以让这个调制解调器显示其速度，如下面的代码所示：

```
device.displaySpeed();
```

名为 device 的 Modem 对象将通过显示文本 “Speed: 28800” 来响应这条语句。

10.4 理解继承

OOP 的一个最大优点是继承，它允许一个对象继承另外一个对象的行为和属性。

当你开始创建对象时，有时会发现将要创建的新对象和你以往开发出的对象有很多相似之处。

当《War Games》在 1983 年上映之时，如果 David Lightman 想要一个能够处理纠错且具有其他高级调制解调器特性的对象，而这些特性在当时还没有，那么他应该怎么办呢？Lightman 可以通过复制 Modem 对象的语句然后对其修改的方式，来创建一个新的 ErrorCorrectionModem 对象。然而，如果 ErrorCorrectionModem 对象的大部分属性和行为与 Modem 对象相同，则上述工作完全没有必要。这也意味着如果日后要进行修改，Lightman 需要升级两个独立的程序。

通过继承，程序员只需定义新类与现有类的不同之处，就能够创建一个新类。Lightman 可以让 ErrorCorrectionModem 类继承 Modem 类，这样只需编写纠错调制解调器不同于以前调制解调器的部分。

要继承其他类，可使用 extend 语句，下面是从 Modem 类继承的 ErrorCorrectionModem 类的框架：

```
public class ErrorCorrectionModem extends Modem {  
    // program goes here  
}
```

10.5 建立继承层次

通过继承，无需做大量重复的工作就可以开发大量相关的类，它使得代码可以从一个类传递给另一个类，再传递给其他类。类这种组织结构称为类层次，在 Java 程序中使用的标准类都属于同一个类层次。

如果知道子类和超类是什么，将更容易理解类层次。从其他类继承而来的类称为子类，被继承的类称为超类。

在前面的《War Games》示例中，Modem 类是 ErrorCorrectionModem 类的超类；而 Error

CorrectionModem 是 Modem 的子类。

在层次结构中，可以从一个类派生出多个类：ISDNModem 可能是 Modem 的另外一个子类，因为 ISDN Modem 包含使其不同于纠错调制解调器的属性和行为。如果 ErrorCorrection Modem 有子类 IntenalErrorCorrectionModem，后者将继承类层次结构中位于它上面的所有类，包括 ErrorCorrectionModem 和 Modem。这些继承关系如图 10.2 所示。

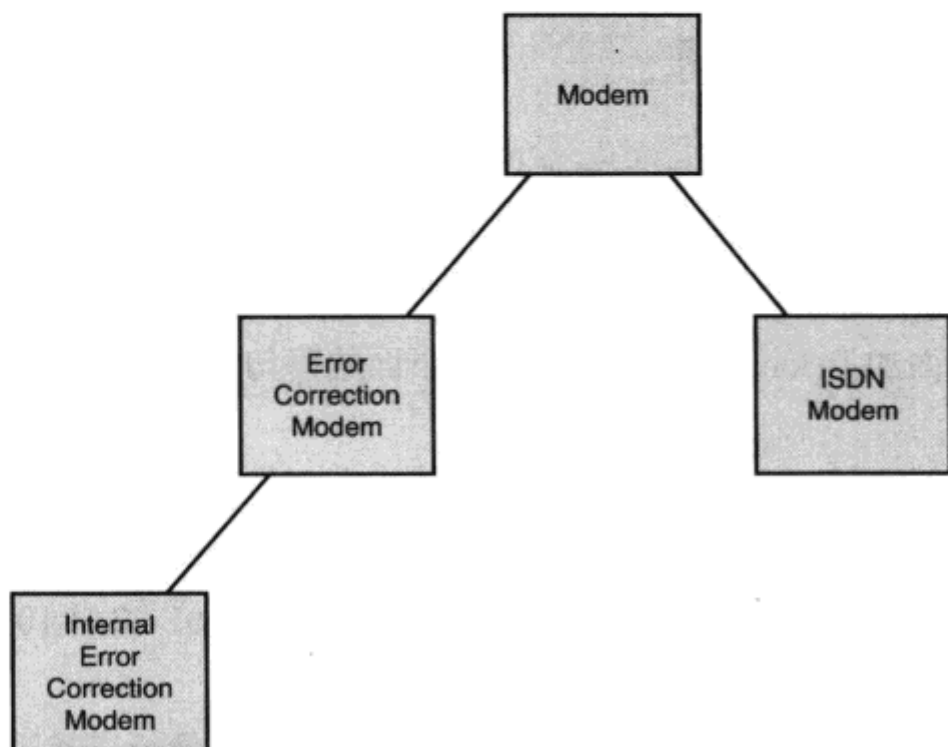


图 10.2

类层次示例

组成标准 Java 语言的类充分利用了继承，了解这一点很有必要。第 12 章将详细地介绍继承。

10.6 转换对象和简单变量

在 Java 中需要完成的一种最常见的任务是，将信息从一种格式转换为另一种格式。可以执行多种转换，如下所示：

- 将一个对象转换为另一个对象；
- 将简单变量转换为另一种类型的变量；
- 使用对象创建简单变量；
- 使用简单变量创建对象。

简单变量的类型为第 5 章介绍的基本数据类型，包括 int、float、char、long 和 double。

在程序中使用方法或表达式时，必须使用方法或表达式所要求的信息类型。例如，如果需要 Calendar 对象的方法必须接收 Calendar 对象。如果使用的方法接受单个整型参数，但传递的却是浮点数，在编译程序时将报错。

注意：

诸如 System.out.println() 这样的方法需要字符串参数时，你可以使用 + 运算符将参数中不同类型的信息合并起来。只要待合并中的信息有一个是字符串，则合并后的参数将被转换为字符串。

By the Way

在程序中将信息转换为另一种类型称为类型转换。类型转换得到的信息的类型与原来的变量或对象不同。进行类型转换时，并不会修改原来的变量或对象的值，而是创建一个所需类型的新变量或对象。

讨论类型转换时，术语“源”和“目标”很有帮助。“源”指的是原始格式的信息（无论是变量还是对象），而“目标”是转换后得到的新版本。

10.6.1 简单变量的类型转换

对于简单变量，最常见的是在数值类型之间进行类型转换，如整数和浮点数之间。有一种类型的变量不能进行类型转换，这就是布尔值。

要将信息转换为其他类型，可以在变量前指定新类型，并将其用括号括起。例如，要将变量转换为 long 变量，可在前面加上(long)。下面的语句将下面的语句将 float 值转换为 int 值：

```
float source = 7.06F;  
int destination = (int) source;
```

在类型转换中，如果目标类型的取值范围比源类型大，转换将很容易，如将 byte 值转换为 int 值。byte 的取值范围为-128~127，而 int 的取值范围为 $-2.1 \times 10^9 \sim 2.1 \times 10^9$ 。无论 byte 变量存储的什么值，新的 int 变量都有足够的空间存储它。

有时不用进行类型转换就可使用类型不同的变量，例如，char 变量可直接用作 int 变量，int 变量可直接用作 long 变量，而任何数值变量都可直接用作 double 变量。

在大多数情况下，由于目标类型的取值范围大于源类型，转换时不会修改信息。特例是将 int 或 long 变量转换为 float 变量，以及将 long 变量转换为 double 变量。

要从取值范围大的类型转换为取值范围小的类型，必须显式地进行转换，如下面的语句所示：

```
int xNum = 103;  
byte val = (byte) xNum;
```

这里将一个名为 xNum 的 int 变量转换为一个名为 val 的 byte 变量。在这里，目标变量的取值范围比源类型小，byte 变量能够存储-128~127 的整数，而 int 变量的取值范围大得多。

如果在转换操作中，源变量的值在目标变量中放不下，Java 将修改源值，以便能够存储。如果不想改变值，这将导致意想不到的结果。

10.6.2 对象类型转换

只要源对象和目标对象存在继承关系，就可以在它们之间进行类型转换，其中一个类必须是另一个类的子类。

有些对象根本不需要转换，可在任何需要超类的地方使用子类对象。Java 中的所有对象都是 Object 类的子类，因此可在任何需要 Object 的地方使用任何对象。

也可以将对象用于需要其子类的地方，然而，由于子类通常比其超类包含更多的信息，

因此可能缺少某些信息。如果对象没有其子类的方法，而程序使用了该方法，这将导致错误。

要将对象用于需要其子类的地方，必须使用类似于下面的语句对其进行显式的类型转换：

```
Graphics comp = new Graphics();
Graphics2D comp2D = (Graphics2D) comp;
```

这将 `Graphics` 对象 `comp` 转换为一个 `Graphics2D` 对象。转换过程中不会丢失任何信息，但将得到子类定义的所有方法和变量。

10.6.3 在简单变量和对象之间进行转换

不能在对象和简单变量之间进行类型转换。在 Java 中，每一个简单的变量类型都有很多类，比如 `Boolean`、`Byte`、`Character`、`Double`、`Float`、`Integer`、`Long` 和 `Short` 等。这些类名的首字母都是大写的，而不是简单变量类型。

使用这些类中定义的方法，可以将变量的值作为参数来创建一个对象。下面的语句使用 5309 创建一个 `Integer` 对象：

```
Integer suffix = new Integer(5309);
```

用这种方式创建对象后，可以像使用其他对象那样使用该对象。需要将该值作为简单变量使用时，也可以使用相应的类方法。例如，要通过前面的 `suffix` 对象得到一个 `int` 值，可使用下面的语句：

```
int newSuffix = suffix.intValue();
```

这条语句将变量 `newSuff` 的值设置为 5309，该变量的类型为 `int`。最常见的在对象和变量之间进行的转换是将字符串作为数值使用。为此，可以使用 `Integer` 类的 `parseInt()` 方法，如下例所示：

```
String count = "25";
int myCount = Integer.parseInt(count);
```

该语句将包含文本 25 的字符串转换为整型值 25。如果字符串的值不是有效的证书，则不进行转换。

接下来将创建一个应用程序，它将命令行参数中的字符串值转换为数值，这是通过命令行从用户那里获得输入时常用的一种技巧。

在 NetBeans 中打开 Java24 项目，选择 File->New File，创建一个新的 Java 空文件，并命名为 `NewRoot`。在源代码编辑器中输入程序清单 10.1 中的内容，然后保存。

程序清单 10.1 `NewRoot.java` 的全部代码

```
1: class NewRoot {
2:     public static void main(String[] args) {
3:         int number = 100;
4:         if (args.length > 0) {
5:             number = Integer.parseInt(args[0]);
```



```

6:      }
7:      System.out.println("The square root of "
8:          + number
9:          + " is "
10:         + Math.sqrt(number) );
11:    }
12: }

```

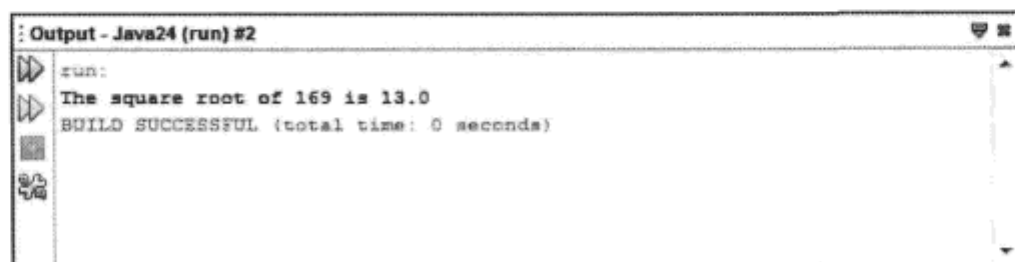
在运行该程序之前，必须对 NetBean 进行配置，使其能够运行命令行参数。选择菜单命令 **Run->Set Project Configuration->Customize**，打开 **Project Properties** 对话框。将主类命名为 **NewRoot**，并在参数字段中输入 169。单击 **OK** 按钮关闭对话框。

选择 **Run->Run Main Project**（而不是 **Run File**）来运行该程序。该程序将显示数值 169 以及该数值的平方根，如图 10.3 所示。

应用程序 **NewRoot** 是第 4 章示例程序的扩展，后者显示整数 225 的平方根。

图 10.3

NewRoot 程序的
输出



如果该程序能够接受用户提交的整数并显示其平方根，将更有用。这就需要将字符串转换为整数。每个命令行参数都被存储在 **String** 数组的元素中，因此在数学表达式中使用它们之前，必须将其转换为数字。

为了根据字符串的内容创建一个整数值，可调用 **Integer.parseInt()** 方法，并将字符串作为唯一的参数，就像第 5 行那样：

```
number = Integer.parseInt(args[0]);
```

由于 **args[0]** 存储的是程序运行时用户提供的第一个参数，因此当以“169”作为参数来运行该程序时，字符串“169”被转换为整数 169。

10.6.4 自动封装和拆封

在 Java 中，每一种基本的数据类型都有对应的类：**boolean** 对应 **Boolean** 类，**byte** 对应 **Byte** 类，**char** 对应 **Character** 类，**double** 对应 **Double** 类，**float** 对应 **Float** 类，**int** 对应 **Integer** 类，**long** 对应 **Long** 类，**short** 对应 **Short** 类。

在每种基本类型及其对应的类中，可存储的值是相同的，唯一的差别是信息的格式。例如，整数值 413 可以用 **int** 变量表示，也可以用 **Integer** 对象表示。

Java 的自动封装和拆封功能使得可以互换地使用基本数据类型及其对应的对象格式。

自动封装功能将简单变量值转换为相对应的类。

拆封功能将对象转换为相对应的简单变量值。

这些功能在幕后工作，确保在需要基本数据类型（比如 **float**）时，将对象转换为该类型且值保持不变。反之亦然，即必要时自动将基本数据类型转换为相应的对象。

下面的语句显示的是自动封装和拆封是如何使用的。

```
Float total = new Float(1.3F);
float sum = total / 5;
```

在 Java 的早期版本中 (Java 1.5 之前), 这将会报错, 原因是不能在第二条语句中使用 Float 对象。Java 现在可以拆封 total 对象以让该语句运行, 从而 sum 的最终结果是 0.26。

10.7 创建对象

下面来看一个关于类和继承的例子, 为此需要创建代表两种对象类型的类: 一种是电缆调制解调器, 对应的类为 CableModem; 另一种是 DSL 调制解调器, 对应的类为 DslModem 这里将重点放在这些对象的简单属性和行为上:

- 每个对象都有速度 (speed), 并可以显示出来;
- 每个对象都应能够连接到 Internet。

电缆调制解调器和 DSL 调制解调器的一个共同点是, 它们都有速度。因为这是它们共有的, 可将其放在 CableModem 和 DslModem 的超类 Modem 中。在 NetBeans 中创建一个新的 Java 空类, 将其命名为 Modem, 然后在源代码编辑器中输入程序清单 10.2 中的文本, 并保存。

程序清单 10.2 Modem.java 的完整源代码

```
1: public class Modem {
2:     int speed;
3:
4:     public void displaySpeed() {
5:         System.out.println("Speed: " + speed);
6:     }
7: }
```

该文件将被自动编译为 Modem.class。该程序不能直接运行, 但是可以在其他类中使用它。这个 Modem 类可以处理 CableModem 和 DslModem 类共有的行为。创建类 CableModem 和 DslModem 时, 通过使用 extends 语句可以使其都成为 Modem 的子类。

在 NetBeans 中创建一个新的 Java 空文件, 将其命名为 CableModem。输入程序清单 10.3 中的文本, 然后进行保存。

程序清单 10.3 CableModem.java 的完整源代码

```
1: public class CableModem extends Modem {
2:     String method = "cable connection";
3:
4:     public void connect() {
5:         System.out.println("Connecting to the Internet ...");
6:         System.out.println("Using a " + method);
7:     }
8: }
```

在 NetBeans 中创建第 3 个文件, 将其命名为 DslModem。输入程序清单 10.4 中的文本, 然后保存。

程序清单 10.4 DslModem.java 的完整源代码

```

1: public class DslModem extends Modem {
2:     String method = "DSL phone connection";
3:
4:     public void connect() {
5:         System.out.println("Connecting to the Internet ...");
6:         System.out.println("Using a " + method);
7:     }
8: }

```

如果此时没有错误，将会得到 3 个类文件：Modem.class、CableModem.class 和 DslModem.class。然而，此时不能运行这 3 个类文件的任何一个，因为它们都没有 main() 块。需要创建一个小程序来测试刚才建立的类层次。

返回 NetBeans，然后创建一个新的 Java 空文件，将其命名为 ModemTester。在源代码编辑器中输入程序清单 10.5 中的文本，然后保存。

程序清单 10.5 ModemTester.java 的完整源代码

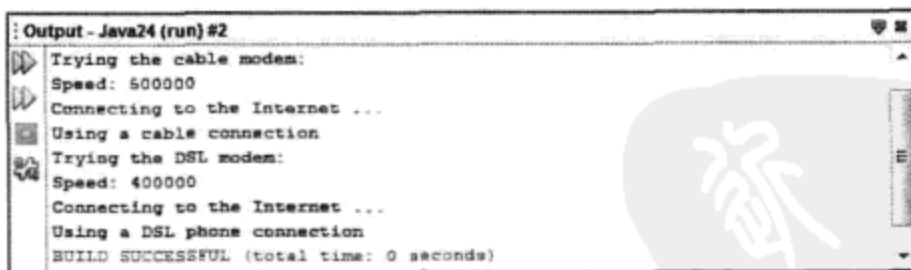
```

1: public class ModemTester {
2:     public static void main(String[] args) {
3:         CableModem surfBoard = new CableModem();
4:         DslModem gateway = new DslModem();
5:         surfBoard.speed = 500000;
6:         gateway.speed = 400000;
7:         System.out.println("Trying the cable modem:");
8:         surfBoard.displaySpeed();
9:         surfBoard.connect();
10:        System.out.println("Trying the DSL modem:");
11:        gateway.displaySpeed();
12:        gateway.connect();
13:    }
14: }

```

当运行该程序时，就会看到图 10.4 所示的输出。

图 10.4
ModemTester 程序
的输出



下面是对该程序的详细解释。

- 第 3~4 行：创建两个新对象，一个是名为 surfBoard 的 CableModem 对象，另一个是名为 gateway 的 DslModem 对象。
- 第 5 行：将 CableModem 对象 surfBoard 的 speed 变量设置为 500000。
- 第 6 行：将 DslModem 对象 gateway 的 speed 变量设置为 400000。
- 第 8 行：调用 surfBoard 对象的 displaySpeed() 方法，该方法是从 Modem 类继承的，虽然 CableModem 类中没有这个方法，但也可以调用它。

- 第 9 行：调用 `surfBoard` 对象的 `connect()` 方法。
- 第 11 行：调用 `gateway` 对象的 `displaySpeed()` 方法。
- 第 12 行：调用 `gateway` 对象的 `connect()` 方法。

10.8 总结

创建第一个对象类并建立包含几个类的层次结构后，读者应该对术语“面向对象编程”有更深入的理解。在接下来的两章中，读者将开始创建更复杂的对象，学到更多有关对象行为和属性的知识。

有了更多面向对象编程的经验后，读者将更深入地理解术语“程序”、“类”和“对象”的含义。面向对象编程需要花段时间才能习惯，但一旦掌握它，将发现它是一种设计、开发和调试计算机程序的高效方式。

10.9 问与答

问：一个类可以继承多个类吗？

答：这在有些编程语言中（比如 C++）是可以的，但在 Java 中不行。多继承是一项功能强大的特性，但也使得面向对象编程更难于学习和使用。Java 的开发人员决定对继承进行限制，即任何类只能有一个超类，虽然同一个类可以有多个子类。一种补偿这种限制的方式是，可以从接口这种特殊类继承方法。第 19 章将更详细地介绍接口。

问：什么时候应创建非 `public` 的方法？

答：当你不想让方法被其他程序使用时，就需要对该方法进行限制，使其仅用于你编写的程序。如果创建了一款游戏程序，而且你编写的 `shootRayGun()` 方法仅适合用于该游戏，可将其声明为 `private` 的。要将方法声明为 `private` 的，可省略方法名前面的 `public`。

问：为什么可将 `char` 值当做 `int` 值来使用？

答：由于每一个字符在字符集中都有一个可以表示其位置的数值代码，因此可将字符值当做 `int` 值来使用。如果有一个变量 `k`，其值为 67，则对 `(char) k` 进行转换时，则会生成字符值 'C'，原因是在 ASCII 字符集中，与大写字母 C 对应的数值代码是 67。ASCII 字符集是 Unicode 字符标准的一个子集，其中后者由 Java 语言采用。

10.10 测验

回答下面的问题，以测试读者对有关对象及使用对象的程序的理解程度。

10.10.1 问题

1. 什么语句让一个类能够继承另一个类？
 - a. `inherits`。

- b. `extends`。
 - c. `handitOverAndNobodyGetsHurt`。
2. 为什么编译后的 Java 程序的文件扩展名为 `.class`?
- a. Java 的开发者认为这是一种优等 (`classy`) 语言。
 - b. 这是对全球教师的一种巧妙赞扬。
 - c. 任何 Java 程序都是一个类。
3. 对象由哪两项内容组成?
- a. 属性和行为。
 - b. 命令和数据文件。
 - c. 唾液和醋。

10.10.2 答案

- 1. b. 之所以使用 `extends` 语句, 是因为子类是对超类及其超类的属性和行为的扩展。
- 2. c. 程序至少由一个主类以及所需的其他类组成。
- 3. a. 从某种意义上说, b 也对, 因为命令相当于行为, 而数据文件相当于属性。

10.11 练习

如果读者不反对 (`object`), 可通过下面的练习拓展 (`extends`) 有关本章主题的知识:

- 创建一个 `AcousticModem` 类, 其速度为 300, 且有 `connect()` 方法。
- 在有关调制解调器的项目中, 在某个类中添加一个 `disconnect()` 方法, 让各种调制解调器 (电缆调制解调器、DSL 调制解调器、声学调制解调器) 能够断开连接。

有关为完成这些练习而编写的 Java 程序, 请访问本书的配套网站 www.java24hours.com。



第 11 章

描述对象

本章介绍如下内容：

- 为类或对象创建变量；
- 使用对象和类的方法；
- 调用方法并返回一个值；
- 创建构造函数；
- 给方法传递参数；
- 使用 `this` 来引用对象；
- 创建新对象。

上一章在介绍面向对象编程时讲到，对象是一种组织程序的方式，以便拥有完成任务所需的一切。对象由属性和行为组成。

属性是存储在对象中的信息，可以是整型、字符型或布尔变量，也可以是其他对象，如 `String` 对象和 `Calendar` 对象。行为是对象中用于处理特定任务的一组语句。每组语句称为一个方法。

到目前为止，读者在无意识的情况下使用了对对象的方法和变量。如果语句中包含句点，但它不是小数点或字符串的一部分，则很可能使用了对对象。

11.1 创建变量

本章将介绍一个名为 `Virus` 的类对象，它活着的唯一目标就是尽可能地繁殖，有点像我在大学时认识的人。为完成其工作，`Virus` 需要执行多种操作，这些都被实现为类的行为。方法需要的信息将存储在属性中。

对象的属性提供了对象为完成其任务所需的变量。这些变量可以为简单数据类型，如整数、字符、浮点数，也可以为数组或类对象，如 `String` 和 `Calendar`。对象的变量可以在对象所包含的任何一个方法中使用。按照惯例，在创建类语句之后，在创建方法之前，需要创建变量。

`Virus` 对象需要一种方式指出文件已感染病毒。有些计算机病毒修改这样的字段，即存储文件最后修改时间的字段，例如，有些病毒将时间从 13:41:20 改为 13:41:61。由于正常情况下，文件不可能在第 61 秒修改，因此该时间表明文件感染了病毒。`Virus` 对象在整型变量 `newSeconds` 中使用了 86 这个数值，由于秒数的最大值为 60，因此在这个数值是不正确的。

下面的语句创建一个名为 `Virus` 的类，它有一个名为 `newSeconds` 的属性以及其他两个属性：

```
public class Virus {
    public int newSeconds = 86;
    public String author = "Sam Snett";
    int maxFileSize = 30000;
}
```

这 3 个变量都是类的属性，它们是 `newSeconds`、`maxFileSize` 和 `author`。

在变量声明语句中指定 `public` 等关键字称为访问控制，因为它决定了其他类对象将如何使用该变量或它们是否可以使用这些变量。

将变量声明为 `public` 后，使用 `Virus` 对象的其他程序也可以来修改该变量的值。

例如，如果另外一个程序为数值 92 赋予了特别重要的意义，则它可以将 `newSeconds` 变量的值改为 92。下面的语句。下面的语句创建一个名为 `influenza` 的 `Virus` 对象，并设置其变量 `newSeconds`：

```
Virus influenza = new Virus();
influenza.newSeconds = 92;
```

在 `Virus` 类中，变量 `author` 也是 `public` 的，因此也可以在其他程序中修改它的值。另一个变量 `masFileSize` 只能在其所属的类中使用。

将类中的变量声明为 `public` 时，该类将无法控制其他程序使用该变量的方式。在很多情况下，这可能不是问题。例如，可将 `author` 变量该为任何表示病毒作者的名字或假名。如果制造病毒的作者被起诉，则他的名字可能会出现在法庭文件中，所以不要用一个很傻的名字。

通过限制变量的访问权限，可避免因其他程序设置错误的值而导致错误。就 `Virus` 类而言，如果 `newSeconds` 被设置为 60 或更小，就不能成为报告文件是否感染病毒的可靠方式。因为有些文件无论是否感染病毒，都可能在那个秒数被保存。要避免 `Virus` 类出现这样的问题，需要做下面的两项工作：

- 将变量从 `public` 该为 `protected` 或 `private`，后两者提供了更严格的访问限制；
- 添加修改变量值的行为以及向其他程序报告变量值的行为。

`protected` 变量只能在其所在的类、该类的子类以及同一个包（`package`）的其他类中使用。包是一组用于完成相同目标的相关类。例如，`java.util` 包包含很多实用功能类，如日期和时间编程以及文件归档。通过在 Java 程序中使用 `import` 和 `*`，如 `import java.util.*`，就可以在程序

中使用包中的所有类。

`private` 变量的访问限制比 `protected` 变量更严格，只能在其所属的类中使用。除非对变量做任何修改都不会影响类的功能，否则应将变量声明为 `private` 或 `protected`。

下面的语句将变量 `newSeconds` 声明为 `private` 的：

```
private int newSeconds = 86;
```

如果要让其他程序能够使用变量 `newSeconds`，必须创建行为，这将在本章后面介绍。

还有一种访问控制类型：创建变量时不指定 `public`、`private` 或 `protected`。

在本章之前创建的程序中，就没有指定上述任何关键字。没有指定访问控制类型时，变量只能在同一个包的类中使用。这被称为默认访问控制类型或包访问控制类型。

11.2 创建类变量

创建对象时，它将拥有相应类中所有变量的版本。每个 `Virus` 对象都有自己的 `newSeconds`、`maxFileSize` 和 `author` 变量。如果修改对象的变量，将不会影响其他 `Virus` 对象中的同一个变量。

有时属性与整个类而不是特定对象相关联，它们称之为类变量。例如，如果要跟踪在程序中使用了多少个 `Virus` 对象，则可以使用一个类变量来存储这种信息。而且整个类只有该变量的一个拷贝。前面为对象创建的变量称为“对象变量”，因为它们与具体对象相关联。

这两种变量的创建方法和使用方法相同，但是创建类变量时指定使用关键字 `static`。下面的语句为 `Virus` 类创建一个类变量：

```
static int virusCount = 0;
```

修改类变量的方法与修改对象变量完全相同，如果有一个名为 `tuberculosis` 的 `Virus` 对象，可以使用下面的语句来修改类变量 `virusCount`：

```
tuberculosis.virusCount++;
```

由于类变量用于整个类而不是特定对象，因此可以直接使用类名：

```
Virus.virusCount++;
```

这两条语句完成相同的工作，但处理类变量时使用类名有个优点，即表明 `virusCount` 是个类变量而不是对象变量。如果引用类变量时使用对象名，在不仔细查看类的源代码的情况下，将无法确定是类变量还是对象变量。

类变量也称为静态变量。

警告：

尽管类变量很有用，但是也不要过度使用它。因为这些变量在类运行之时就会一直存在。如果类变量中存储的是一个大型的对象数组，则会占据很大的一块内存，而且不会将其释放掉。

**Watch
Out!**

PDG

11.3 用方法来创建行为

属性用于记录有关对象的信息，但是要让类实现它的目的，必须创建行为。行为描述了类中完成特定任务的不同部分，每部分都称为方法。

读者虽然没有意识到，但在前面的程序中一直在使用方法，尤其是 `println()`，它在屏幕上显示文本。和变量一样，方法也是通过对象或类来使用的：在对象名或类名后跟句点和方法名，如 `screen2D.drawString()` 和 `Integer.parseInt()`。

By the Way

注意：

方法 `System.out.println()` 有点令人困惑，因为其中包含两个句点。这是因为其中涉及两个类：`System` 和 `PrintStream`。`System` 类有一个名为 `out` 的变量，后者是一个 `PrintStream` 对象，而 `println()` 是 `PrintStream` 类的一个方法。因此，`System.out.println()` 语句的意思是，使用 `System` 类的 `out` 变量的 `println()` 方法。可以使用这种方法串接对变量和方法的引用。

11.3.1 声明方法

创建方法的语句与创建类的语句有点类似，它们都可以在名称后面的括号中指定参数，都使用大括号“{”和“}”指示开始和结束。不同之处在于，方法可以在执行完毕后返回一个值，返回值可以是简单类型（如整数或布尔值），也可以是对象。

下面是一个 `Virus` 类可用于感染文件的方法：

```
public boolean infectFile(String filename) {
    boolean success = false;
    // file-infecting statements go here
    return success;
}
```

该方法接受一个参数——名为 `filename` 的字符串变量，该变量表示要攻击的文件。如果感染成功，`success` 变量的值将被设置为 `true`。

在方法开头的语句中，方法名 `infectFile` 前有 `boolean`。该关键字指出方法执行完毕后将返回一个布尔值。实际用于返回值的是 `return` 语句，这里返回的是变量 `success` 的值。

如果方法不应该返回值，则方法名前面使用 `void` 关键字。

当方法返回值时，可以将其用于表达式中。例如，如果创建了 `Virus` 对象 `malaria`，可以使用下面这样的语句：

```
if (malaria.infectFile(currentFile)) {
    System.out.println(currentFile + " has been infected!");
} else {
    System.out.println("Curses! Foiled again!");
}
```

返回值的方法可用于程序中任何可以使用变量的地方。

在本章前面，为防止其他程序读取或修改变量 `newSeconds`，将其声明为 `private`。但是也可以通过其他方式在别处使用 `newSeconds` 变量：在 `Virus` 类中创建读写 `newSeconds` 变量的 `public`

方法。不同于变量 `newSeconds` 本身，这些新方法应该是 `public` 的，这样其他程序才能调用它们。

请看下面两个方法：

```
public int getSeconds() {
    return newSeconds;
}

public void setSeconds(int newValue) {
    if (newValue > 60) {
        newSeconds = newValue;
    }
}
```

这些方法称为 `accessor` 方法，原因是它们可以允许 `newSeconds` 变量被其他对象访问。

`getSeconds()` 方法用于返回 `newSeconds` 变量的当前值，它没有任何参数，但是在方法名后面还需要要有一对括号。`setSeconds()` 方法接受一个参数：整型变量 `newValue`。该参数是 `newSeconds` 的新值。如果 `newValue` 大于 60，则进行修改。

在这个例子中，`Virus` 类控制其他类如何使用 `newSeconds`，这被称为“封装”，它是面向对象编程的基本概念。对象防止自己被误用的能力越强，在其他程序中使用它时越有用。

尽管 `newSeconds` 变量是 `private`，但是新方法 `getSeconds()` 和 `setSeconds()` 能够处理 `newSeconds` 变量，原因是它们位于同一个类中。

11.3.2 参数不同的类似方法

正如读者在 `setSeconds()` 方法中看到的，可以向方法传递参数来影响其行为。在类中，不同的方法名称不同，但如果接受不同的参数，方法也可以同名。

如果两个方法接受的参数数量不同或参数类型不同，它们可以同名。例如，对于 `Virus` 对象，有两个 `tauntUser()` 方法可能很有帮助。其中一个方法不接受任何参数，它发出通用的嘲笑 (`taunt`)，另一个接受指定嘲笑字符串的参数。下面的语句实现了这两种方法：

```
void tauntUser() {
    System.out.println("That has gotta hurt!");
}

void tauntUser(String taunt) {
    System.out.println(taunt);
}
```

这两个方法具有相同的名字，但是参数不同：一个方法没有参数，另外一个方法有一个 `String` 参数。传递给方法的参数称为方法签名 (`signature`)。在每一个方法具有不同签名的前提下，类可以具有多个名字相同的不同方法。

11.3.3 构造函数

在程序中创建对象时，使用关键字 `new`，如下所示：

```
Virus typhoid = new Virus();
```

这条语句创建一个名为 `typhoid` 的 `Virus` 对象。使用关键字 `new` 时，将调用类的一个特殊

方法，该方法称为构造函数（**constructor**），因为它处理创建对象所需做的工作。构造函数用于设置对象正常工作所需的变量及方法。

定义构造函数的方式与其他方法类似，只是它不能像其他方法那样返回值。下面是 **Virus** 类的两个构造函数：

```
public Virus() {
    String author = "Ignoto";
    int maxFileSize = 30000;
}

public Virus(String name, int size) {
    author = name;
    maxFileSize = size;
}
```

与其他方法类似，通过使用不同的参数，可以在同一个类中定义多个构造函数。在这里，像下面那样使用关键字 **new** 来创建对象时，将调用第一个构造函数：

```
Virus mumps = new Virus();
```

仅当在 **new** 语句传递一个字符串参数和一个整型参数时，才会调用另一个构造函数，如下所示：

```
Virus rubella = new Virus("April Mayhem", 60000);
```

如果在类中不包含任何构造函数，将从超类那里继承一个不接受参数的构造函数。也可能继承其他构造函数，这取决于使用的超类。

使用 **new** 语句创建对象时，类中必须有一个这样的构造函数，即其参数数量和类型与 **new** 语句中指定的完全相同。在 **Virus** 类中，有两个构造函数：**Virus()** 和 **Virus(String name, int size)**，所以只能使用两种类型的 **new** 语句创建 **Virus** 对象：一种没有参数，另一种将一个字符串和一个整数作为参数。

11.3.4 类方法

与类变量相似，类方法提供与整个类而不是特定对象相关联的方法。当方法不影响单个对象时将其声明为类方法。一个这样的例子是前一章使用的 **Integer** 类的 **parseInt()** 方法，该方法用于将字符串转换为整型变量，如下所示：

```
int fontSize = Integer.parseInt(fontText);
```

这是一个类方法。要将方法声明为类方法，在方法名前使用关键字 **static**，如下所示：

```
static void showVirusCount() {
    System.out.println("There are " + virusCount + " viruses");
}
```

在本章前面，使用类变量 **virusCount** 来记录程序创建了多少个 **Virus** 对象。方法 **showVirusCount()** 是一个类方法，它显示创建的对象总数，可使用下面这样的语句来调用它：

```
Virus.showVirusCount();
```

11.3.5 方法中变量的作用域

当在类的方法中创建变量或对象时，它们只在该方法内可用，原因在于变量的作用域。作用域是变量可用的程序部分，离开作用域定义的程序部分，就不能使用该变量。

程序中的大括号“{”和“}”定义变量的范围。在大括号内创建的变量不能在大括号外使用，例如，请看下面的语句：

```
if (numFiles < 1) {
    String warning = "No files remaining.";
}
System.out.println(warning);
```

这个例子不能正常运行（在 NetBeans 中不能被编译），因为 `warning` 变量是在 `if` 块句的大括号之间创建的，这对括号定义了该变量的作用域。在这对大括号外，变量 `warning` 不存在，因此 `System.out.println()` 方法不能将它用作参数。

在一对大括号内又使用另一对大括号时，要注意使用的变量的作用域，请看下面的例子：

```
if (infectedFiles < 5) {
    int status = 1;
    if (infectedFiles < 1) {
        boolean firstVirus = true;
        status = 0;
    } else {
        firstVirus = false;
    }
}
```

看到问题了么？在这个例子中，`status` 变量可以在任何地方使用，但是为 `firstVirus` 变量赋值的语句将导致编译错误，因为 `firstVirus` 变量是在 `if (infectedFiles < 1)` 语句内创建的，在后面的 `else` 语句中不存在。

要修复这种问题，必须在这两个语句块外创建 `firstVirus` 变量，这样其作用域将包含这两个语句块。一种解决办法是在创建 `status` 变量之后创建 `firstVirus` 变量。

作用域规则使程序更容易调试，因为作用域限制了变量的使用区域。这避免了在其他编程语言中最常见的一种错误：在程序的不同部分以不同的方式使用相同的变量。

作用域的概念也适用于方法，因为它们是用左大括号（{）和右大括号（}）定义的。在一个方法中创建的变量不能在其他方法中使用。如果变量是作为对象变量或类变量创建的，才可以在多个方法中使用。

11.4 将一个类放在另一个类中

虽然有时候 Java 程序也称为类，但很多时候一个程序需要多个类才能完成其工作。包含多个类的程序有一个主类和任意数目的助手类组成。

将程序分成多个类时，有两种方法可用于定义助手类。一种是独立地定义每个类，如下例所示：


```

public class Wrecker {
    String author = "Ignoto";

    public void infectFile() {
        VirusCode vic = new VirusCode(1024);
    }
}

class VirusCode {
    int vSize;

    VirusCode(int size) {
        vSize = size;
    }
}

```

在本例中，VirusCode 类被用作 Wrecker 类的助手类。通常在同一个.java 源文件中定义助手类，因为它们是辅助类。源文件编译后，将生成多个类文件。上面的例子在编译时将生成文件 Wrecker.class 和 VirusCode.class。

Watch Out!

警告：

如果在同一个源文件中定义多个类，只能有一个类为 public，在其他类的 class 语句中不能指定 public。另外，源文件的名称应与它定义的 public 类的名称匹配。

创建一个主类和一个助手类时，也可以将助手类放在主类中。在这种情况下，助手类称为内部类。

内部类放在另一个类的左大括号（{）和右大括号（}）之间。

```

public class Wrecker {
    String author = "Ignoto";

    public void infectFile() {
        VirusCode vic = new VirusCode(1024);
    }

    class VirusCode {
        int vSize;

        VirusCode(int size) {
            vSize = size;
        }
    }
}

```

内部类的使用方法与其他助手类相同，主要差别（除位置外）出现在编译之后。内部类没有使用 class 语句来指定名称，而是由编译器给它们指定名称，其中包含主类名。

在上一个示例中，编译器将生成 Wrecker.class 和 Wrecker\$VirusCode.class。

11.5 使用关键字 this

由于可以同时引用其他类的变量和方法以及当前类的变量和方法，所以有时候引用的变量将不清楚。为使变量引用更清晰，一种解决办法是使用关键字 this，它引用程序本身的对象。

使用对象的方法或变量时，将对象名放在变量名或方法名之前，并用句点隔开，请看下面的例子：

```
Virus chickenpox = new Virus();
chickenpox.name = "LoveHandles";
chickenpox.setSeconds(75);
```

这些语句创建一个名为 `chickenpox` 的 `Virus` 对象，设置其 `chickenpox` 变量的名字，然后调用其 `setSeconds()` 方法。

在程序中有时需要引用当前对象，即程序本身代表的对象。例如，在 `Virus` 类中，可能有一个方法，该方法有自己的 `author` 变量：

```
public void checkAuthor() {
    String author = null;
}
```

在 `chickenpox()` 方法内存在一个名为 `author` 的变量，它不同于对象的变量 `author`。如果要引用当前对象的 `author` 变量，必须使用关键字 `this`，如下所示：

```
System.out.println(this.author);
```

通过使用 `this`，可以明确地指出了要引用的变量或方法。在类中，可以 `this` 代替对象名。例如，如果要将当前对象作为参数传递给方法，可以使用下面的语句：

```
verifyData(this);
```

在很多情况下，无需使用 `this` 就能清晰地引用对象的变量或方法。然而，使用 `this` 确保引用明确也没有什么害处。

当设置对象实例的变量值时，这个 `this` 关键字将在构造函数中发挥作用。来看具有 `author` 和 `maxFileSize` 变量的 `Virus` 对象，下面这个构造函数将会设置这两个变量：

```
public Virus(String author, int maxFileSize) {
    this.author = author;
    this.maxFileSize = maxFileSize;
}
```

11.6 使用类方法和类变量

在我们的律师的坚持下，这里将不再创建病毒程序，而创建一个简单的 `Virus` 对象，它可以统计程序创建的 `Virus` 对象数并报告该数目。

在 NetBeans 选择 File->New File，创建一个新的 Java 空文件，将其命名为 `Virus`，然后在源代码编辑器中输入程序清单 11.1 中的内容。

程序清单 11.1 `Virus.java` 程序

```
1: public class Virus {
2:     static int virusCount = 0;
3:
4:     public Virus() {
5:         virusCount++;
```

```

6:    }
7:
8:    static int getVirusCount() {
9:        return virusCount;
10:    }
11: }

```

在 NetBeans 自动编译之后，保存该文件。这个类缺乏 `main()` 方法，因此不能直接运行。为了测试这个新的 `Virus` 类，需要再创建一个类，它能够创建 `Virus` 对象。

`VirusLab` 类是一个简单的应用程序，它创建 `Virus` 对象，然后通过 `Virus` 类的 `getVirusCount()` 方法来计算创建的对象数量。

在字处理程序中新建一个文件，输入程序清单 11.2 所示的内容，完成后将文件保存为 `VirusLab.java`。

程序清单 11.2 `VirusLab.java` 程序

```

1: public class VirusLab {
2:     public static void main(String[] args) {
3:         int numViruses = Integer.parseInt(args[0]);
4:         if (numViruses > 0) {
5:             Virus[] virii = new Virus[numViruses];
6:             for (int i = 0; i < numViruses; i++) {
7:                 virii[i] = new Virus();
8:             }
9:             System.out.println("There are " + Virus.getVirusCount()
10:                                + " viruses.");
11:         }
12:     }
13: }

```

`VirusLab` 类是一个应用程序，运行时从命令行接受一个参数：要创建的 `Virus` 对象数。要在 NetBeans 中指定命令行参数，请执行如下操作。

1. 选择 `Run->Set Project Configuration->Customize`，打开 `The Project Properties` 对话框。
2. 在 `Main Class` 字段输入“`VirusLab`”，在 `Arguments` 字段输入希望程序创建的 `Virus` 对象的数量。
3. 单击 `OK` 按钮关闭对话框。

以这种方式配置过程序之后，在 NetBeans 中选择 `Run->Run Main Project` 命令来运行该程序。

参数将使用发送给 `main()` 方法的字符数组读入到应用程序中。在 `VirusLab` 类中，这将在第 2 行执行。

为了将参数用作整数，必须将其从 `String` 对象转换为整数，这就需要使用 `Integer` 类的 `parseInt()` 方法。在第 3 行，使用通过命令行传递给程序的第一个参数创建了一个名为 `numViruses` 的变量。

如果 `numViruses` 变量大于 0，在 `VirusLab` 应用程序中将发生下列事情。

- 第 5 行：创建一个 `Virus` 对象数组，该数组的长度由 `numViruses` 变量指定。
- 第 6~8 行：用一个 `for` 循环为数组中的每个 `Virus` 对象调用构造函数。

- 第9~10行：创建完所有 Virus 对象后，使用 Virus 类的 `getVirusCount()` 方法计算创建了多少个对象，这应该与运行应用程序 VirusLab 时在命令行输入的参数相同。

如果 `numViruses` 变量不大于 0，应用程序 VirusLab 将什么都不做。

编译文件 `VirusLab.java` 后，使用任何命令行参数测试该应用程序。可创建的 Virus 对象数量与取决于运行 VirusLab 应用程序的计算机系统的可用内存量。在作者的计算机系统上，超过 5.5×10^6 个 Virus 对象时，将导致程序显示消息 “`OutOfMemoryError`”，然后崩溃。

如果指定的 Virus 对象数量小于你的系统可以处理的数量，则输出应该会如图 11.1 所示。

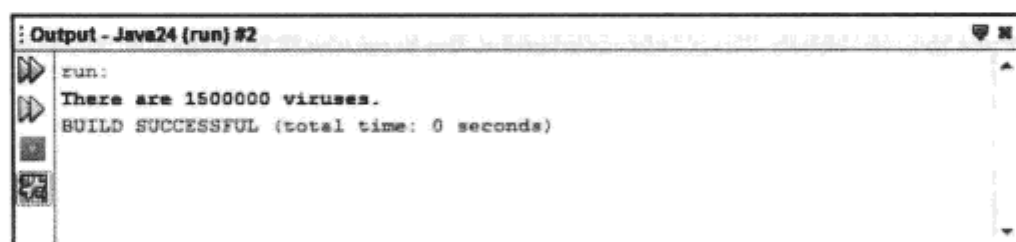


图 11.1

VirusLab 程序的输出

11.7 总结

至此，读者已经学完了本书三章面向对象编程内容中的两章。你学习了如何创建对象、为对象和类定义行为和属性，以及如何使用类型转换将对象和变量转换为其他类型。

以对象的方式思考是学习 Java 编程语言面临的严峻挑战之一。然而，理解对象后，将意识到整个 Java 语言都是利用了对象和类。

下一章将介绍如何为对象指定父对象和子对象。

11.8 问与答

问：为了使用类变量或类方法，必须创建一个对象吗？

答：由于类变量和类方法不与特定对象相关联，不需要仅为使用它们而创建对象。`Integer.parseInt()` 方法的使用就是一个例子，你不需要为将字符串转换为整数而创建一个新的 `Integer` 对象。

问：有没有 Java 支持的所有内置方法的列表？

答：Oracle 提供了 Java 语言中所有类的完整文档，包括你使用的所有公有方法。该文档位于 <http://download.oracle.com/javase/7/docs/api>。

11.9 测验

回答下面的问题，以检测你是否有理解面向对象编程技术所需的属性和行为。

11.9.1 问题

1. 在 Java 类中，方法是一种什么？

a. 属性。

- b. 语句。
 - c. 行为。
2. 要将变量声明为类变量，创建它时必须使用什么关键字？
- a. `new`。
 - b. `public`。
 - c. `static`。
3. 变量可用的程序部分称为什么？
- a. 窝。
 - b. 作用域。
 - c. 变量流域。

11.9.2 答案

- 1. c. 方法是由语句组成的，但它是一种行为。
- 2. c. 如果不指定关键字 `static`，将为对象变量而不是类变量。
- 3. b. 在变量的作用域外使用变量时，编辑器将会提示错误。

11.10 练习

如果谈论病毒没有让你生病，可通过下面的练习加深对本章主题的理解：

- 在 `Virus` 类添加一个 `private` 变量，它用于存储整数，名为 `newSeconds`。创建两个方法，其中一个返回变量 `newSeconds` 的值；另一个在提供的新值位于 60 和 100 之间时，将变量 `newSeconds` 设置为提供的值。
- 编写一个 Java 应用程序，它接受一个字符串参数，并依次将其转换为 `float` 变量、`Float` 对象和 `int` 变量。使用不同的参数运行程序多次，看看结果如何。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 12 章

充分利用现有对象

本章介绍如下内容：

- 超类和子类的设计；
- 建立继承层次；
- 覆盖方法。

Java 对象非常适合繁殖。当创建了一个对象（一组属性和行为）时，实际上你设计了可遗传给后代的品质。这些子对象将继承父对象的属性和行为，而且它们还可以添加父对象没有的属性和行为。

这种规律称为继承，也就是超类（父母）遗传给子类（孩子）。继承是面向对象编程最有用的方面，本章将详细介绍它。

面向对象编程的另一个有用的方面是能够创建可在不同程序中使用的对象。可重用性使得开发没有错误的可靠程序更加容易。

12.1 继承的威力

每当读者使用标准 Java 类（如 String 或 Integer）时，你使用的就是继承。Java 类被组织成金字塔型的类层次结构，其中所有的类都是从 Object 类派生而来的。

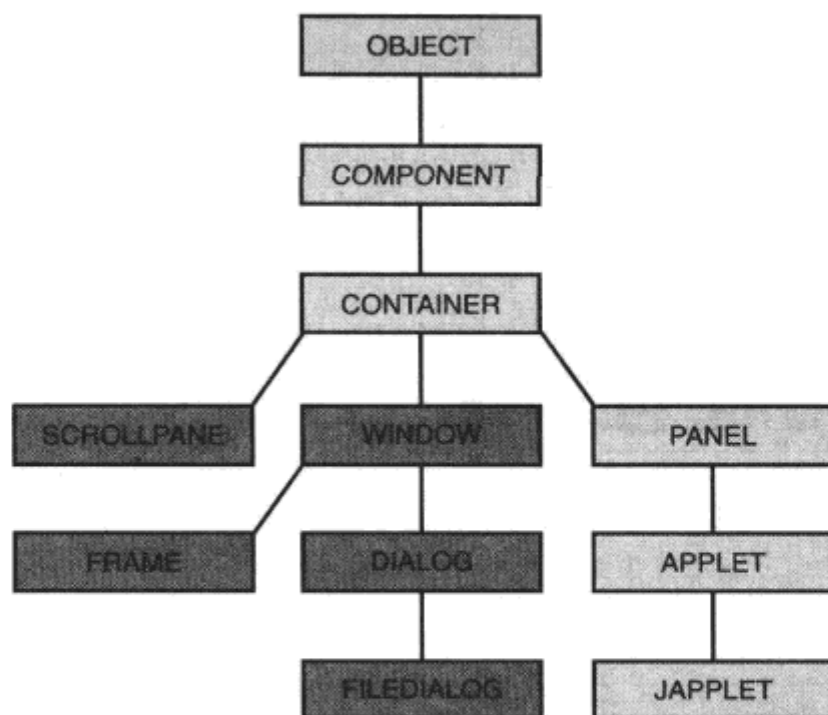
类继承其上面所有的超类。要了解其中的原理，来看看 JApplet 类。这个类是所有 applet 的超类，后者是使用图形用户界面框架（称之为 Swing）而且基于浏览器的 Java 程序。JApplet 是 Applet 的子类。

图 12.1 显示了 JApplet 的部分家谱，其中每个方块都是一个类，超类与其下面的子类用直线连接。

最顶端是 Object 类。在继承层次结构中，JApplet 有 5 个超类：Applet、Panel、Container、Component、Object。

图 12.1

Applet 类的家谱



JApplet 从这些类继承属性和行为，因为在继承层次结构中，它们都在 JApplet 的正上方。JApplet 不继承图 12.1 中用阴影标识的 5 个类，包括 Dialog 和 Frame，因为在继承层次结构中，它们不在 JApplet 的上方。

如果感到迷惑，可将层次结构视为家谱。JApplet 继承父亲、祖父，不断向上直到太祖父 Object 类。然而，JApplet 不会从兄弟或堂兄那里继承。

创建一个新的类可以归结为执行如下任务：你指需要定义其不同于现有类的地方即可，其他工作都已经为你做好。

12.1.1 继承行为和属性

类的行为和属性由两部分组成：自己的行为 and 属性以及从超类继承的行为和属性。

下面是 Applet 的一些行为和属性：

- equals() 方法确定 JApplet 对象的值是否与另一个对象相同；
- setBackground() 方法设置 applet 窗口的背景颜色；
- add() 方法给 applet 添加用户界面组件，如按钮和文本框等；
- setLayout() 方法定义如何组织 applet 的图形用户界面。

JApplet 类可以使用所有这些方法，即使 setLayout() 方法不是从其他类继承来的。equals() 方法是在 Object 类中定义的，setBackground() 方法来自 Component 类，而 add() 方法来自 Container 类。

12.1.2 覆盖方法

在 JApplet 类中定义的有些方法也在其超类中定义了，例如，update() 方法在 JApplet 和 Component 类中都有定义。当方法在子类和超类中都定义了时，将使用子类中的定义；因此子类可以修改、替换或完全删除超类的行为和属性。就 update() 方法而言，它旨在删除超类

中的一些行为。

在子类中创建新方法以修改从超类继承来的行为被称为“覆盖”(overriding)方法。如果继承的行为不能获得所需的结果,则需要覆盖相应的方法。

12.2 建立继承

使用关键字 `extends` 将一个类指定为另一个类的子类,如下所示:

```
class AnimatedLogo extends JApplet {
    // behavior and attributes go here
}
```

上述语句创建 `AnimatedLogo` 类,并将其指定为 `JApplet` 的子类。读者在第 17 章将会看到,所有的 `Swing applet` 都必须是 `JApplet` 的子类,因为在 `Web` 页面中运行时,它们需要这个类提供的功能。

`AnimatedLogo` 必须覆盖方法 `paint()`,该方法用于绘制要在程序窗口中显示的所有内容。`paint()`方法是由 `Component` 类实现的,并向下传递给 `AnimatedLogo` 类,但 `paint()`不做任何事情,`Component` 类提供该方法旨在让其子类需要显示内容时可使用它。

要覆盖该方法,必须以与超类相同的方式声明它,即 `public` 方法仍必须是 `public` 的,方法的返回值类型必须相同,且参数的数量和类型都不能变。

在 `Component` 类中, `paint()`方法的声明如下:

```
public void paint(Graphics g) {
```

在 `AnimatedLogo` 中覆盖该方法时,也使用类似于下面的语句:

```
public void paint(Graphics screen) {
```

唯一的差别是 `Graphics` 对象的名称,这对于判断创建方法的方式是否相同没有影响。这两个语句是相同的,因为下面几项相同:

- 两个 `paint()`方法都是 `public` 的;
- 两个方法都没有返回值,因为声明时使用了关键字 `void`;
- 都接受 `Graphics` 对象作为唯一的参数。

在子类中使用 `this` 和 `super`

在子类中 `this` 和 `super` 是两个很有用的关键字。

前一章讲到,关键字 `this` 用于引用当前对象。创建类时,要引用根据该类创建的特定对象,可使用 `this`,如下面的语句所示:

```
this.title = "Cagney";
```

这条语句将对象的 `title` 变量设置为文本 `Cagney`。

关键字 `super` 的用途与此类似：引用对象的上一级超类。可以下面几种方式使用 `super`：

- 引用超类的构造函数，如 `super("Adam", 12)`；
- 引用超类的变量，如 `super.hawaii = 50`；
- 引用超类的方法，如 `super.dragnet()`。

使用关键字 `super` 的方式之一是在子类的构造函数中。子类从其超类继承所有的行为和属性，因此将子类的构造函数与超类的构造函数关联起来，否则有些行为和属性可能不能正确设置，导致子类不能正常工作。

为此，在子类的构造函数中，第一条语句必须调用超类的构造函数，因此需要使用关键字 `super`，如下面的语句所示：

```
public readFiles(String name, int length) {
    super(name, length);
}
```

这是子类的构造函数，它使用 `super(name, length)` 调用超类相应的构造函数。

如果不使用 `super` 来调用超类的构造函数，则在子类构造函数执行时，Java 将自动调用无参数的超类构造函数。如果该超类构造函数不存在或提供了意料之外的行为，将导致错误，因此最好手工调用超类的构造函数。

12.3 使用现有的对象

面向对象编程鼓励代码重用。如果在某个 Java 编程项目中开发了一个对象，可以将其用于其他项目中，而无需做任何修改。

如果 Java 类是设计良好的，完全可以在其他程序中使用它。在程序中可以使用的对象越多，编写软件需要做的工作就越少。如果有优秀的拼写检查对象能够满足你的需要，便可以使用它而不用自己编写。这甚至可以给老板错觉：将拼写检查功能添加到程序中需要做很多的工作，这样你就有更多的时间在办公室打长途电话。

By the Way

注意：

本书作者像很多人一样，是个自由职业者，在家办公。评估其上述建议时，别忘了考虑工作环境。

Java 刚面世时，共享对象的系统很不正规。程序员尽可能独立地开发其对象，并且通过使用私有变量和读写变量的公有方法来进行保护，以防误用。

有了开发可重用对象的标准后，共享对象变得功能强大。

标准的好处体现在下面几个方面：

- 可以更少地编写关于对象工作原理的文档，因为知道标准的人都对其工作原理很清楚；
- 可以根据标准来设计开发工具，使得使用对象更容易；
- 两个遵循标准的对象可以相互交互，而无需通过特殊编程使其互相兼容。

12.4 将相同类型的对象存储到 Vector 中

编写计算机程序时，需要做出的一个重要决策是，将数据存储在哪里。在本书的前半部分，介绍了 3 种存储信息的地方：基本数据类型（如 `int` 和 `char`）、数组和 `String` 对象。

任何 Java 类都能够存储数据。最有用的一种是 `Vector`，它是一种存储相同类对象的数据结构。

`Vector` 类似于数组，也存储相关的数据，但其长度可动态地增减。

`Vector` 类位于 `java.util` 包中，这是 Java 类库中最有用的一个包。在程序中使用它，可使用一条 `import` 语句：

```
import java.util.Vector;
```

`Vector` 存储的对象要么属于同一个类，要么有相同的超类。要创建 `Vector`，需要引用两个类：`Vector` 和要存储在 `Vector` 的类。

将要在 `Vector` 中存储的类的名称放在 “<” 和 “>” 之间，如下述语句所示：

```
Vector<String> vector = new Vector<String>();
```

上述语句创建一个存储字符串的 `Vector`。以这种方式指定的存储在 `Vector` 中的类使用了通用类型，通用类型用来指示数据结构（比如 `Vector`）可以存储的对象类型。在以前的 Java 版本中，要使用 `Vector`，需要调用构造函数，如下所示：

```
Vector vector = new Vector();
```

虽然仍可以这样做，但通用类型使代码更可靠，因为它向编译器提供了一种防止程序员误用 `Vector` 的方法。如果试图将一个 `Integer` 对象存储到本应存储 `String` 对象的 `Vector` 中，编译器将报错。

与数组不同，`Vector` 存储的元素数量并非固定的。`Vector` 默认包含 10 个元素，如果你知道需要存储更多的存储，可通过构造函数的参数指定长度。下面的语句创建一个包含 300 个元素的 `Vector`：

```
Vector<String> victoria = new Vector<String>(300);
```

要将对象加入到 `Vector` 中，可调用 `Vector` 的 `add()` 方法，并将对象作为其唯一的参数：

```
victoria.add("Vance");  
victoria.add("Vernon");  
victoria.add("Velma");
```

对象按照顺序加入到 `Vector` 中，因此如果这是加入到 `victoria` 中的前 3 个元素，则元素 0 为 “Vance”，元素 1 为 “Vernon”，元素 2 为 “Velma”。

要从 `Vector` 中检索元素，可使用方法 `get()` 并将元素的索引号作为其参数：

```
String name = victoria.get(1);
```

该语句将 “Vernon” 存储到字符串变量 name 中。

要查看 Vector 的元素中是否包含某个对象，可调用 contains() 方法并将该对象作为参数：

```
if (victoria.contains("Velma")) {
    System.out.println("Velma found");
}
```

要将对象从 Vector 中删除，可调用 remove() 方法并将该对象或其索引号作为参数：

```
victoria.remove(0);
victoria.remove("Vernon");
```

这两条语句导致 Vector 中只留下 “Velma”。

遍历 Vector

Java 新增了一种 for 循环以方便载入 Vector 以及依次检查其每个元素。

该循环由两部分组成，比第 8 章介绍的 for 循环少一部分。

第一部分是初始化部分：变量的类及其名称，该变量用于存储从 Vector 中取出的每个对象。该对象与 Vector 中存储的对象必须属于同一个类。

第二部分指出要遍历的 Vector。

下面的代码遍历 Vector victoria，并将其中的每个对象的名字显示到屏幕上：

```
for (String name : victoria) {
    System.out.println(name);
}
```

本章将要创建的第一个应用程序使用 Vector 和特殊的 for 循环将一系列字符串按字母顺序显示到屏幕上。这些字符串来自一个数组和命令行参数。

在 NetBeans 中打开 Java24 项目，然后选择 File->New File，创建一个新的 Java 空文件，并将其命名为 StringLister。然后在源代码编辑器中输入程序清单 12.1 中的所有文本，并保存。

程序清单 12.1 StringLister.java 程序的完整代码

```
1: import java.util.*;
2:
3: public class StringLister {
4:     String[] names = { "Spanky", "Alfalfa", "Buckwheat", "Daria",
5:         "Stymie", "Marianne", "Scotty", "Tommy", "Chubby" };
6:
7:     public StringLister(String[] moreNames) {
8:         Vector<String> list = new Vector<String>();
9:         for (int i = 0; i < names.length; i++) {
10:             list.add(names[i]);
11:         }
12:         for (int i = 0; i < moreNames.length; i++) {
13:             list.add(moreNames[i]);
14:         }
15:         Collections.sort(list);
16:         for (String name : list) {
17:             System.out.println(name);
18:         }
19:     }
20: }
```

```

18:     }
19: }
20:
21: public static void main(String[] args) {
22:     StringLister lister = new StringLister(args);
23: }
24: }

```

在运行该应用程序时（通过 Run->Run File 来运行），应该先选择 Run->Set Project Configuration->Customize 命令，将 main class 设置为 StringLister，并将参数设置为名字，当有多个名字时，需要使用空格分开，比如 Jackie Nickey Farina Woim。

命令行中指定的名字将添加到第 4~5 行的字符串数组中。由于在程序运行之前无法得知名字个数，因此使用 Vector 来存储这些字符串比数组更合适。

使用 Collection 类的一个方法对存储在 Vector 中的字符串按字母顺序排序：

```
Collections.sort(list);
```

和 Vector 一样，这个类也位于 java.util 包中。在 Java 中，Vector 以及其他有用的数据结构被称为集合。

当运行该程序时，输出结果应该是一组按字母排序的名字（见图 12.2）。Vector 在存储空间上的灵活性可以让你将额外的名字添加到数据结构中，并与其他名字一起排序。

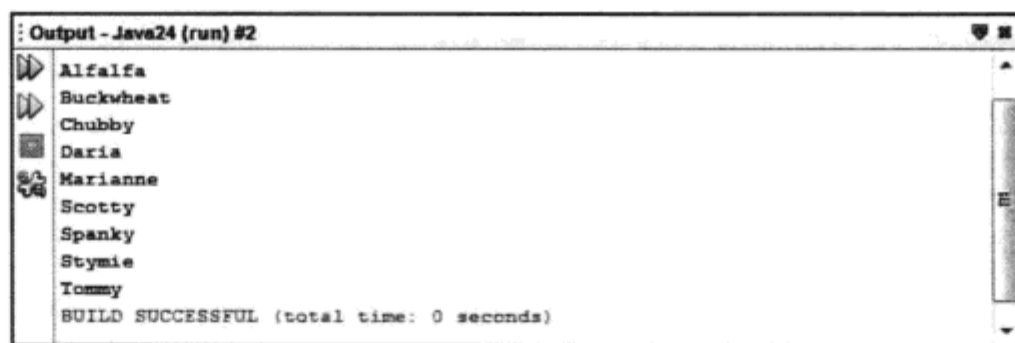


图 12.2
StringLister 程序的
输出结果

12.5 创建子类

为了看一个关于继承的示例，现在创建一个名为 Point3D 的类，它代表三维空间中的一个点。二维点可用坐标 (x, y) 表示。applet 使用 (x, y) 坐标来决定在哪里显示文本和图形，而三维空间添加第三个坐标，可将其称为 z。

Point3D 对象需要做下面 3 件事：

- 记录对象的 (x, y, z) 坐标；
- 需要时将对象移到新坐标 (x, y, z) 处；
- 需要时沿 x、y 和 z 轴移动特定的距离。

Java 有一个表示二维点的标准类，名为 Point。它有两个整型变量 x 和 y，用于存储 Point 对象 (x, y) 坐标。它还有名为 move() 的方法，用于将一个点放在指定的位置，还有一个名为 translate() 的方法，将对象沿 x 和 y 轴移动特定的距离。

在 NetBeans 中的 Java24 项目中, 创建一个新的空文件, 将其命名为 Point3D, 然后在源代码编辑器窗口中输入程序清单 12.2 中的全部文本, 并保存。

程序清单 12.2 Point3D.java 的完整源代码

```
1: import java.awt.*;
2:
3: public class Point3D extends Point {
4:     public int z;
5:
6:     public Point3D(int x, int y, int z) {
7:         super(x,y);
8:         this.z = z;
9:     }
10:
11:     public void move(int x, int y, int z) {
12:         this.z = z;
13:         super.move(x, y);
14:     }
15:
16:     public void translate(int x, int y, int z) {
17:         this.z += z;
18:         super.translate(x, y);
19:     }
20: }
```

Point3D 类没有 main() 块语句, 所以不能使用 Java 解释器运行它。但是你可以在需要三维空间点的 Java 程序中使用它。

Point3D 类只需完成其超类 Point 不能完成的工作, 主要是记录整型变量 z, 将 z 作为 move() 方法、translate() 方法和 Point3D() 构造函数的参数。

这些方法都使用关键字 supper 和 this。this 用于引用当前的 Point3D 对象, 因此第 8 行的 this.z=z; 语句将变量 z 设置为第 6 行作为参数传入的 z 的值。

super 语句引用当前对象的超类 Point, 用于设置 Point3D 类继承的变量和调用继承的方法。第 7 行的语句 super(x, y) 调用超类的构造函数 Point(x, y), 该构造函数用来设置 Point3D 对象的 (x, y) 坐标。由于 Point 类能够处理坐标轴 x 和 y, 如果 Point3D 类也这样做将多余。

要测试编译后的 Point3D 类, 可以创建一个使用 Point 对象和 Point3D 对象的程序, 并在屏幕上移动它们。在 NetBeans 中创建一个名为 PointTester 的新文件, 然后输入程序清单 12.3 中的全部文本。当保存该文件时, 将会自动对其编译。

程序清单 12.3 PointTester.java 程序的完整源代码

```
1: import java.awt.*;
2:
3: class PointTester {
4:     public static void main(String[] args) {
5:         Point object1 = new Point(11,22);
6:         Point3D object2 = new Point3D(7,6,64);
7:
8:         System.out.println("The 2D point is located at (" + object1.x
9:             + ", " + object1.y + ")");
10:        System.out.println("\tIt's being moved to (4, 13)");
11:        object1.move(4,13);
12:        System.out.println("The 2D point is now at (" + object1.x
```

```

13:         + ", " + object1.y + "));
14:     System.out.println("\tIt's being moved -10 units on both the x "
15:         + "and y axes");
16:     object1.translate(-10,-10);
17:     System.out.println("The 2D point ends up at (" + object1.x
18:         + ", " + object1.y + ")\n");
19:
20:     System.out.println("The 3D point is located at (" + object2.x
21:         + ", " + object2.y + ", " + object2.z + "));
22:     System.out.println("\tIt's being moved to (10, 22, 71)");
23:     object2.move(10,22,71);
24:     System.out.println("The 3D point is now at (" + object2.x
25:         + ", " + object2.y + ", " + object2.z + "));
26:     System.out.println("\tIt's being moved -20 units on the x, y "
27:         + "and z axes");
28:     object2.translate(-20,-20,-20);
29:     System.out.println("The 3D point ends up at (" + object2.x
30:         + ", " + object2.y + ", " + object2.z + "));
31: }
32: }

```

通过选择 Run->Run File 运行该文件时，如果该程序已经成功编译，则会看到如图 12.3 所示的输出。否则，查看源代码编辑器窗口旁边的红色图标，它可以指示是哪一行代码触发了错误。

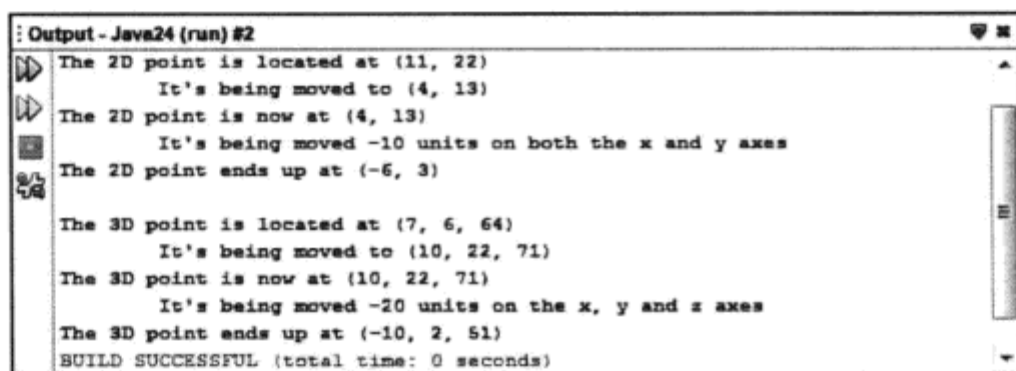


图 12.3

PointTester 程序的
输出

12.6 总结

当人们谈论生育奇迹时，他们谈论的可能不是从 Java 超类派生子类的方式或在类层次结构中继承行为和属性的方式。

如果现实世界与面向对象编程的工作方式相同，则莫扎特的每个子孙都可以选择成为一名卓越的作曲家，马克吐温的所有子孙都能写出密西西比河边生活的诗歌。所有从祖先那里继承的技能都可以由你直接支配，不需要通过任何努力。

从奇迹的程度看，继承不能同物种延续和减免大笔税款相比。然而，这是一种设计软件的高效方法，可最大限度地减少重复工作。

12.7 问与答

问：到目前为止，所创建的大多数 Java 程序都没有使用 extends 来从超类继承，这是否就意味着它们不在层次结构中？

答：使用 Java 创建的所有类都是 Java 类层次结构的一部分，因为编写程序时如果不使

用关键字 `extends`，则默认超类为 `Object` 对象。所有类的方法 `equals()` 和 `toString()` 都是自动从 `Object` 类继承而来的行为。

12.8 测验

为测试读者从本章继承了什么知识，请回答下列问题。

12.8.1 问题

1. 如果在子类中要以不同于超类的方式处理方法，应该如何做？
 - a. 删除超类中的方法。
 - b. 覆盖超类中的方法。
 - c. 给《San Jose Mercury News》的编辑写信，希望 Java 的开发者能看到这封信。
2. 哪个方法用于检索存储在 `Vector` 中的元素？
 - a. `get()`。
 - b. `read()`。
 - c. `elementAt()`。
3. 可以使用哪个关键字来引用当前对象的方法和变量？
 - a. `this`。
 - b. `that`。
 - c. `theOther`。

12.8.2 答案

1. b. 由于可以覆盖方法，所以无需修改超类的任何方面。
2. a. 方法 `get()` 接受一个参数——元素的索引号。
3. a. `this` 关键字引用当前对象。

12.9 练习

如果你脑海中出现了学习更多知识的强烈愿望，可以通过下面的练习来获得更多有关继承的知识。

- 创建一个 `Point4D` 类，在 `Point3D` 类创建的 (x, y, z) 坐标系的基础上再加上 `t` 坐标。`t` 坐标代表时间，因此必须确保它不会被设置为负值。
- 使用足球队的进攻成员——边锋、外接手、阻挡线队员、跑锋、四分卫，设计一个代表这些球员技能的类层次结构，将通用技能放在层次结构的较上层。例如，阻挡行为应由边锋和阻挡线队员继承，速度应由外接手和跑锋继承。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 13 章

创建简单的用户界面

本章介绍如下内容：

- 创建按钮这样的用户界面组件；
- 创建标签、文本框和其他组件；
- 将组件编组；
- 将组件放在其他组件中；
- 打开和关闭窗口。

在本章，所有的事情将变得繁琐起来。读者在使用 Java 创建第一个图形用户界面（GUI）时，将会面临很大的一个烂摊子。

计算机用户期望其软件有图形用户界面、通过鼠标接受用户的输入并像其他程序那样工作。尽管有些用户仍然希望在命令行环境下（比如 MS-DOS 或 Linux shell）工作，但是大多数用户会非常不适应不提供点选、拖放图形界面的软件。

Java 通过使用 Swing 来支持这种类型的软件，Swing 是 Java 类的集合，它可以表示所有不同的按钮、文本框、滑块，以及可以成为 GUI 一部分的其他组件，同时它还包括了可以从上述组件中接收用户输入的类。

13.1 Swing 和抽象窗口工具包

Java 是一种跨平台语言，使用它可以为很多不同的操作系统编写程序，因此其图形用户软件必须是灵活的，不仅能够支持 Windows 和 Mac 窗口风格，还必须能够满足其他平台的要求。

在 Java 中，使用两组类来开发程序的用户界面：Swing 和抽象窗口工具包（Abstract Windowing Toolkit, AWT）。这些类让你能够创建图形用户界面以及接收用户输入。

Swing 提供了编写使用图形用户界面的程序所需的一切。使用 Java 的用户界面类，可以创建包括下述元素的 GUI：

- 按钮、复选框、标签和其他简单组件；
- 文本框、滑块和其他复杂组件；
- 下拉菜单和弹出菜单；
- 窗口、框架、对话框和 applet 窗口。

13.2 使用组件

在 Java 中，图形用户界面的每部分都由 Swing 包中的一个类表示。对于按钮，是 JButton 类，窗口是 JWindow 类，文本框是 JTextField 类等。

为创建并显示界面，需要创建对象，设置其变量并调用其方法。使用的技巧与前面介绍面向对象编程的三章中相同。

组织图形用户界面时，需要使用两类对象：组件和容器。组件是用户界面中的独立元素，如按钮或滑块；容器是用于容纳其他组件的组件。

创建界面的第一步是创建能够容纳组件的容器。在应用程序中，该容器通常是窗口或框架。

13.2.1 窗口和框架

窗口和框架是能够在用户界面上显示，并且可以容纳其他组件的容器。窗口是一种简单容器，不像常规图形用户界面那样，在顶端有标题栏和其他按钮。框架是一种这样的窗口：包含用户运行软件时希望看到的所有常见的窗口特性，如关闭按钮、最大化按钮和最小化按钮。

这些容器分别是使用 Swing 包中的 JWindow 和 JFrame 类创建的。为了在 Java 程序使用 Swing 包，可使用下面的语句：

```
import javax.swing.*;
```

一种在 Java 应用程序中使用框架的方法是，将应用程序声明为 JFrame 的子类，这样应用程序将继承作为框架所需的行为。下面的语句创建了 JFrame 的一个子类：

```
import javax.swing.*;

public class MainFrame extends JFrame {
    public MainFrame() {
        // set up the frame
    }
}
```

这个类创建了一个框架，但没有完整地设置。创建框架时，必须在框架的构造函数中执行几种操作：

- 调用超类 `JFrame` 的构造函数；
- 设置框架的标题；
- 设置框架的大小；
- 设置框架的外观；
- 定义用户关闭框架时应执行的操作。

还必须显示框架，除非由于某种原因，在应用程序开始运行时不应显示框架。

所有这些操作都可以在框架的构造函数中完成。该方法首先应使用关键字 `super` 调用 `JFrame` 的一个构造函数。下面是一个例子：

```
super();
```

该语句调用不接受参数的 `JFrame` 构造函数。也可以调用将框架标题作为参数的构造函数：

```
super("Main Frame");
```

这将框架的标题设置为指定的字符串，标题出现框架顶端的标题栏中。在这里，标题为 `Main Frame`。

如果不采用这种方式设置标题，可以调用框架的 `setTitle()` 方法，并将一个字符串作为参数：

```
setTitle("Main Frame");
```

要指定框架的大小，可调用 `setSize()` 方法并指定两个参数：宽度和高度。下面的语句将框架设置为宽 350 像素，高 125 像素：

```
setSize(350, 125);
```

设置框架大小的另一种方式是，先用组件填充它，然后调用不带参数的 `pack()` 方法，如下例所示：

```
pack();
```

方法 `pack()` 根据框架中每个组件的首选尺寸来设置框架。每个界面组件都有首选尺寸，虽然有时会忽略首选尺寸，这取决于组件在界面中是如何排列的。调用 `pack()` 方法前，无需显式地设置框架的大小，该方法在框架显示前将其设置为足够大。

每个框架的标题栏上都有一个用于关闭框架的按钮。在 Windows 系统中，该按钮出现在框架的右上角并用“X”标识。要定义单击该按钮发生的情况，可调用框架的 `setDefaultCloseOperation()` 方法，并将 4 个 `JFrame` 类变量之一作为参数。

- `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)`：按钮被单击时退出程序。
- `setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)`：关闭框架并销毁框架对象，但应用程序继续运行。
- `setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE)`：让框架打开并继续运行。

➤ `setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE)`: 关闭框架并继续运行。

当使用 Swing 创建了图形用户界面时，可以对它的外观进行自定义，从而控制按钮和其他组件的显示方式和行为。

Java 7 中引入了增强的外观，名为 Nimbus，当时在类中使用时，必须先启用。通过调用主 Swing 包中 `UIManager` 类的 `setLookAndFeel()` 方法可以设置外观。该方法接受一个参数：外观类的完整名称。

下面的语句将 Nimbus 设置为外观：

```
UIManager.setLookAndFeel(
    "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
);
```

最后要做的一件事情是显示框架：使用 `true` 作为参数调用 `setVisible()` 方法：

```
setVisible(true);
```

这将以指定的宽度和高度打开框架。也可以用 `false` 作为参数调用该方法，以隐藏框架。

程序清单 13.1 包含本节介绍的源代码，在 Java 空文件中输入这些语句并将文件保存为 `SalutonFrame.java`。

程序清单 13.1 SalutonFrame.java 程序

```
1: import javax.swing.*;
2:
3: public class SalutonFrame extends JFrame {
4:     public SalutonFrame() {
5:         super("Saluton mondo!");
6:         setLookAndFeel();
7:         setSize(350, 100);
8:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9:         setVisible(true);
10:    }
11:
12:    private void setLookAndFeel() {
13:        try {
14:            UIManager.setLookAndFeel(
15:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
16:            );
17:        } catch (Exception exc) {
18:            // ignore error
19:        }
20:    }
21:
22:    public static void main(String[] arguments) {
23:        SalutonFrame sal = new SalutonFrame();
24:    }
25: }
```

在程序清单 13.1 中，第 22~24 行包括了一个 `main()` 方法，它使该框架类变为一个应用程序。当运行该程序时，将会看到图 13.1 所示的框架。



图 13.1

在应用程序中显示
框架

程序 `SalutonFrame` 只是显示一个标题：世界语问候“`Saluton mondo!`”。该框架是一个空窗口，因为它没有包含任何组件。

要在框架中添加组件，必须创建组件并将其加入到容器中。每个容器都有一个 `add()` 方法，该方法接受一个参数：要显示的组件。

`SalutonFrame` 类包含一个 `setLookAndFeel()` 方法，它可以将 Nimbus 指定为框架的外观。代码第 14~16 行调用了 `UIManager` 类的这个 `setLookAndFeel()` 方法来实现该目的。

调用该方法的语句放置在 `try-catch` 语句块中，从而可以处理有可能发生的错误。`try` 和 `catch` 语句在前面一直没有介绍，因此对读者来说还较为陌生。使用这些语句来处理错误的知识将在第 18 章详细讲解。

此时，你只需要知道调用 `UIManager.setLookAndFeel()` 方法可以设置 GUI 的外观即可。如果调用该方法时发生了错误，程序将显示其默认的外观，而不是 Nimbus。

13.2.2 按钮

可以添加到容器中的一种简单组件是 `JButton` 对象。`JButton` 和本章介绍的其他组件一样，也位于 `java.awt.swing` 包中。`JButton` 对象是一个可单击的按钮，其标签描述了单击按钮的结果。该标签可以是文本、图像或两者的组合。下面的语句创建一个名为 `okButton` 的 `JButton` 对象，并将其标签设置为 OK：

```
JButton okButton = new JButton("OK");
```

创建 `JButton` 等组件后，应调用 `add()` 方法将其加入到容器中：

```
add(okButton);
```

在容器中添加组件时，不需要指明组件在容器中显示的位置，组件的布局由被称为布局管理器的对象决定。最简单的布局管理器是 `FlowLayout` 类，它位于 `java.awt` 包中。

要让容器使用特定的布局管理器，必须首先创建该布局管理器的对象。使用下面这样的语句创建 `FlowLayout` 对象：

```
FlowLayout flo = new FlowLayout();
```

创建布局管理器后，调用容器的 `setLayout()` 方法将其管理器同容器关联起来，该方法将布局管理器对象作为唯一的参数，如下例所示：

```
pane.setLayout(flo);
```

这条语句将 `flo` 对象指定为容器 `pane` 的布局管理器。

接下来要创建的应用程序为 `Playback` 类，它可以显示一个包含 3 个按钮的框架。在一个

Java 空文件中输入程序清单 13.2 中的所有文本，然后保存。

程序清单 13.2 Playback.java 的完整源代码

```

1: import javax.swing.*;
2: import java.awt.*;
3:
4: public class Playback extends JFrame {
5:     public Playback() {
6:         super("Playback");
7:         setLookAndFeel();
8:         setSize(225, 80);
9:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10:        FlowLayout flo = new FlowLayout();
11:        setLayout(flo);
12:        JButton play = new JButton("Play");
13:        JButton stop = new JButton("Stop");
14:        JButton pause = new JButton("Pause");
15:        add(play);
16:        add(stop);
17:        add(pause);
18:        setVisible(true);
19:    }
20:
21:    private void setLookAndFeel() {
22:        try {
23:            UIManager.setLookAndFeel(
24:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
25:            );
26:        } catch (Exception exc) {
27:            // ignore error
28:        }
29:    }
30:
31:    public static void main(String[] arguments) {
32:        Playback pb = new Playback();
33:    }
34: }

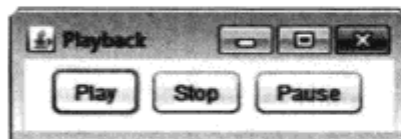
```

Playback 程序在代码第 10 行创建了一个 FlowLayout 布局管理器，并在代码第 11 行将其用于框架。当在代码第 15~17 行将 3 个按钮添加到框架中时，它们将由该布局管理器进行安置。

当运行该程序时，其输出结果应该如图 13.2 所示。用户可以单击其中的每个按钮，但什么也不会发生，因为该程序没有包含任何接收用户输入的方法，这种方法将在第 15 章介绍。

图 13.2

在 GVI 上显示按钮



可以通过这种方法将多个 Swing 用户组件添加到容器中。

By the Way

注意：

本章将介绍很多不同种类的用户界面组件，这里将不会列出创建每个组件的源代码。读者可在本书的配套网站 www.java24hours.com 找到本章所有程序的源代码。

13.2.3 标签和文本框

JLabel 组件显示用户不能修改的信息，这种信息可以是文本、图形或两者的组合。这些组件常用于标识界面中的其他组件，因此而得名。它们常用于标识文本框。

JTextField 组件是用户可以输入单行文本的区域。创建文本框时，可以设置其宽度。

下面的语句创建一个 **JLabel** 组件和一个 **JTextField** 对象，并将它们加入到容器中：

```
JLabel pageLabel = new JLabel("Web page address: ", JLabel.RIGHT);
JTextField pageAddress = new JTextField(20);
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(pageLabel);
add(pageAddress);
```

图 13.3 并排地显示该标签和文本框。这里的两条语句都使用一个参数来设置组件的外观。

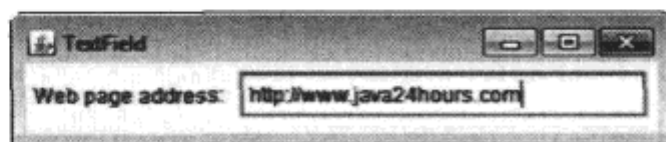


图 13.3

显示标签和文本框

标签 **pageLabel** 的文本被设置为 “Web page address:”，并使用了参数 **JLabel.RIGHT**。该参数将文本与标签右对齐，**JLabel.LEFT** 将文本左对齐，而 **JLabel.CENTER** 居中显示文本。用于 **JTextField** 的参数指定文本框的宽度为 20 个字符，也可以使用下面这样的语句指定显示在文本框的默认文本：

```
JTextField country = new JTextField("US", 29);
```

该语句创建一个 **JTextField** 对象，其宽度为 20 个字符，且内容默认为 “US”。

对象包含的文本可使用方法 **getText()** 来检索，它返回一个字符串：

```
String countryChoice = country.getText();
```

读者可能猜到了，也可以使用相应的方法来设置文本：

```
country.setText("Separate Customs Territory of Taiwan, Penghu, Kinmen,  
and Matsu");
```

13.2.4 复选框

JCheckBox 组件由文件和方框组成，用户可以选中它，也可以不选中。下面的语句创建一个 **JCheckBox** 对象并将其加入到一个容器中：

```
JCheckBox jumboSize = new JCheckBox("Jumbo Size");
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(jumboSize);
```

构造函数 **JCheckBox()** 的参数指定了显示在复选框旁边的文本。如果要选中该复选框，

可使用下面的语句：

```
JCheckBox jumboSize = new JCheckBox("Jumbo Size", true);
```

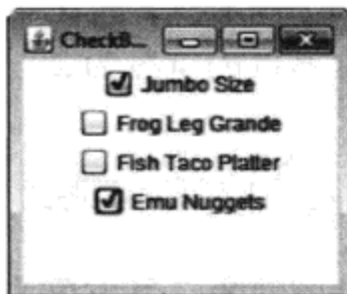
JCheckBox 可以单个显示，也可以编成组。在一组复选框中不能同时选中多个。要使 JCheckBox 对象成为某个组的一部分，必须创建一个 ButtonGroup 对象，请看下面的语句：

```
JCheckBox frogLegs = new JCheckBox("Frog Leg Grande", true);
JCheckBox fishTacos = new JCheckBox("Fish Taco Platter", false);
JCheckBox emuNuggets = new JCheckBox("Emu Nuggets", false);
FlowLayout flo = new FlowLayout();
ButtonGroup meals = new ButtonGroup();
meals.add(frogLegs);
meals.add(fishTacos);
meals.add(emuNuggets);
setLayout(flo);
add(jumboSize);
add(frogLegs);
add(fishTacos);
add(emuNuggets);
```

上述语句创建了 3 个复选框，并将它们都编组到名为 meals 的 ButtonGroup 对象下。默认情况下，复选框 Frog Leg Grande 被选中，但如果用户选择其他复选框，复选框 Frog Leg Grande 的选中标记将消失。图 13.4 显示了本节介绍的复选框。

图 13.4

显示复选框组件



13.2.5 组合框

JComboBox 组件是一个弹出式选择列表，也可以设置成能够接收文本输入。在这种情况下，用户可以使用鼠标选中列表项，也可以使用键盘来输入文本。组合框有点像一组复选框，但在没有打开列表时，只能看见其中的一个选项。

要创建 JComboBox 对象，必须在创建这种对象后加入每个选项，如下例所示：

```
JComboBox profession = new JComboBox();
FlowLayout flo = new FlowLayout();
profession.addItem("Butcher");
profession.addItem("Baker");
profession.addItem("Candlestick maker");
profession.addItem("Fletcher");
profession.addItem("Fighter");
profession.addItem("Technical writer");
setLayout(flo);
add(profession);
```

该示例创建一个 JComboBox 组件，其中包含 6 个可供用户选择的选项。被选中的选项

出现在该组件的可视区域。图 13.5 显示了弹出式列表被打开时的情况。

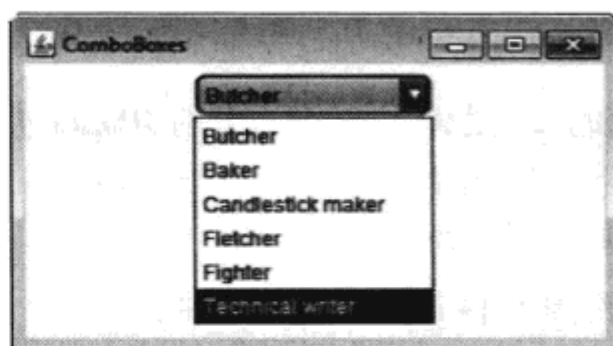


图 13.5

显示组合框组件

要让 JComboBox 组件能够接收文本输入，必须使用 true 作为参数调用其 setEditable() 方法，如下面的语句所示：

```
profession.setEditable(true);
```

必须在将组件加入到容器中之前调用该方法。

13.2.6 文本区域

JTextArea 组件允许用户输入多行文本，你可以指定该组件的宽度和高度。下面的语句创建一个 JTextArea 组件，其宽度大约为 40 个字符，高度为 8 行，然后将该组件加入到容器中：

```
JTextArea comments = new JTextArea(8, 40);
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(comments);
```

图 13.6 显示了将该组件加入到框架中的情况。

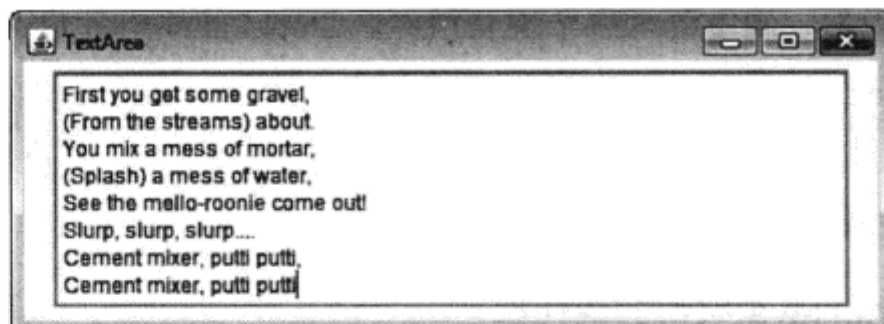


图 13.6

显示文本区域组件

可以在构造函数 JTextArea() 中指定要在文本区域显示的字符串，可以使用换行符 “\n” 进行换行，如下所示：

```
JTextArea comments = new JTextArea("I should have been a pair\n"
    + "of ragged claws.", 10, 25);
```

警告：

文本区域组件的行为可能与读者预期的不同：用户达到文本区域底部时，它自动增大，文本区域的底部和右边没有滚动条。要实现类似于其他窗口软件中的文本区域，必须将文本区域放在被称为滚动窗格的容器中，这将在第 16 章介绍。

Watch Out!

PDG

13.2.7 面板

本章要介绍的最后一种组件是面板，它是使用 Swing 中的 `JPanel` 类创建的。`JPanel` 对象是可在 Swing 界面中使用的最简单的容器，用于将显示区域划分成不同的容器组。将显示区域分成几部分后，可以采用不同的规则来组织每个部分。

可使用下面的语句来创建 `JPanel` 对象并将其加入到容器中：

```
JPanel topRow = new JPanel();
FlowLayout flo = new FlowLayout();
setLayout(flo);
add(topRow);
```

面板常用于排列界面中的组件，这将在第 14 章介绍。

通过调用面板的 `add()` 方法可以将组件添加到面板中。还可以通过调用 `setLayout()` 方法，直接给面板指定布局管理器。

当需要在界面包含绘画区域（如显示图像文件中的图像）时，也可以使用面板。

`JPanel` 的另一种用途是，用于创建可加入到其他类中的组件，这将在本章后面介绍。

13.3 创建自己的组件

面向对象编程的一个主要优点是，能够在不同的项目中重用类。在接下来的程序中，读者将创建一个特殊的面板组件，可在其他 Java 程序中重用它。该组件名为 `ClockPanel`，它像第 7 章的 `ClockTalk` 程序那样，显示当前的日期和时间。

创建用户界面组件的第一步是，确定继承哪个现有组件。`ClockPanel` 组件是 `JPanel` 的子类。

程序清单 13.3 定义了 `ClockPanel` 类，这是一个面板组件，包含一个显示当前日期和时间的标签。

在一个新的 Java 空文件中输入程序清单 13.3 中的全部文本，并保存。

程序清单 13.3 `ClockPanel.java` 完整的源代码

```
1: import javax.swing.*;
2: import java.awt.*;
3: import java.util.*;
4:
5: public class ClockPanel extends JPanel {
6:     public ClockPanel() {
7:         super();
8:         String currentTime = getTime();
9:         JLabel time = new JLabel("Time: ");
10:        JLabel current = new JLabel(currentTime);
11:        add(time);
12:        add(current);
13:    }
14:
15:    final String getTime() {
16:        String time;
```



```

17:         // get current time and date
18:         Calendar now = Calendar.getInstance();
19:         int hour = now.get(Calendar.HOUR_OF_DAY);
20:         int minute = now.get(Calendar.MINUTE);
21:         int month = now.get(Calendar.MONTH) + 1;
22:         int day = now.get(Calendar.DAY_OF_MONTH);
23:         int year = now.get(Calendar.YEAR);
24:
25:         String monthName = "";
26:         switch (month) {
27:             case (1):
28:                 monthName = "January";
29:                 break;
30:             case (2):
31:                 monthName = "February";
32:                 break;
33:             case (3):
34:                 monthName = "March";
35:                 break;
36:             case (4):
37:                 monthName = "April";
38:                 break;
39:             case (5):
40:                 monthName = "May";
41:                 break;
42:             case (6):
43:                 monthName = "June";
44:                 break;
45:             case (7):
46:                 monthName = "July";
47:                 break;
48:             case (8):
49:                 monthName = "August";
50:                 break;
51:             case (9):
52:                 monthName = "September";
53:                 break;
54:             case (10):
55:                 monthName = "October";
56:                 break;
57:             case (11):
58:                 monthName = "November";
59:                 break;
60:             case (12):
61:                 monthName = "December";
62:         }
63:         time = monthName + " " + day + ", " + year + " "
64:             + hour + ":" + minute;
65:         return time;
66:     }
67: }

```

ClockPanel 程序的 `getTime()` 方法检索当前日期和时间, 使用的技巧与第 7 章的应用程序 ClockTalk 相同。在第 15 行声明该方法时, 带有 `final` 关键字:

```

final String getTime() {
    // ...
}

```

该方法使用了 `final` 关键字后可以防止它被子类中的方法覆盖。作为 GUI 组件，这对 `ClockPanel` 是必须的。

第 6~13 行的构造函数创建面板，具体情况如下。

- 第 8 行：调用 `getTime()` 方法来检索当前日期和时间，并将返回的值存储在字符串变量 `currentTime` 中。
- 第 9 行：创建一个名为 `time` 的新标签，并将其文本设置为 “Time:”。
- 第 10 行：将 `currentTime` 用作标签组件 `current` 的文本。
- 第 11 行：调用面板的 `add()` 方法并将标签 `time` 作为参数，从而将其加入到面板中。
- 第 12 行：以相同的方式将 `current` 标签加入到面板中。

为测试该面板，创建应用程序 `ClockFrame`，其代码如程序清单 13.4 所示。

程序清单 13.4 `ClockFrame.java` 的完整源代码

```

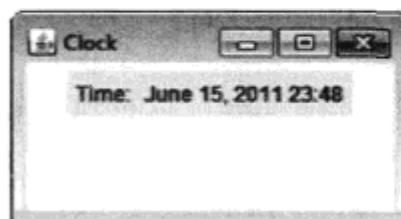
1: import java.awt.*;
2: import javax.swing.*;
3:
4: public class ClockFrame extends JFrame {
5:     public ClockFrame() {
6:         super("Clock");
7:         setLookAndFeel();
8:         setSize(225, 125);
9:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10:        FlowLayout flo = new FlowLayout();
11:        setLayout(flo);
12:        ClockPanel time = new ClockPanel();
13:        add(time);
14:        setVisible(true);
15:    }
16:
17:    private void setLookAndFeel() {
18:        try {
19:            UIManager.setLookAndFeel(
20:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
21:            );
22:        } catch (Exception exc) {
23:            // ignore error
24:        }
25:    }
26:
27:    public static void main(String[] arguments) {
28:        ClockFrame clock = new ClockFrame();
29:    }
30: }

```

运行该程序时，输出结果如图 13.7 所示。

图 13.7

显示 `ClockPanel`
组件



13.4 总结

用户期望其运行的程序有可指向并单击的可视化界面，这希望给创建软件提出了更严峻的挑战，Java 通过 Swing 向程序员提供了这种功能。Swing 提供了创建可运行的 GUI 所需的所有类，不管用户在何种操作系统中运行 Java 程序。

下一章将介绍如何使用布局管理器来设计图形用户界面，布局管理器是用于指定如何在容器排列组件的类。

13.5 问与答

问：如果没有给容器指定布局管理器，组件将如何排列？

答：在简单容器（如面板）中，默认使用 FlowLayout 来排列组件，组件像在页面中显示单词那样被加入到容器：从左到右排列，当前行没有空间后进行下一行，再按从左到右的顺序排列。下一章将讲到，框架、窗口和 applet 默认使用 GridLayout 布局。

问：为什么很多图形用户界面的类的名字都以 J 打头，比如 JFrame 和 JLabel？

答：这些类都属于 Swing 框架，该框架是 Java 类中第二种可包含图形用户界面类的方式。抽象窗口工具包（AWT）是第一种方式，它的类名要更为简单，比如 Frame 和 Label。

AWT 类属于 java.awt 包及其相关包，而 Swing 属于 javax.swing 包和相关包，因此它们有相同的类名。只所以在名字面前使用字母 J 是为了对两者进行区分。

Swing 类也称之为 Java 基础类（JFC）。

13.6 测验

经过本章的艰苦学习，如果读者的头脑还没有变成 GUI 碎片，请回答下列问题以测试自己的技能。

13.6.1 问题

1. 下列哪种用户组件可用作容纳其他组件的容器？
 - a. TupperWare。
 - b. JPanel。
 - c. Choice。
2. 在容器中必须首先做下面哪项工作？
 - a. 指定布局管理器。
 - b. 添加组件。
 - c. 无关紧要。



3. 哪个方法决定如何在容器中排列组件？

- a. `setLayout()`。
- b. `setLayoutManager()`。
- c. `setVisible()`。

13.6.2 答案

- 1. b. 可以在面板中添加组件，然后将面板加入到其他容器中，如框架。
- 2. a. 必须在添加组件前指定布局管理器，这样才能正确地加入组件。
- 3. a. 方法 `setLayout()` 接受一个参数，即决定如何排列组件的布局管理器对象。

13.7 练习

为更深入地理解 GUI 设计，请完成下面的练习：

- 修改应用程序 `SalutonFrame`，使其在框架的主要区域而不是标题栏中显示文本“`Saluton Mondo!`”；
- 创建一个包含另一个框架的框架并同时显示它们。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 14 章

用户界面的布局

本章介绍如下内容：

- 创建布局管理器；
- 为容器指定布局管理器；
- 使用面板来组织界面中的组件；
- 使用不常见的布局；
- 为 Java 应用程序创建原型。

开始为 Java 程序设计图形用户界面时，面临的一个障碍是组件会移动。容器大小发生变化时（如用户调整框架的大小时），容器中的组件将根据容器的新尺寸重新排列。

这种变化对程序员有利，因为它考虑到了界面组件在不同操作系统中的显示方式。对于同一个 Java 程序，可单击的按钮在 Windows、Linux 和 Mac 操作系统中的外观可能不同。

使用一组称为布局管理器的类来排列界面中的组件，这些类定义了组件如何在容器中显示。界面中的每个容器都有自己的布局管理器。

14.1 使用布局管理器

在 Java 中，组件在容器中的位置取决于其他组件的大小以及容器的宽度和高度。下列因素将影响按钮、文本框和其他组件的布局：

- 容器的大小；
- 其他组件和容器的大小；
- 使用的布局管理器。

可使用几种布局管理器来影响组件的显示。面板的默认布局管理器是 `FlowLayout` 类，前

一章使用的就是它，它位于 `java.awt` 包中。

使用 `FlowLayout` 时，像在页面中排列英文单词那样排列组件：从左到右排列，当前行没有空间后进入下一行。

当框架中添加近组件时，它可以使用如下代码示例来调用浮动的布局：

```
FlowLayout topLayout = new FlowLayout();
setLayout(topLayout);
```

也可以指定用于特定容器（如 `JPanel` 对象）的布局管理器，为此可以使用该容器对象的 `setLayout()` 方法。

创建一个新的 Java 空文件，将其命名为 `Crisis`，然后输入程序清单 14.1 中的全部文本，并保存。该应用程序的图形用户界面包含 5 个按钮。

程序清单 14.1 `Crisis.java` 程序的完整源代码

```
1: import java.awt.*;
2: import javax.swing.*;
3:
4: public class Crisis extends JFrame {
5:     JButton panicButton;
6:     JButton dontPanicButton;
7:     JButton blameButton;
8:     JButton mediaButton;
9:     JButton saveButton;
10:
11:     public Crisis() {
12:         super("Crisis");
13:         setLookAndFeel();
14:         setSize(348, 128);
15:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16:         FlowLayout flo = new FlowLayout();
17:         setLayout(flo);
18:         panicButton = new JButton("Panic");
19:         dontPanicButton = new JButton("Don't Panic");
20:         blameButton = new JButton("Blame Others");
21:         mediaButton = new JButton("Notify the Media");
22:         saveButton = new JButton("Save Yourself");
23:         add(panicButton);
24:         add(dontPanicButton);
25:         add(blameButton);
26:         add(mediaButton);
27:         add(saveButton);
28:         setVisible(true);
29:     }
30:
31:     private void setLookAndFeel() {
32:         try {
33:             UIManager.setLookAndFeel(
34:                 "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
35:             );
36:         } catch (Exception exc) {
37:             // ignore error
38:         }
39:     }
40: }
```

```

41:    public static void main(String[] arguments) {
42:        Crisis cr = new Crisis();
43:    }
44: }

```

图 14.1 所示为该应用程序的运行结果。

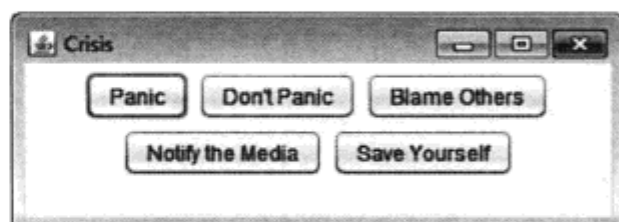


图 14.1

使用浮动布来排列组件

FlowLayout 类仅根据容器的尺寸来排列组件。调整应用程序窗口的大小时，组件将立刻重新排列。将窗口的宽度增大到原来的两倍，将发现所有 JButton 组件都显示在同一行。

14.1.1 GridLayout 管理器

GridLayout 类位于 java.awt 包中，它将容器中所有的组件组织为指定的行数和列数。分配给每个组件的显示区域都相同，因此如果指定 3 行 3 列的网格，容器将被划分成 9 个大小相等的区域。

当组件加入到容器中时，GridLayout 将所有的组件放置到网格中的某个位置，而且组件是从左向右依次添加，当这一行满了之后，在从下一行的最左边开始添加。

下面的语句创建一个容器，并将其设置为使用网格布局，该网格为 2 行 3 列：

```

GridLayout grid = new GridLayout(2, 3);
setLayout(grid);

```

图 14.2 显示了使用网格布局时，应用程序 Crisis 的外观。

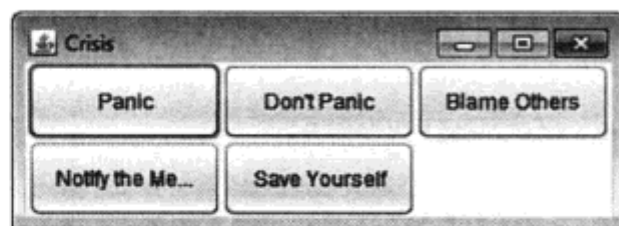


图 14.2

使用网格布局排列组件

在图 14.2 中，有些标签显示的文本被截短。如果组件容纳不下标签文本，将使用省略号 (...) 表示省略的文本。

14.1.2 BorderLayout 管理器

BorderLayout 类它也位于 java.awt 包中，它将容器中的组件放置在特定的位置，该位置有 5 个方位：东、西、南、北、中。

BorderLayout 管理器将组件放置到 5 个位置：其中 4 个位置由罗盘方向 (compass direction) 指定，另外一个由中心区域指定。当在该布局下添加组件时，add() 方法会包含第 2 个参数，用于指示组件应该放置的位置。该参数应该是 BorderLayout 类的 5 个类变量之一：NORTH、

SOUTH、EAST、WEST 和 CENTER。

与 GridLayout 类相同，BorderLayout 也会将所有可用空间都分配给组件。在周围放置 4 个边界组件后，余下的空间都分配给中央的组件，因此它通常是最大的。

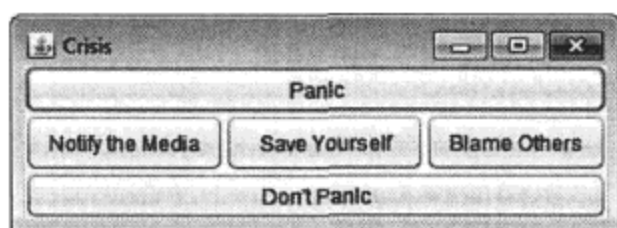
下面的语句创建一个使用边界布局的容器：

```
BorderLayout crisisLayout = new BorderLayout();
setLayout(crisisLayout);
add(panicButton, BorderLayout.NORTH);
add(dontPanicButton, BorderLayout.SOUTH);
add(blameButton, BorderLayout.EAST);
add(mediaButton, BorderLayout.WEST);
add(saveButton, BorderLayout.CENTER);
```

图 14.3 是使用这种布局时应用程序 Crisis 的外观。

图 14.3

使用边界布局排列
组件



14.1.3 BoxLayout 管理器

另一种方便的布局管理器是 BoxLayout，它位于 javax.swing 包中，它可以将组件排列成一行或一列。

使用该布局时，先创建一个放置组件的面板，然后再创建一个布局管理器，它带有 2 个参数：

- 以框式布局（box layout）组织的组件；
- BorderLayout.Y_AXIS 指定垂直排列，BoxLayout.X_AXIS 指定水平排列。

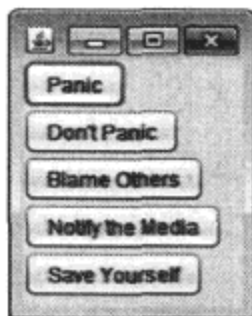
下面是用于在应用程序 Crisis 中排列组件的代码：

```
JPanel pane = new JPanel();
BoxLayout box = new BoxLayout(pane, BoxLayout.Y_AXIS);
pane.setLayout(box);
pane.add(panicButton);
pane.add(dontPanicButton);
pane.add(blameButton);
pane.add(mediaButton);
pane.add(saveButton);
add(pane);
```

图 14.4 显示了组件的排列情况。

图 14.4

使用框式布局排列
组件



14.1.4 使用 Insets 将组件隔开

在容器中排列组件时，可以使用 Insets 令组件远离容器边缘，Insets 是代表容器边界区域的对象。

Insets 类位于 java.awt 包中，它有一个接受 4 个参数的构造函数：在容器上、下、左、右留出的空间。每个参数都以像素为单位，像素是定义框架大小时使用的度量单位。

下面的语句创建一个 Insets 对象：

```
Insets around = new Insets(10, 6, 10, 3);
```

around 对象代表容器的边界：上边缘内 10 像素、左边缘内 6 像素、下边缘内 10 像素、右边缘内 3 像素。

要想在容器中使用 Insets 对象，必须覆盖容器的 getInsets() 方法。该方法不接受任何参数，并返回一个 Insets 对象，如下例所示：

```
public Insets getInsets() {
    Insets squeeze = new Insets(50, 15, 10, 15);
    return squeeze;
}
```

图 14.5 说明了这对前一个示例的影响，后者使用 BorderLayout 布局管理器，界面如图 14.1 所示。

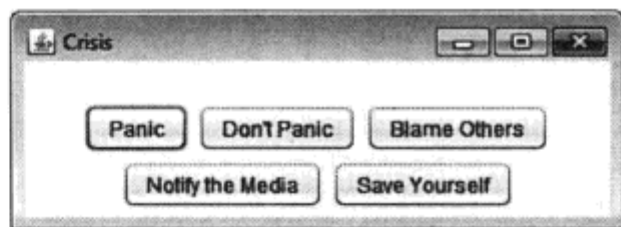


图 14.5

使用 Insets 增大组件周围的空间

在图 14.5 所示的容器中，左边框为 15 像素，下边框为 10 像素，右边框为 15 像素，上边框为 50 像素。

注意：

JFrame 容器有内置的 Inset，用于为标题栏留出空间。当覆盖 getInsets() 方法时并设置自定义值时，过小的 inset 将导致容器将组件显示在标题栏下面。

By the Way

14.2 应用程序的界面布局

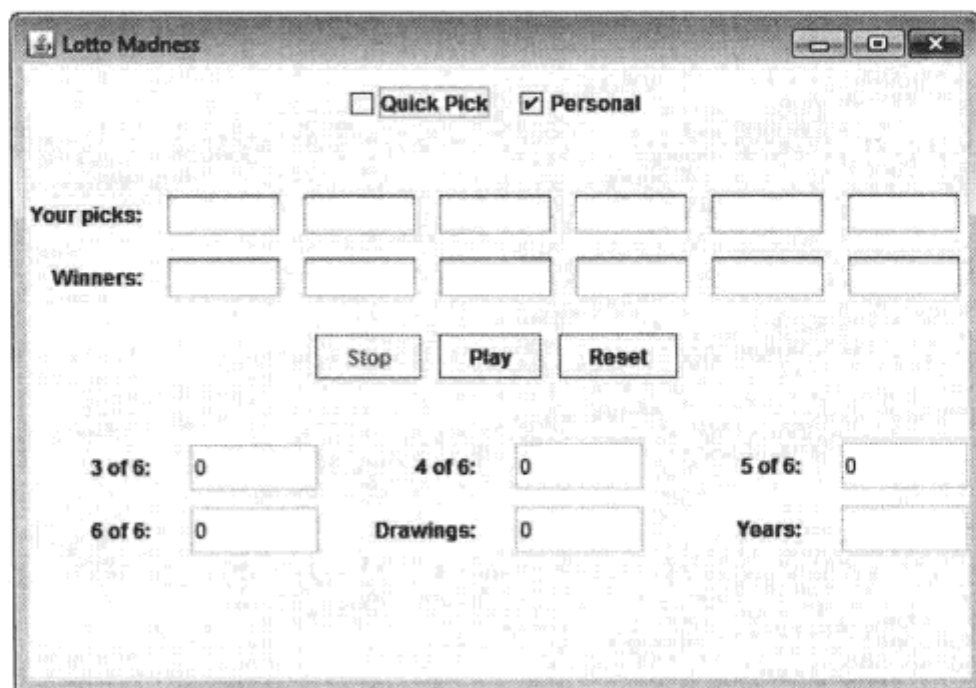
到目前为止，布局管理器应用于整个框架：调用框架的 setLayout() 方法，所有组件遵循相同的规则。这适用于有些程序，但使用 Swing 开发图形用户界面时，将经常发现这些布局管理器都不合适。

解决这种问题的方法之一是，将一组 JPanel 对象作为容器，用于放置图形用户界面的不同部分。对于其中每部分，可以使用 JPanel 对象的 setLayout() 方法设置不同的布局规则。这些面板包含需要包含的组件后，就可以将这些面板直接加入到框架中。

接下来将开发一个完整的界面，该界面可以用于你在下一章编写的程序。该程序是一个猜数游戏，确定用户一生中赢得数百万大奖的机会。它不断随机生成 6 个数，直到用户选择的数字与摇出的数字相同。图 14.6 是要开发的应用程序 GUI。

图 14.6

显示应用程序 Lotto
Madness 的图形
用户界面



创建一个新的 Java 空文件，将其命名为 LottoMadness，然后输入程序清单 14.2 中的所有文本，并保存。

程序清单 14.2 LottoMadness.java 的完整源代码

```

1: import java.awt.*;
2: import javax.swing.*;
3:
4: public class LottoMadness extends JFrame {
5:
6:     // set up row 1
7:     JPanel row1 = new JPanel();
8:     ButtonGroup option = new ButtonGroup();
9:     JCheckBox quickpick = new JCheckBox("Quick Pick", false);
10:    JCheckBox personal = new JCheckBox("Personal", true);
11:    // set up row 2
12:    JPanel row2 = new JPanel();
13:    JLabel numbersLabel = new JLabel("Your picks: ", JLabel.RIGHT);
14:    JTextField[] numbers = new JTextField[6];
15:    JLabel winnersLabel = new JLabel("Winners: ", JLabel.RIGHT);
16:    JTextField[] winners = new JTextField[6];
17:    // set up row 3
18:    JPanel row3 = new JPanel();
19:    JButton stop = new JButton("Stop");
20:    JButton play = new JButton("Play");
21:    JButton reset = new JButton("Reset");
22:    // set up row 4
23:    JPanel row4 = new JPanel();
24:    JLabel got3Label = new JLabel("3 of 6: ", JLabel.RIGHT);
25:    JTextField got3 = new JTextField("0");
26:    JLabel got4Label = new JLabel("4 of 6: ", JLabel.RIGHT);
27:    JTextField got4 = new JTextField("0");
28:    JLabel got5Label = new JLabel("5 of 6: ", JLabel.RIGHT);
29:    JTextField got5 = new JTextField("0");
30:    JLabel got6Label = new JLabel("6 of 6: ", JLabel.RIGHT);
31:    JTextField got6 = new JTextField("0", 10);
32:    JLabel drawingsLabel = new JLabel("Drawings: ", JLabel.RIGHT);

```

```

33:   JTextField drawings = new JTextField("0");
34:   JLabel yearsLabel = new JLabel("Years: ", JLabel.RIGHT);
35:   JTextField years = new JTextField();
36:
37:   public LottoMadness() {
38:       super("Lotto Madness");
39:       setLookAndFeel();
40:       setSize(550, 400);
41:       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
42:       GridLayout layout = new GridLayout(5, 1, 10, 10);
43:       setLayout(layout);
44:
45:       FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
46:           10, 10);
47:       option.add(quickpick);
48:       option.add(personal);
49:       row1.setLayout(layout1);
50:       row1.add(quickpick);
51:       row1.add(personal);
52:       add(row1);
53:
54:       GridLayout layout2 = new GridLayout(2, 7, 10, 10);
55:       row2.setLayout(layout2);
56:       row2.add(numbersLabel);
57:       for (int i = 0; i < 6; i++) {
58:           numbers[i] = new JTextField();
59:           row2.add(numbers[i]);
60:       }
61:       row2.add(winnersLabel);
62:       for (int i = 0; i < 6; i++) {
63:           winners[i] = new JTextField();
64:           winners[i].setEditable(false);
65:           row2.add(winners[i]);
66:       }
67:       add(row2);
68:
69:       FlowLayout layout3 = new FlowLayout(FlowLayout.CENTER,
70:           10, 10);
71:       row3.setLayout(layout3);
72:       stop.setEnabled(false);
73:       row3.add(stop);
74:       row3.add(play);
75:       row3.add(reset);
76:       add(row3);
77:
78:       GridLayout layout4 = new GridLayout(2, 3, 20, 10);
79:       row4.setLayout(layout4);
80:       row4.add(got3Label);
81:       got3.setEditable(false);
82:       row4.add(got3);
83:       row4.add(got4Label);
84:       got4.setEditable(false);
85:       row4.add(got4);
86:       row4.add(got5Label);
87:       got5.setEditable(false);
88:       row4.add(got5);
89:       row4.add(got6Label);
90:       got6.setEditable(false);
91:       row4.add(got6);
92:       row4.add(drawingsLabel);
93:       drawings.setEditable(false);
94:       row4.add(drawings);

```



```

95:         row4.add(yearsLabel);
96:         years.setEditable(false);
97:         row4.add(years);
98:         add(row4);
99:
100:        setVisible(true);
101:    }
102:
103:    private void setLookAndFeel() {
104:        try {
105:            UIManager.setLookAndFeel(
106:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
107:            );
108:        } catch (Exception exc) {
109:            // ignore error
110:        }
111:    }
112:
113:    public static void main(String[] arguments) {
114:        LottoMadness frame = new LottoMadness();
115:    }
116: }

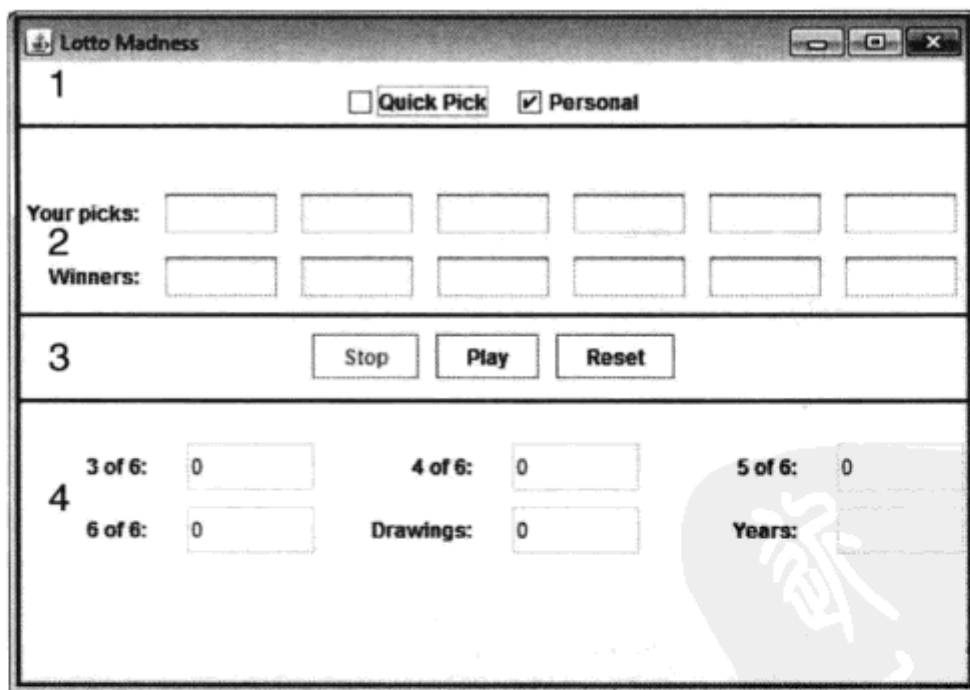
```

尽管现在还没有在该程序中添加任何语句，让其执行某些工作，但是现在也可以运行该程序，来确定图形界面的布置是否正确，是否可以收集所需要的信息。

该应用程序使用了几种不同的布局管理器。为更清楚应用程序用户界面的布局，请看图 14.7。该界面分成 4 排，各排间用水平黑线分隔。每行都是一个 JPanel 对象，应用程序使用 GridLayout 布局管理器将它们组织成 4 行 1 列。

图 14.7

将 LottoMadness 应用程序划分为不同的面板



在各排中，使用不同的布局管理器来排列组件，第 1 排和第 3 排使用 FlowLayout 对象，程序的第 45~46 行说明这些对象如何创建的：

```

FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
    10, 10);

```

在 FlowLayout() 构造函数中使用了 3 个参数。第一个参数是 FlowLayout.CENTER，指定组件应在容器（它们所属的 JPanel）中居中。后面两个参数指定组件之间的水平间距和垂直

间距，分别为 10 像素。

界面的第 2 排被设置为 2 行 7 列宽的网格。在 `GridLayout()` 构造函数中，也将组件之间的水平距离和垂直距离指定为 10 像素。程序的第 53~54 行设置了该网格：

```
GridLayout layout2 = new GridLayout(2, 7, 10, 10);
row2.setLayout(layout2);
```

界面的第 4 排使用 `GirdLayout` 将组件组织成 2 行 3 列的网格。

应用程序 `LottoMadness` 使用了本章介绍的多种组件。程序代码第 7~35 行创建界面中所有的组件对象。首先创建用于当前排的 `JPanel` 对象，然后创建这排中的组件。这段代码创建了所有的组件和容器，但调用 `add()` 方法将它们加入到应用程序的主框架中之前，这些组件和容器不会显示出来。

第 45~96 行添加组件。第 45~52 行是构造函数 `LottoMadness()`：

```
FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
    10, 10);
option.add(quickpick);
option.add(personal);
row1.setLayout(layout1);
row1.add(quickpick);
row1.add(personal);
add(row1);
```

创建完布局管理器对象后，将其作为对应 `JPanel` 对象（这里为 `row1`）的 `setLayout()` 方法的参数。指定布局后，使用 `add()` 方法将组件加入到 `JPanel` 中。放置好所有的组件后，调用 `row1` 对象的 `add()` 方法，将 `row1` 对象加入到框架中。

14.3 总结

当第一次设计 Java 程序的图形用户界面时，可能不相信组件能够移动是优点。布局管理器提供了开发引人入胜的图形用户界面的途径，它足够灵活，能够应对不同的显示方式。

下一章将更详细地介绍图形用户界面，将 `LottoMadness` 界面用于摇号，确定中奖号码。

14.4 问与答

问：在 `LottoMadness` 应用程序中，为何有些文本框为灰色，而其他文本框为白色？

答：使用 `setEditable()` 方法将有些文本框设置为不能编辑的，因此是灰色的。文本框的默认行为是，允许用户在文本框内单击并输入来修改文本框的值。然而有些文本框只是用来显示信息而不是接收用户输入。`SetEditable()` 方法可禁止用户修改不允许修改的文本框。

14.5 测验

通过回答下列问题可以检测对 Java 布局管理器的理解程度。

14.5.1 问题

1. 当将界面分成几个使用不同布局管理器的部分，通常使用下面哪种容器？
 - a. JWindow。
 - b. JPanel。
 - c. Container。
2. 面板的默认布局管理器是什么？
 - a. FlowLayout。
 - b. GridLayout。
 - c. 没有默认布局管理器。
3. BorderLayout 类因什么而得名？
 - a. 每个组件的边界。
 - b. 组件沿容器边界的排列方式。
 - c. Java 的开发者随便取的。

14.5.2 答案

1. b. JPanel 是最简单的容器。
2. a. 面板默认使用浮动布局，但框架和窗口默认使用网格布局。
3. b. 将组件加入到容器中时，必须使用 BorderLayout.WEST 和 BorderLayout.EAST 等方向变量指定其位置。

14.6 练习

如果你想继续深入了解流程（网格和边界）布局，请完成下面的实际操作部分：

- 修改应用程序 Crisis，使用一种布局管理器来组织对象 panic 和 dontPanic，使用另一种布局管理器来组织其他 3 个按钮。
- 复制文件 LottoMadness.java，然后将其重命名为 NewMadness.java。修改该程序，将 quick pick 和 personal choice 作为组合框，将按钮 start、stop 和 reset 改为复选框。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 15 章

响应用户输入

本章介绍如下内容：

- 让程序感知事件；
- 设置组件，使其引发事件；
- 忽略一些事件；
- 找出程序中事件的结束点；
- 存储界面中的信息；
- 准换存储在文本框中的值。

在前两章开发的图形用户界面无须任何修改就能独立运行。用户可以单击按钮，在文本框输入文本，调整窗口大小。然而，即使最没有识别能力的用户迟早也会提出更多的需求，因此程序提供的图形用户界面必须响应鼠标单击和键盘输入。

如果 Java 程序能够响应用户事件，就可以实现上述功能。响应用户事件通常称为事件处理，这是本章要介绍的主题。

15.1 让程序监听

在 Java 中，用户事件是这样定义的：当用户使用鼠标、键盘或其他输入设备执行某种操作时，所引发的行为。

在接收事件之前，必须知道如何让对象监听。要响应用户事件，必须使用一个或多个 `EventListener` 接口。接口是 Java 面向对象编程的一个特性，能够让类继承原本无法使用的行为。接口很像与其他类签订的合同，用来确保类包含具体的方法。

`EventListener` 接口包含的方法可以接受特定类型的用户输入信息。

要添加 `EventListener` 接口，必须完成两项工作。首先，监听类位于 `java.awt.event` 包中，

因此必须通过下面的语句使其在程序中可用：

```
import java.awt.event.*;
```

其次，必须使用关键字 `implements` 将类声明为支持一个或多个监听接口。下面的语句创建一个这样的类，即使用 `ActionListener` 接口来响应按钮和菜单单击：

```
public class Graph implements ActionListener {
```

`EventListener` 接口让图形用户界面中的组件引发用户事件。如果没有一个监听者，组件将不能做任何让程序的其他部分能够知道的事情。程序中每个要监听的组件都必须有监听者接口。要让程序响应用鼠标单击按钮或在文本框中按回车键，必须包括 `ActionListener` 接口。要对用户选择列表或复选框进行响应，需要使用 `ItemListener` 接口。

在同一个类中需要多个接口时，可在关键字 `implements` 的后面列出接口名，并用逗号将它们隔开，如下列代码所示：

```
public class Graph3D implements ActionListener, MouseListener {
    // ...
}
```

15.2 设置要监听的组件

为组件实现所需的接口后，还必须设置该组件使其生成用户事件。`ActionListener` 接口监听操作事件，比如单击按钮或按下回车键。要让 `JButton` 对象引发一个事件，可使用 `addActionListener()` 方法，如下所示：

```
JButton fireTorpedos = new JButton("Fire torpedos");
fireTorpedos.addActionListener(this);
```

这段代码创建按钮 `fireTorpedos`，然后调用其 `addActionListener()` 方法。通过将 `this` 关键字作为参数传递给 `addActionListener()` 方法，指出当前对象将接收用户事件并在需要进行处理。

By the Way

注意：

很多读者在第一次接触到 `this` 关键字时，并不懂它的含义。关键字 `this` 指的是其所在的对象。因此，如果创建一个 `LottoMadess` 类，并在一条语句中使用 `this`，则它指的是执行该语句的 `LottoMadess` 对象。

15.3 处理用户事件

当有监听者的组件引发用户事件时，将自动调用一个方法，该方法位于将监听者同组件关联起来时指定的类中。

每个监听者有不同的方法，用于接受其事件。`ActionListener` 接口将事件发送给方法 `actionPerformed()`。下面是一个简短的 `actionPerformed()` 方法示例：

```
public void actionPerformed(ActionEvent event) {
    // method goes here
}
```

程序中所有的操作事件都将发送给该方法。如果程序只有一个组件可以发送操作事件，可以将处理事件的语句放在该方法中。如果程序有多个组件可以发送操作事件，则需要指定发送事件的对象。

在这种情况下，需要将一个 `ActionEvent` 对象传递给 `actionPerformed()` 方法。有几种对象用于表示程序中发送的事件，这些类包含可用于判断事件由哪个组件引发的方法。在 `actionPerformed()` 方法中，如果 `ActionEvent` 对象名为 `event`，可使用下面的语句来确定引发事件的组件：

```
String cmd = event.getActionCommand();
```

`GetActionCommand()` 方法返回一个字符串。如果引发事件的组件是按钮，返回的字符串将是该按钮的标签；如果是文本框，返回的字符串为文本框包含的文本。`getSource()` 方法返回引发事件的对象。

可以使用下面的 `actionPerformed()` 方法来接收来自 3 个组件的事件：名为 `start` 的 `JButton` 对象、名为 `speed` 的 `JTextField` 对象和名为 `viscosity` 的 `JTextField` 对象：

```
public void actionPerformed(ActionEvent event) {
    Object source = event.getSource();
    if (source == speed) {
        // speed field caused event
    } else if (source == viscosity) {
        // viscosity caused event
    } else {
        // start caused event
    }
}
```

通过使用 `getSource()` 方法，可以确定引发任何事件的对象。

15.3.1 复选框和组合框事件

复选框和组合框需要 `ItemListener` 接口。调用组件的 `addItemListener()` 方法使其生成用户事件，例如，下面的语句创建一个名为 `superSize` 的复选框，并使其被选中或取消选中时引发用户事件：

```
JCheckBox superSize = new JCheckBox("Super Size", true);
superSize.addItemListener(this);
```

这些事件由 `itemStateChanged()` 方法接收，该方法将一个 `ItemEvent` 对象作为参数。要确定事件是由哪个对象引发的，可使用事件对象的 `getItem()` 方法。

要确定复选框是否被选中，可使用方法 `getStateChange()` 及常数 `ItemEvent.SELECTED` 和 `ItemEvent.DESELECTED`。下面是一个例子：


```
int status = item.getStateChange();
if (status == ItemEvent.SELECTED) {
    // item was selected
}
```

要确定 JComboBox 对象中选定的值, 可使用 `getItem()` 方法并将返回值转换为字符串, 如下所示:

```
Object which = item.getItem();
String answer = which.toString();
```

15.3.2 键盘事件

当一个键按下, 程序需要立即做出响应时, 它使用的是键盘事件和 `KeyListener` 接口。

第一步是调用组件的 `addKeyListener()` 方法, 注册接收按键事件的组件。该方法的参数应为实现 `KeyListener` 接口的对象, 如果是当前类, 可使用 `this` 作为参数。

处理键盘事件的对象必须实现以下 3 个方法。

- `void keyPressed(KeyEvent)`: 按下键时调用该方法。
- `void keyReleased(KeyEvent)`: 松开键时调用该方法。
- `void keyTyped(KeyEvent)`: 按下并松开键时调用该方法。

其中每个方法都将一个 `KeyEvent` 对象作为参数, 可以调用该对象的方法来更详细地了解事件。调用 `getKeyChar()` 方法可确定按下的是哪个键, 这是通过返回一个字符来指出的, 且只适用于字母、数字和标点符号。

要监视键盘上的每个键, 包括回车键、Home 键、Page Up 键和 Page Down 键, 可调用 `getKeyCode()` 方法。该方法返回一个代表键的整数, 然后将该整数作为参数调用 `getKeyText()` 方法, 从而返回一个包含键名 (如 Home、F1 等) 的 `String` 对象。

程序清单 15.1 所示的 Java 应用程序使用 `getKeyChar()` 方法在标签上显示最近按下的键。该应用程序实现了 `KeyListener` 接口, 因此包含方法 `keyTyped()`、`keyPressed()` 和 `keyReleased()`, 但只在第 21~24 行实现了 `keyTyped()` 方法。创建一个名为 `KeyViewer` 的 Java 文件, 然后输入程序清单 15.1 中的全部文本, 并保存。

程序清单 15.1 KeyView.java 的完整源代码

```
1: import javax.swing.*;
2: import java.awt.event.*;
3: import java.awt.*;
4:
5: public class KeyViewer extends JFrame implements KeyListener {
6:     JTextField keyText = new JTextField(80);
7:     JLabel keyLabel = new JLabel("Press any key in the text field.");
8:
9:     KeyViewer() {
10:         super("KeyViewer");
11:         setLookAndFeel();
12:         setSize(350, 100);
13:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14:         keyText.addKeyListener(this);
```

```

15:     BorderLayout bord = new BorderLayout();
16:     setLayout(bord);
17:     add(keyLabel, BorderLayout.NORTH);
18:     add(keyText, BorderLayout.CENTER);
19:     setVisible(true);
20: }
21:
22: public void keyTyped(KeyEvent input) {
23:     char key = input.getKeyChar();
24:     keyLabel.setText("You pressed " + key);
25: }
26:
27: public void keyPressed(KeyEvent txt) {
28:     // do nothing
29: }
30:
31: public void keyReleased(KeyEvent txt) {
32:     // do nothing
33: }
34:
35: private void setLookAndFeel() {
36:     try {
37:         UIManager.setLookAndFeel(
38:             "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
39:         );
40:     } catch (Exception exc) {
41:         // ignore error
42:     }
43: }
44:
45: public static void main(String[] arguments) {
46:     KeyViewer frame = new KeyViewer();
47: }
48: }

```

运行该程序时，其输出结果应该如图 15.1 所示。

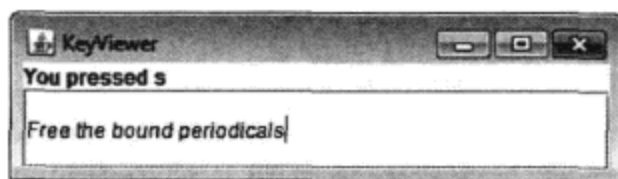


图 15.1

在程序中处理键盘事件

15.3.3 启用和禁用组件

读者可能在程序看到过与众不同的组件，它呈灰色。

灰色表示用户不能对组件执行任何操作，因为它被禁用。要在程序运行时启用和禁用组件，可以调用组件的 `setEnabled()` 方法，并将一个布尔值作为参数传递给它，因此 `setEnabled(true)` 启用组件，而 `setEnabled(false)` 禁用组件。

下面的语句创建 3 个按钮（它们的标签分别是 Previous、Next 和 Finish）并禁用第一个按钮：

```

JButton previousButton = new JButton("Previous");
JButton nextButton = new JButton("Next");
JButton finishButton = new JButton("Finish");

```

```
previousButton.setEnabled(false);
```

这将禁止组件发送用户事件。例如，编写 Java 应用程序，通过文本框接收用户地址时，可能希望用户在提供街道地址、城市、州名以及 ZIP 编码之前，禁用 Save 按钮。

15.4 完善图形应用程序

为了理解如何在 Java 程序中使用 Swing 的事件处理类，这里将完善第 14 章介绍过的摇奖模拟程序 LottoMadness。

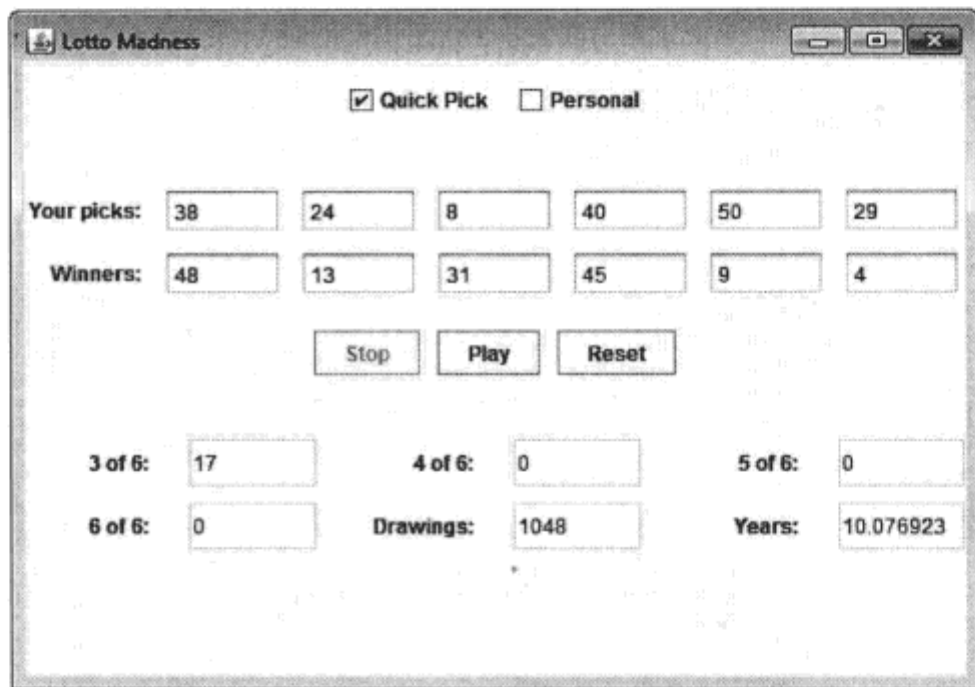
当前，LottoMadness 只是一个图形用户界面。用户可以单击按钮或在文本框中输入文本，但没有任何响应。这里将创建一个 LottoEvent 类，它接收用户输入、进行摇奖并记录用户中奖的次数。编写好这个类后，需要在 LottoMadness 中添加几行代码，以便使用 LottoEvent。通常以这种方式划分 Swing 项目：将图形用户界面放在一个类中，将事件处理方法放在另一个类中。

该应用程序旨在评估用户一生中赢六合彩的几率。图 15.2 是该程序运行时的屏幕截图。

图 15.2

运行程序

LottoMadness



这里没有使用概率论来解决这个问题，计算机使用了一种更有趣的解决方法：它不断摇奖，直到用户中奖。由于 6 个数全中几乎不可能，用户猜对其中的 3 个、4 个和 5 个数时，程序也将报告。

在创建的界面中，包含用于显示六合彩号码的 12 个文本框和两个标签分别为 Quick Pick 和 Personal 的复选框。其中有 6 个文本框是不能输入的，它们用于显示每次摇出的中奖号码；其他 6 个文本框显示用户选择的号码。当用户选中 Quick Pick 复选框时，将为用户随机显示 6 个数字；如果选中复选框 Personal，则用户可以选择所需要的数字。

3 个按钮用于控制程序：Stop、Play 和 Reset。用户单击按钮 Play 时，程序将调用线程 playing，并生成中奖号码。

用户按下 Stop 按钮时，线程停止，再按下 Reset 按钮将清除所有的文本框，以便开始新一轮游戏。第 19 章将详细讲解线程。

LottoEvent 类实现了 3 个接口：ActionListener、ItemListener 和 Runnable。Runnable 接口

与线程相关。该程序需要一个接听者来监听由应用程序的按钮和复选框生成的用户事件，但是它不需要监听与文本框相关的任何事件，因为这些文本框只用于存储用户选择的数字。用户界面将自动处理这种存储功能。

这个类需要使用 Java 类库中的两个包：主 Swing 包 `javax.swing` 和 Java 事件处理包 `java.awt.event`。

这个类包含如下两个实例变量。

- `gui`: 一个 `LottoMadness` 对象。
- `playing`: 用于不断摇奖的 `Thread` 对象。

变量 `gui` 用于同包含图形用户界面的 `LottoMadness` 对象通信。当需要修改界面或检索文本框中的值时，需要使用 `gui` 对象的实例变量。

例如，`LottoMadness` 的实例变量 `play` 代表 Play 按钮，要在 `LottoEvent` 中禁用该按钮，可使用下面的语句：

```
gui.play.setEnabled(false);
```

下面的语句可用于检索 `TextField` 对象 `got3` 的值：

```
String got3value = gui.got3.getText();
```

程序清单 15.2 列出了 `LottoEvent` 类的全部源代码。在 NetBeans 中创建一个名为 `LottoEvent` 的 Java 空文件，然后输入程序清单 15.2 中的所有文本，并保存。

程序清单 15.2 `LottoEvent.java` 的完整源代码

```
1: import javax.swing.*;
2: import java.awt.event.*;
3:
4: public class LottoEvent implements ItemListener, ActionListener,
5:     Runnable {
6:
7:     LottoMadness gui;
8:     Thread playing;
9:
10:    public LottoEvent(LottoMadness in) {
11:        gui = in;
12:    }
13:
14:    public void actionPerformed(ActionEvent event) {
15:        String command = event.getActionCommand();
16:        if (command.equals("Play")) {
17:            startPlaying();
18:        }
19:        if (command.equals("Stop")) {
20:            stopPlaying();
21:        }
22:        if (command.equals("Reset")) {
23:            clearAllFields();
24:        }
25:    }
26:
27:    void startPlaying() {
28:        playing = new Thread(this);
```

```

29:         playing.start();
30:         gui.play.setEnabled(false);
31:         gui.stop.setEnabled(true);
32:         gui.reset.setEnabled(false);
33:         gui.quickpick.setEnabled(false);
34:         gui.personal.setEnabled(false);
35:     }
36:
37:     void stopPlaying() {
38:         gui.stop.setEnabled(false);
39:         gui.play.setEnabled(true);
40:         gui.reset.setEnabled(true);
41:         gui.quickpick.setEnabled(true);
42:         gui.personal.setEnabled(true);
43:         playing = null;
44:     }
45:
46:     void clearAllFields() {
47:         for (int i = 0; i < 6; i++) {
48:             gui.numbers[i].setText(null);
49:             gui.winners[i].setText(null);
50:         }
51:         gui.got3.setText("0");
52:         gui.got4.setText("0");
53:         gui.got5.setText("0");
54:         gui.got6.setText("0");
55:         gui.drawings.setText("0");
56:         gui.years.setText("0");
57:     }
58:
59:     public void itemStateChanged(ItemEvent event) {
60:         Object item = event.getItem();
61:         if (item == gui.quickpick) {
62:             for (int i = 0; i < 6; i++) {
63:                 int pick;
64:                 do {
65:                     pick = (int) Math.floor(Math.random() * 50 + 1);
66:                 } while (numberGone(pick, gui.numbers, i));
67:                 gui.numbers[i].setText("" + pick);
68:             }
69:         } else {
70:             for (int i = 0; i < 6; i++) {
71:                 gui.numbers[i].setText(null);
72:             }
73:         }
74:     }
75:
76:     void addOneToField(JTextField field) {
77:         int num = Integer.parseInt("0" + field.getText());
78:         num++;
79:         field.setText("" + num);
80:     }
81:
82:     boolean numberGone(int num, JTextField[] pastNums, int count) {
83:         for (int i = 0; i < count; i++) {
84:             if (Integer.parseInt(pastNums[i].getText()) == num) {
85:                 return true;
86:             }
87:         }
88:         return false;
89:     }

```

```

90:
91:     boolean matchedOne(JTextField win, JTextField[] allPicks) {
92:         for (int i = 0; i < 6; i++) {
93:             String winText = win.getText();
94:             if ( winText.equals( allPicks[i].getText() ) ) {
95:                 return true;
96:             }
97:         }
98:         return false;
99:     }
100:
101:     public void run() {
102:         Thread thisThread = Thread.currentThread();
103:         while (playing == thisThread) {
104:             addOneToField(gui.drawings);
105:             int draw = Integer.parseInt(gui.drawings.getText());
106:             float numYears = (float)draw / 104;
107:             gui.years.setText("" + numYears);
108:
109:             int matches = 0;
110:             for (int i = 0; i < 6; i++) {
111:                 int ball;
112:                 do {
113:                     ball = (int) Math.floor(Math.random() * 50 + 1);
114:                 } while (numberGone(ball, gui.winners, i));
115:                 gui.winners[i].setText("" + ball);
116:                 if (matchedOne(gui.winners[i], gui.numbers)) {
117:                     matches++;
118:                 }
119:             }
120:             switch (matches) {
121:                 case 3:
122:                     addOneToField(gui.got3);
123:                     break;
124:                 case 4:
125:                     addOneToField(gui.got4);
126:                     break;
127:                 case 5:
128:                     addOneToField(gui.got5);
129:                     break;
130:                 case 6:
131:                     addOneToField(gui.got6);
132:                     gui.stop.setEnabled(false);
133:                     gui.play.setEnabled(true);
134:                     playing = null;
135:                 }
136:             try {
137:                 Thread.sleep(100);
138:             } catch (InterruptedException e) {
139:                 // do nothing
140:             }
141:         }
142:     }
143: }

```

LottoEvent 类有一个构造函数：LottoEvent (LottoMadness)，它将 LottoMadness 对象作为参数，这表明要依靠 LottoEvent 来处理用户事件和进行摇奖。

在这个类中使用下述方法来完成具体的任务。

- `clearAllField()` 方法清空应用程序中所有的文本框，它在用户单击 `Reset` 按钮时被调用。
- `addOneToField()` 方法将文本框的内容转换为整数值，将其加 1 后再转换为字符串并存储到文本框中。由于所有的文本框存储的都是字符串，要将其作为数字使用，必须采取特殊措施。
- `numberGone()` 方法接受 3 个参数——已摇出的一个数字、可存储多个 `JTextField` 对象的数组和一个计数器。该方法确保中奖号码不包含相同的数字。
- `matchedOne()` 方法接受 2 个参数——一个 `JTextField` 对象和一个包含 6 个 `JTextField` 对象的数组。该方法检查用户选中的某个数是否与当前中奖号码中的数字相同。

应用程序的 `actionPerformed()` 方法接收用户单击按钮时引发的事件。`getActionCommand()` 方法检索按钮的标签，以确定哪个按钮被单击。

单击 `Play` 按钮将调用 `startPlaying()` 方法，该方法禁用 4 个组件。单击 `Stop` 按钮将调用 `stopPlaying()` 方法，该方法启用除 `Stop` 按钮外的所有组件。

`ItemStateChanged()` 方法接受用户选择复选框 `Quick Pick` 和 `Personal` 时触发的事件。`getItem()` 方法返回一个对象，指出哪个复选框被单击。如果是复选框 `Quick Pick`，则为生成 6 个 1~50 的随机数，作为用户选择的彩票号码；否则清空用户用来输入号码的文本框。

`LottoEvent` 类使用 1~50 的数字表示摇出的小球。这是在第 113 行完成的，它将方法 `Math.random()` 返回的值乘以 50 再加 1，并将结果作为 `Math.floor()` 方法的参数，最终得到一个 1~50 的随机整数。如果将这里和第 65 行的 50 替换为其他数字，可将 `LottoMadness` 用于进行更大或更小范围的摇奖模拟。

`LottoMadness` 项目没有使用变量来记录摇出的号码、中奖次数和文本框中的彩票号码。相反，它使用界面来存储值并自动显示它们。

为完成该项目，在 `NetBeans` 中重新打开 `LottoMadness.java` 文件。只需要添加 6 行代码就可以使用 `LottoEvent` 类。

首先使用下面的语句添加一个新的实例变量，用于存储 `LottoEvent` 对象：

```
LottoEvent lotto = new LottoEvent(this);
```

接下来在构造函数 `LottoMadness()` 中，对于每个可接受用户输入的界面组件，调用其方法 `addItemListener()` 和 `addActionListener()`：

```
// Add listeners
quickpick.addItemListener(lotto);
personal.addItemListener(lotto);
stop.addActionListener(lotto);
play.addActionListener(lotto);
reset.addActionListener(lotto);
```

程序清单 15.3 列出了修改之后的 `LottoMadness.java` 的完整源代码。其中新增的代码用黑色阴影表示，其他代码与前一章相同。

程序 15.3 `LottoMadness.java` 的完整源代码

```
1: import java.awt.*;
2: import javax.swing.*;
```

```

3:
4: public class LottoMadness extends JFrame {
5:     LottoEvent lotto = new LottoEvent(this);
6:
7:     // set up row 1
8:     JPanel row1 = new JPanel();
9:     ButtonGroup option = new ButtonGroup();
10:    JCheckBox quickpick = new JCheckBox("Quick Pick", false);
11:    JCheckBox personal = new JCheckBox("Personal", true);
12:    // set up row 2
13:    JPanel row2 = new JPanel();
14:    JLabel numbersLabel = new JLabel("Your picks: ", JLabel.RIGHT);
15:    JTextField[] numbers = new JTextField[6];
16:    JLabel winnersLabel = new JLabel("Winners: ", JLabel.RIGHT);
17:    JTextField[] winners = new JTextField[6];
18:    // set up row 3
19:    JPanel row3 = new JPanel();
20:    JButton stop = new JButton("Stop");
21:    JButton play = new JButton("Play");
22:    JButton reset = new JButton("Reset");
23:    // set up row 4
24:    JPanel row4 = new JPanel();
25:    JLabel got3Label = new JLabel("3 of 6: ", JLabel.RIGHT);
26:    JTextField got3 = new JTextField("0");
27:    JLabel got4Label = new JLabel("4 of 6: ", JLabel.RIGHT);
28:    JTextField got4 = new JTextField("0");
29:    JLabel got5Label = new JLabel("5 of 6: ", JLabel.RIGHT);
30:    JTextField got5 = new JTextField("0");
31:    JLabel got6Label = new JLabel("6 of 6: ", JLabel.RIGHT);
32:    JTextField got6 = new JTextField("0", 10);
33:    JLabel drawingsLabel = new JLabel("Drawings: ", JLabel.RIGHT);
34:    JTextField drawings = new JTextField("0");
35:    JLabel yearsLabel = new JLabel("Years: ", JLabel.RIGHT);
36:    JTextField years = new JTextField("0");
37:
38:    public LottoMadness() {
39:        super("Lotto Madness");
40:        setLookAndFeel();
41:        setSize(550, 270);
42:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
43:        GridLayout layout = new GridLayout(5, 1, 10, 10);
44:        setLayout(layout);
45:
46:        // Add listeners
47:        quickpick.addItemListener(lotto);
48:        personal.addItemListener(lotto);
49:        stop.addActionListener(lotto);
50:        play.addActionListener(lotto);
51:        reset.addActionListener(lotto);
52:
53:        FlowLayout layout1 = new FlowLayout(FlowLayout.CENTER,
54:            10, 10);
55:        option.add(quickpick);
56:        option.add(personal);
57:        row1.setLayout(layout1);
58:        row1.add(quickpick);
59:        row1.add(personal);
60:        add(row1);
61:
62:        GridLayout layout2 = new GridLayout(2, 7, 10, 10);
63:        row2.setLayout(layout2);

```

```

64:         row2.add(numbersLabel);
65:         for (int i = 0; i < 6; i++) {
66:             numbers[i] = new JTextField();
67:             row2.add(numbers[i]);
68:         }
69:         row2.add(winnersLabel);
70:         for (int i = 0; i < 6; i++) {
71:             winners[i] = new JTextField();
72:             winners[i].setEditable(false);
73:             row2.add(winners[i]);
74:         }
75:         add(row2);
76:
77:         FlowLayout layout3 = new FlowLayout(FlowLayout.CENTER,
78:             10, 10);
79:         row3.setLayout(layout3);
80:         stop.setEnabled(false);
81:         row3.add(stop);
82:         row3.add(play);
83:         row3.add(reset);
84:         add(row3);
85:
86:         GridLayout layout4 = new GridLayout(2, 3, 20, 10);
87:         row4.setLayout(layout4);
88:         row4.add(got3Label);
89:         got3.setEditable(false);
90:         row4.add(got3);
91:         row4.add(got4Label);
92:         got4.setEditable(false);
93:         row4.add(got4);
94:         row4.add(got5Label);
95:         got5.setEditable(false);
96:         row4.add(got5);
97:         row4.add(got6Label);
98:         got6.setEditable(false);
99:         row4.add(got6);
100:        row4.add(drawingsLabel);
101:        drawings.setEditable(false);
102:        row4.add(drawings);
103:        row4.add(yearsLabel);
104:        years.setEditable(false);
105:        row4.add(years);
106:        add(row4);
107:
108:        setVisible(true);
109:    }
110:
111:    private void setLookAndFeel() {
112:        try {
113:            UIManager.setLookAndFeel(
114:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
115:            );
116:        } catch (Exception exc) {
117:            // ignore error
118:        }
119:    }
120:
121:    public static void main(String[] arguments) {
122:        LottoMadness frame = new LottoMadness();
123:    }
124: }

```

在添加完阴影显示的代码行之后，可以运行该程序，它可以测试你玩彩票的技能。正如读者预期的，这种练习毫无意义，因为在一生中赢得六合彩的机会非常小，即使你活得和圣经中的人物一样长。

提示：

本书的配套网站 www.java24hours.com 包含了 applet 版本 LottoMadness 程序的链接。本书付印时，摇了 410732244 次奖，这相当于 39000000 年（每两周摇一次）。其中 6 个数字中猜对 3 个的有 6364880 次，猜对 4 个有 337285 次，猜对 5 个的有 6476 次，全部猜对的有 51 次。Bill Teer 于 2000 年 8 月 14 日中了该虚拟六合彩，这是在这个 applet 在网上发布后的 4 年多之后发生的。他的中奖号码是 3、7、1、15、34 和 43，是在下了 241225 注（相当于 2319.47 年）后中的。

**Did you
Know?**

15.5 总结

通过使用 Swing，只需进行少量的编程就可以创建专业级程序。虽然应用程序 LottoMadness 比前 14 章创建的很多示例程序都长，但有一半语句是用于创建界面的。

如果花些时间运行该程序，将更加羡慕六合彩得主的好运气。

我最近运行该程序的结果表明，我挥霍 27000 美元和人生中美好的 266 年买奖票，却只有几注是 6 中 4 和 6 中 3 的。与这样的几率相比，通过选择 Java 编程技能来赚钱无疑更现实。

15.6 问与答

问：为反映文本框变化，需要使用方法 `paint()` 或 `repaint()` 吗？

答：使用文本组件的 `setText()` 方法修改其值后，什么也不需要做。Swing 负责处理必要的更新，以显示新的值。

问：为何经常同时导入一个包及其子包，例如，在程序清单 15.1 中导入 `java.awt.*` 和 `java.awt.event.*`。难道第一条语句没有覆盖第二条吗？

答：虽然从名称上看，`java.awt` 和 `java.awt.event` 是相关的，但在 Java 中包是不能继承的。一个包不可能是另一个包的子包。

在 `import` 语句中使用星号实际上是将包中所有的类导入。

星号只适用于类而不适用于包，因此单条 `import` 语句最多只能导入单个包中的所有类。

15.7 测验

如果 LottoMadness 程序激起了你玩游戏的热情，请回答下列问题以检测玩游戏的技能。

15.7.1 问题

1. 操作事件因何而得名?
 - a. 它们因响应其他事情而发生。
 - b. 表示为响应而采取某种操作。
 - c. 对电影中的冒险家 Action Jackson 表示敬意。
2. 作为方法 `AddActionListener()` 的参数时, `this` 表示什么?
 - a. 有事件发生时, 应使用该监听者。
 - b. 该事件优先于其他事件。
 - c. 该对象将处理事件。
3. 下面哪个组件将用户输入作为整数存储?
 - a. `JButton`。
 - b. `TextField` 或 `TextArea`。
 - c. 都不是。

15.7.2 答案

1. b. 操作事件包括单击按钮和从下拉菜单中选择一个菜单项。
2. c. 关键字 `this` 表示当前对象。如果使用的是对象名而不是关键字 `this`, 该对象将接收并处理事件。
3. c. `TextField` 和 `TextArea` 组件都将其值存储为文本, 因此要将其值作为整数、浮点数或其他非文本值使用时, 必须进行转换。

15.8 练习

如果本章的主要事件没有满足你的胃口, 请完成下列练习:

- 在应用程序 `LottoMadness` 中添加一个文本框, 它与 `LottoEvent` 类的 `Thread.sleep()` 协同工作, 以降低摇奖速度;
- 修改 `LottoMadness` 项目, 使其摇出 5 个 1~90 的数字。

有关为完成这些练习而编写的 Java 程序, 请访问本书的配套网站 www.java24hours.com。

第 16 章

创建复杂的用户界面

本章介绍如下内容：

- 水平和竖直滚动组件；
- 使用滑块输入数值；
- 监视用户通过滑块进行的输入；
- 创建图像图标和工具栏。

使用 Swing 创建图形用户界面时，除学习如何使用不同的界面组件、布局管理器和事件处理方法外，还必须熟悉 Swing 提供的所有功能。

Swing 包含 400 多个不同的类，这使其成为 Java 中的最大类库。其中的很多类可用前三章介绍的技术来实现——所有 Swing 容器和组件有相同的超类，这使它们的行为相同。

本章将介绍可在 Swing 程序中使用的其他组件。

16.1 滚动窗格

图形用户界面中的组件常常大于用于显示它们的区域，要从组件的一部分移动到另一部分，可使用垂直和水平滚动条。

在 Swing 中，通过将组件加入到滚动窗格中来提供滚动功能。滚动窗格是一种容器，用 Swing 中的 `JScrollPane` 类表示。

可以使用下面的构造函数来创建滚动窗格。

- `JScrollPane()`：创建必要时显示垂直和水平滚动条的滚动窗格。
- `JScrollPane(int, int)`：创建包含指定的垂直和水平滚动条的滚动窗格。
- `JScrollPane(Component)`：创建包含指定用户界面组件的滚动窗格。
- `JScrollPane(Component, int, int)`：创建包含指定组件以及垂直和水平滚动条的滚动窗格。

在构造函数中使用整型参数指定如何在窗格中使用滚动条，可将使用下述类变量作为参数：

- JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED 或 JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED。
- JScrollPane.VERTICAL_SCROLLBAR_NEVER 或 JScrollPane.HORIZONTAL_SCROLLBAR_NEVER。
- JScrollPane.VERTICAL_SCROLLBAR_ALWAYS 或 JScrollPane.VERTICAL_SCROLLBAR_ALWAYS。

如果创建了一个没有包含组件的滚动窗格，可以使用窗格的 `add(Component)` 方法来添加组件。设置好滚动窗格后，应将其像组件那样加入到容器中。

来看一个包含滚动窗格的应用程序，创建一个名为 `MailWriter` 的 Java 空文件，输入程序清单 16.1 中的文本文件，并保存。

程序清单 16.1 MailWriter.java 程序的完整源代码

```

1: import javax.swing.*;
2: import java.awt.*;
3:
4: public class MailWriter extends JFrame {
5:
6:     public MailWriter() {
7:         super("Write an E-Mail");
8:         setLookAndFeel();
9:         setSize(370, 270);
10:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11:        FlowLayout flow = new FlowLayout(FlowLayout.RIGHT);
12:        setLayout(flow);
13:
14:        JPanel row1 = new JPanel();
15:        JLabel toLabel = new JLabel("To:");
16:        row1.add(toLabel);
17:        JTextField to = new JTextField(24);
18:        row1.add(to);
19:        add(row1);
20:
21:        JPanel row2 = new JPanel();
22:        JLabel subjectLabel = new JLabel("Subject:");
23:        row2.add(subjectLabel);
24:        JTextField subject = new JTextField(24);
25:        row2.add(subject);
26:        add(row2);
27:
28:        JPanel row3 = new JPanel();
29:        JLabel messageLabel = new JLabel("Message:");
30:        row3.add(messageLabel);
31:        JTextArea message = new JTextArea(4, 22);
32:        message.setLineWrap(true);
33:        message.setWrapStyleWord(true);
34:        JScrollPane scroll = new JScrollPane(message,
35:            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
36:            JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
37:        row3.add(scroll);
38:        add(row3);

```



```

39:
40:     JPanel row4 = new JPanel();
41:     JButton send = new JButton("Send");
42:     row4.add(send);
43:     add(row4);
44:
45:     setVisible(true);
46: }
47:
48: private void setLookAndFeel() {
49:     try {
50:         UIManager.setLookAndFeel(
51:             "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
52:         );
53:     } catch (Exception exc) {
54:         // ignore error
55:     }
56: }
57:
58: public static void main(String[] arguments) {
59:     MailWriter mail = new MailWriter();
60: }
61: }

```

运行该程序时，将看到图 16.1 所示的窗口。

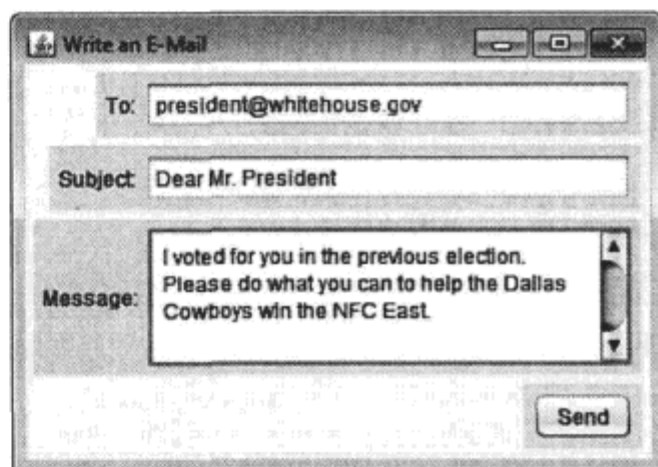


图 16.1

在应用程序中显示可滚动的文本区域

应用程序 Mailwriter 是一个用于编写电子邮件的图形用户界面，其中没有事件处理代码，因此无法对输入的数据执行任何操作。

电子邮件的文本被输入到一个滚动文本区域，后者是使用下面的语句实现的：

```

JTextArea message = new JTextArea(4, 22);
message.setLineWrap(true);
message.setWrapStyleWord(true);
JScrollPane scroll = new JScrollPane(message,
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
row3.add(scroll);

```

16.2 滑块

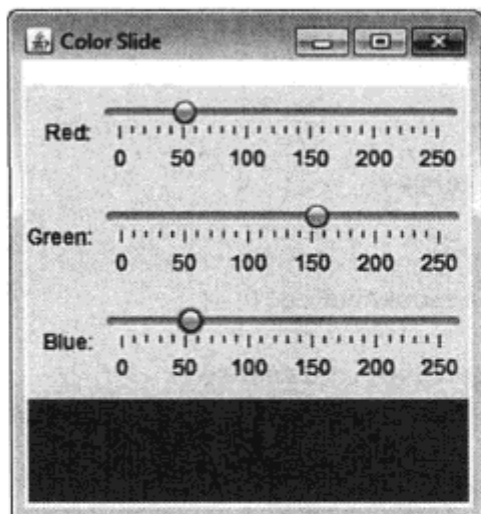
对用户来说，输入数字的最简单方法之一是使用滑块，这是一个可上下或左右拖曳的组件。在 Swing 中，滑块用 JSlider 类表示。

滑块用于从指定范围选择一个数字，这些值可显示在标签上，包括最小值、最大值和中

间值（如图 16.2 所示）。

图 16.2

使用 3 个滑块组件
选择颜色



可使用下面的构造函数之一创建水平滑块。

- `JSlider()`: 创建最小值为 0，最大值为 100，初始值为 50 的滑块。
- `JSlider(int, int)`: 创建具有指定最小值和最大值的滑块。
- `JSlider(int, int, int)`: 创建具有指定最小值、最大值和初始值的滑块。

要创建垂直滑块，必须使用接受另外一个附加参数（滑块方向）的构造函数，该参数应设置为类变量 `JSlider.VERTICAL` 或 `JSlider.HORIZONTAL`。

下面的语句创建一个垂直滑块，可用于选择 1~1000 的数字：

```
JSlider guess = new JSlider(JSlider.VERTICAL, 1, 1000, 500);
```

在该滑块中，用于选择数字标记最初位于 500 处。

要在滑块上显示标签，必须设置标签包含的信息。调用滑块的 `setMajorTickSpacing(int)` 和 `setMinorTickSpacing(int)` 方法指定标签上刻度的密度。主刻度使用的线条比次刻度粗。

在设置了刻度的密度之后，可以使用 `true` 作为参数来调用滑块的 `setPaintTicks(boolean)` 方法。还可以使用 `true` 作为参数来调用滑块的 `setPaintLabels(boolean)` 方法。

16.3 变更监听器

为了监视用户使用滑块进行的输入，必须有实现了 `ChangeListener` 接口的类，该接口位于 `javax.swing.event` 包中。该接口只包含一个方法，格式如下：

```
public void stateChanged(ChangeEvent event); {
    // statements to handle the event
}
```

要将对象注册为变更监听器，可调用滑块所属的容器的 `addChangeListener(Object)` 方法。滑块移动时，将调用监听对象的 `stateChanged()` 方法。

该方法将 `ChangeEvent` 对象作为参数，用于指出其值发生了变化的滑块组件。调用该对象的 `getSource()` 方法并将返回的对象转换为 `JSlider`，如下面的语句所示：

```
JSlider changedSlider = (JSlider) event.getSource();
```

在本例中，`event` 是 `ChangeEvent` 对象，用作方法 `stateChanged()` 的参数。

滑块移动时，将发生变更事件。该事件在滑块开始移动时发生，直到用户松开滑块为止。因此，可能要等到滑块停止移动后才调用 `stateChange()` 方法。

要确定滑块是否在移动，可调用其 `getValueIsAdjusting()` 方法。如果滑块在移动，该方法返回 `true`，否则返回 `false`。

接下来的项目将演示这种技巧，这是一个使用 3 个滑块来选择颜色的 Java 应用程序。在 Java 中，颜色是使用 `Color` 类创建的，这个类位于 `java.awt` 包中。

一种创建 `Color` 对象的方法是指定颜色中红、绿和蓝分量的值。每个值都可以是 0~255 的整数，255 表示最大。

例如，下面的语句创建一个代表奶油糖色的 `Color` 对象：

```
Color butterscotch = new Color(255, 204, 128);
```

创建该 `Color` 对象时使用的红色分量为 255，因此包含最大的红色分量。它还包含较大绿色分量和蓝色分量。

颜色也可表示为特定范围内的值，因此适合用滑块组件来表示。

程序清单 16.2 所示的 `ColorSlider` 应用程序包含 3 个滑块、用于标记滑块的 3 个标签以及一个用于显示颜色的面板。创建一个名为 `ColorSliders` 的 Java 空文件，然后输入程序清单 16.2 中的全部文本，并保存。

程序清单 16.2 `ColorsSliders.java` 程序的完整源代码

```
1: import javax.swing.*;
2: import javax.swing.event.*;
3: import java.awt.*;
4:
5: public class ColorSliders extends JFrame implements ChangeListener {
6:     ColorPanel canvas;
7:     JSlider red;
8:     JSlider green;
9:     JSlider blue;
10:
11:     public ColorSliders() {
12:         super("Color Slide");
13:         setLookAndFeel();
14:         setSize(270, 300);
15:         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16:         setVisible(true);
17:
18:         canvas = new ColorPanel();
19:         red = new JSlider(0, 255, 255);
20:         green = new JSlider(0, 255, 0);
21:         blue = new JSlider(0, 255, 0);
22:
23:         red.setMajorTickSpacing(50);
24:         red.setMinorTickSpacing(10);
25:         red.setPaintTicks(true);
26:         red.setPaintLabels(true);
27:         red.addChangeListener(this);
28:
29:         green.setMajorTickSpacing(50);
```

```

30:         green.setMinorTickSpacing(10);
31:         green.setPaintTicks(true);
32:         green.setPaintLabels(true);
33:         green.addChangeListener(this);
34:
35:         blue.setMajorTickSpacing(50);
36:         blue.setMinorTickSpacing(10);
37:         blue.setPaintTicks(true);
38:         blue.setPaintLabels(true);
39:         blue.addChangeListener(this);
40:
41:         JLabel redLabel = new JLabel("Red: ");
42:         JLabel greenLabel = new JLabel("Green: ");
43:         JLabel blueLabel = new JLabel("Blue: ");
44:         GridLayout grid = new GridLayout(4, 1);
45:         FlowLayout right = new FlowLayout(FlowLayout.RIGHT);
46:         setLayout(grid);
47:
48:         JPanel redPanel = new JPanel();
49:         redPanel.setLayout(right);
50:         redPanel.add(redLabel);
51:         redPanel.add(red);
52:         add(redPanel);
53:
54:         JPanel greenPanel = new JPanel();
55:         greenPanel.setLayout(right);
56:         greenPanel.add(greenLabel);
57:         greenPanel.add(green);
58:         add(greenPanel);
59:
60:         JPanel bluePanel = new JPanel();
61:         bluePanel.setLayout(right);
62:         bluePanel.add(blueLabel);
63:         bluePanel.add(blue);
64:         add(bluePanel);
65:         add(canvas);
66:
67:         setVisible(true);
68:     }
69:
70:     public void stateChanged(ChangeEvent event) {
71:         JSlider source = (JSlider) event.getSource();
72:         if (source.getValueIsAdjusting() != true) {
73:             Color current = new Color(red.getValue(),
74:                                     green.getValue(),
75:                                     blue.getValue());
76:             canvas.changeColor(current);
77:             canvas.repaint();
78:         }
79:     }
80:
81:     public Insets getInsets() {
82:         Insets border = new Insets(45, 10, 10, 10);
83:         return border;
84:     }
85:
86:     private void setLookAndFeel() {
87:         try {
88:             UIManager.setLookAndFeel(
89:                 "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"

```

```

90:         } catch (Exception exc) {
91:             // ignore error
92:         }
93:     }
94:
95:     public static void main(String[] arguments) {
96:         ColorSliders cs = new ColorSliders();
97:     }
98: }
99:
100: class ColorPanel extends JPanel {
101:     Color background;
102:
103:     ColorPanel() {
104:         background = Color.red;
105:     }
106:
107:     public void paintComponent(Graphics comp) {
108:         Graphics2D comp2D = (Graphics2D) comp;
109:         comp2D.setColor(background);
110:         comp2D.fillRect(0, 0, getSize().width, getSize().height);
111:     }
112:
113:     void changeColor(Color newBackground) {
114:         background = newBackground;
115:     }
116: }

```

运行该程序时，将会如图 16.2 所示。其中，一个框架包含 3 个表示红、绿、蓝分量的滑块以及框架底部的面板。

调整每个滑块的值，以修改显示的颜色。在图 16.2 中，该应用程序用于创建 North Texas 中间绿，其红色分量为 50，绿色分量为 150，蓝色分量为 50。这种颜色激励北德克萨斯大学的学生和校友要超越自己，并做出可怕的鹰爪手势表示将恐惧留给对手。

16.4 使用图像图标和工具栏

为改善图形用户界面的视觉效果，最简单的一种方法是使用图标（小型图像）来标识按钮和界面的其他部分。

Swing 类库中有很多组件，可以使用图像（而不是文本）来标记组件，为此可使用 `javax.swing` 包中的 `ImageIcon` 类。

可以通过调用构造函数 `ImageIcon(String)`，使用计算机中的文件来创建 `ImageIcon`。该方法的参数可以是文件名，也可以是文件的位置和名称，所下面的示例所示：

```

ImageIcon stopSign = new ImageIcon("stopsign.gif");
ImageIcon saveFile = new ImageIcon("images/savefile.gif");

```

警告：

尽管有些操作系统使用字符 \ 来分隔文件夹和文件名，但 `ImageIcon` 的构造函数要求使用字符 /。

**Watch
Out!**

用于创建图像图标的图形文件必须为 GIF、JPEG 或 PNG 格式。大多数为 GIF 格式，因为它非常适合用于显示只有几种颜色的小图形。

`ImageIcon` 的构造函数将立即从文件中加载整个图像。

可以通过调用构造函数 `JLabel (ImageIcon)` 和 `JButton(ImageIcon)`，将图标用于标签和按钮，如下例所示：

```
ImageIcon siteLogo = new ImageIcon("siteLogo.gif");
JLabel logoLabel = new JLabel(siteLogo);
ImageIcon searchWeb = new ImageIcon("searchGraphic.gif");
JButton search = new JButton(searchWeb);
```

有几种组件可同时包含图标和文本，下面的语句创建一个同时包含文本和图标的按钮：

```
JButton refresh = new JButton("Refresh",
    "images/refreshIcon.gif");
```

图像图标经常用在工具栏中，工具栏是一种将多个组件放在单行或单列中的容器。

工具栏是使用 `JToolBar` 类创建的，可被设计为允许用户在图形用户界面中移动它们，这被称为“停靠”，而这种组件也称为“可停靠的工具栏”。

可以调用下面的构造函数之一来创建工具栏。

- `JToolBar()`：创建沿水平方向排列组件的工具栏。
- `JToolBar(int)`：创建沿指定方向排列组件的工具栏，指定的方向可以是 `SwingConstants.HORIZONTAL` 或 `SwingConstants.VERTICAL`。

将组件加入到工具栏的方式与加入到其他容器相同：调用 `add(Component)` 方法并将要加入的组件作为参数。

对于可停靠的工具栏，必须放在使用 `BorderLayout` 布局管理器的容器中。这种布局将容器划分为北、南、东、西、中 5 个区域。然而，在使用可停靠的工具栏时，容器应只使用其中的两个区域：中央区域和一个方向区域。

工具栏应加入到方向区域中，下面的语句创建一个可停靠的垂直工具栏，其中包含 3 个图标按钮：

```
BorderLayout border = new BorderLayout();
pane.setLayout(border);
JToolBar bar = new JToolBar(SwingConstants.VERTICAL);
ImageIcon play = new ImageIcon("play.gif");
JButton playButton = new JButton(play);
ImageIcon stop = new ImageIcon("stop.gif");
JButton stopButton = new JButton(stop);
ImageIcon pause = new ImageIcon("pause.gif");
JButton pauseButton = new JButton(pause);
bar.add(playButton);
bar.add(stopButton);
bar.add(pauseButton);
add(bar, BorderLayout.WEST);
```

接下来将创建的项目 `Tool` 是一个 Java 应用程序，它包含图像图标和一个可停靠的工具栏。创建一个名为 `Tool` 的 Java 空文件，然后输入程序清单 16.3 中的全部文本，并保存。

程序清单 16.3 Tool.java 程序的完整源代码

```

1: import java.awt.*;
2: import java.awt.event.*;
3: import javax.swing.*;
4:
5: public class Tool extends JFrame {
6:     public Tool() {
7:         super("Tool");
8:         setLookAndFeel();
9:         setSize(370, 200);
10:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11:
12:        // build toolbar buttons
13:        ImageIcon image1 = new ImageIcon("newfile.gif");
14:        JButton button1 = new JButton(image1);
15:        ImageIcon image2 = new ImageIcon("openfile.gif");
16:        JButton button2 = new JButton(image2);
17:        ImageIcon image3 = new ImageIcon("savefile.gif");
18:        JButton button3 = new JButton(image3);
19:
20:        // build toolbar
21:        JToolBar bar = new JToolBar();
22:        bar.add(button1);
23:        bar.add(button2);
24:        bar.add(button3);
25:
26:        // build text area
27:        JTextArea edit = new JTextArea(8, 40);
28:        JScrollPane scroll = new JScrollPane(edit);
29:
30:        // create frame
31:        BorderLayout border = new BorderLayout();
32:        setLayout(border);
33:        add("North", bar);
34:        add("Center", scroll);
35:        setVisible(true);
36:    }
37:
38:    private void setLookAndFeel() {
39:        try {
40:            UIManager.setLookAndFeel(
41:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
42:            );
43:        } catch (Exception exc) {
44:            // ignore error
45:        }
46:    }
47:
48:    public static void main(String[] arguments) {
49:        Tool frame = new Tool();
50:    }
51: }

```

应用程序 Tool 需要 3 个图形文件用于创建工具栏中的图标：newfile.gif、openfile.gif 和 savefile.gif。从本书的配套网站 www.java24hous.com 中找到第 16 章的站点页面，然后下载这 3 个文件，并保存到 Java24 项目文件夹中（或者是你自己在 NetBeans 中指定的 Java 项目文件夹）。

Did you know?**提示:**

没有找到这个文件夹？在 NetBeans 中打开一个新的项目：选择 File->New File，在弹出的对话框中，将“分类 (category)”设置为 Java，将“项目类型 (project type)”设置为 Java application，然后单击 Next 按钮。The Project Location 文本框应该会包含文件夹 (用来存放这些图标) 的位置。

图 16.3 和图 16.4 是该程序运行时的两个屏幕截图，工具栏已从初始位置 (见图 16.3) 移到界面的另一边 (见图 16.4)。

图 16.3

使用包含工具栏的
应用程序

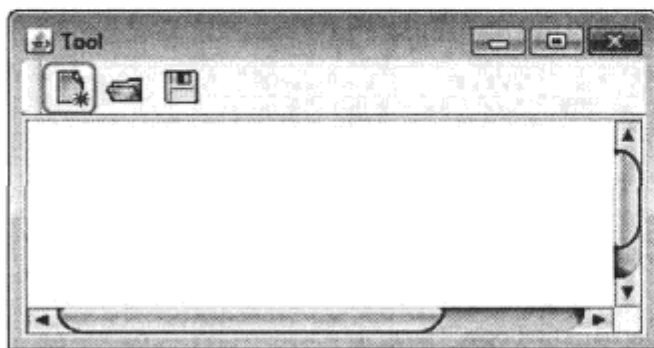


图 16.4

将工具栏停靠到
其他位置



编译并运行该应用程序，然后尝试移动工具栏。按住工具栏的手柄将其拖曳到文本区域的另一边。松开鼠标后，工具栏将沿文本区域的边缘放置，而文本区域将自动调整位置为工具栏腾出空间。

By the Way**注意:**

也可以将可停靠的工具栏拖曳到用户界面外，这将打开一个包含工具栏的新窗口。

Oracle 提供了一个图标库，可以在程序中使用这些图标。本章专题中使用的 3 个图标也来自这个图标库。要查看这些图形，可访问 <http://java.sun.com/developer/techDocs/hi/repository/>。

16.5 总结

本章是介绍 Swing 的四章中的最后一章，Swing 是 Java 语言中支持 GUI 软件的部分。

尽管 Swing 是 Java 类库的最大组成部分，但其中大部分类的用法相似。在知道如何创建组件、如何将组件添加到容器、如何设置容器的布局管理器及如何响应用户输入后，探索该语言时就能够使用众多新的 Swing 类。

16.6 问与答

问：如何了解 Java 类库中的其他 Swing 类？

答：在 Oracle 的 Java 官方网站中，有完整的 Java 类库文档，其网址为 <http://download.oracle.com/javase/7/docs/api>。在这里可以查看前面 4 章介绍的 javax.swing、java.swt 和 java.awt.event 包中的类。所有 Swing 类和接口都有文档，包括它们的构造函数、类变量和实例变量。

16.7 测验

一分耕耘一分收获：回答下面有关滚动窗格、图像图标和其他 Swing 特性的问题，以锻炼你的大脑。

16.7.1 问题

1. ImageIcon 类支持哪些图形文件格式？
 - a. GIF。
 - b. GIF 和 JPEG。
 - c. GIF、PNG 和 JPEG。
2. JSlider 对象的 getValueIsAdjusting() 方法有什么功能？
 - a. 判断滑块的值是否改变？
 - b. 判断滑块的值是否正在改变？
 - c. 不做什么，提供该方法主要是对超类不满意。

16.7.2 答案

1. c. 从 Java 1.3 开始，ImageIcon 开始支持 PNG 格式。
2. b. 如果滑块正在移动，getValueIsAdjusting() 方法将返回 true，否则返回 false。

16.8 练习

为测试是否掌握了 Swing，请完成下面的练习：

- 创建一个图形用户界面，在滚动窗格中包含一个组合框；
- 给应用程序 MailWriter 添加事件处理代码，当用户单击 Send 按钮时，使用 System.out.println() 显示组件 to、subject 和 message 的内容。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 17 章

创建交互式 Web 程序

本章介绍如下内容：

- 启动和停止 applet;
- 将 applet 放到 Web 页面中;
- 在 Web 页面中使用参数来自定义 applet;
- 在应用程序中显示 Web 页面。

作为一种可以在不同平台（比如手机、Web 服务器、Internet 设备）上运行的通用语言，Java 取得了巨大的成功。当 Java 语言在 20 世纪 90 年代中期刚刚面世时，它还是第一款可以在 Web 浏览器上运行的编程语言。

applet 是作为 Web 页面的一部分运行的 Java 程序。当在页面中遇到 Java applet 时，它将被下载到用户的计算机然后再开始运行。

使用 Java 编写 applet 程序与使用 Java 创建应用程序很不相同。因为每当运行 applet 时，必须先从页面上下载下来，因此为了缩短下载时间，它们比大多数应用程序都小。另外，applet 是在使用它的用户计算机上运行，为阻止恶意或破坏性代码运行，对 applet 会有众多安全限制。

17.1 标准 applet 方法

创建 applet 的第一步是将其声明为 JApplet 的子类，后者位于 Swing 包 javax.swing 中。applet 被视为 Web 页面内的一个可见的窗口，因此 JApplet 与按钮、滚动条及其他用户界面组件一样，也是 Swing 的一部分。

你编写的 applet 在作为 Web 页面的一部分运行时，可以继续所需的所有属性和行为。在开始编写 applet 程序的语句之前，它们就能与 Web 浏览器交互、自我加载和卸载，并根据浏览器窗口的变化刷新其窗口以及处理其他必要的任务。

应用程序运行时，首先要执行 `main()` 块语句中的第一条语句，而 Java applet 没有 `main()` 方法，因此无法设置程序的起点。相反，applet 有一组标准方法，它们在 applet 运行时为响应特定事件而被调用。

下面是可能导致 applet 方法被调用的事件：

- 程序第一次加载时，applet 的方法 `init()` 和 `start()` 被调用；
- 当需要重新显示 applet 窗口时，applet 的 `paint()` 方法被调用；
- 浏览器停止运行程序时，applet 的 `stop()` 方法被调用；
- 程序停止后重新运行时，applet 的 `start()` 方法被调用；
- 程序运行完后卸载时，applet 的 `destory()` 方法被调用。

下面是 applet 的框架：

```
public class Skeleton extends javax.swing.JApplet {
    // program will go here
}
```

applet 类文件必须是 `public` 的，因为 `JApplet` 类也是 `public` 的（如果 applet 使用自定义类，则可以不声明为 `public` 的）。

Applet 类继承所有必要时自动调用的方法：`init()`、`paint()`、`start()`、`stop()`、`destroy()`。然而，这些方法什么也不做。要在 applet 中执行操作，必须在 applet 程序中覆盖这些方法。

17.1.1 在 applet 窗口中绘画

`paint()` 方法用于在 applet 窗口中显示文本和图形。需要在 applet 窗口中显示或重新显示内容时，将调用 `paint()` 方法。也可以使用下面的语句在 applet 中强行调用 `paint()`：

```
repaint();
```

当浏览器或运行浏览器的操作系统有什么变化时，将调用 `paint()` 方法。例如，如果用户关闭了位于浏览器前端的一个窗口，该窗口属于另外一个应用程序，将在 applet 中调用 `paint()` 来重新显示当前程序的所有内容。

`paint()` 方法接受一个参数，该参数为来自 `java.awt` 包的 `Graphics` 对象。

```
public void paint(Graphics screen) {
    Graphics2D screen2D = (Graphics2D) screen;
    // display statements go here
}
```

`Graphics` 类表示可在其中显示内容的环境。和 Swing 程序中一样，可以将其转换为 `Graphics2D` 对象（属于 `java.awt` 包），以使用 Swing 的图形功能。

本章后面将介绍 `drawString()` 方法，它是 `Graphics2D` 类的方法，用于显示文本。

要在 applet 中使用 `Graphics` 或 `Graphics2D` 对象，必须在源代码文件的开头（`class` 语句前）添加下面的 `import` 语句：

```
import java.awt.Graphics;  
import java.awt.Graphics2D;
```

通过使用通配符*, 可以使用这个包中的所有类。

```
import java.awt.*;
```

17.1.2 初始化 applet

当 applet 运行时, 将调用 `init()` 方法一次, 而且只调用一次。所以, 这里是对 applet 成功运行所需的对象和变量进行设置的理想位置。该方法也是设置字体、颜色和屏幕背景色的理想位置。下面是一个例子:

```
public void init() {  
    FlowLayout flo = new FlowLayout();  
    setLayout(flo);  
    JButton run = new JButton("Run");  
    add(run);  
}
```

如果想在其他方法中使用变量, 不应在 `init()` 方法中创建, 因为它只在该方法内存在。通过将变量创建为实例变量的方式, 可以在整个类中使用该变量。

17.1.3 启动和停止 applet

每当开始运行 applet 程序时, 都将调用 `start()` 方法。当程序首次运行时, `init()` 方法后面紧跟着 `start()` 方法。之后, 只有当 applet 在某处停止执行, 并需要启动时, 才会再次调用 `start()` 方法。

applet 停止执行时, 将调用 `stop()` 方法。当用户从包含 applet 的 Web 页面离开并切换到其他页面时将发生该事件。直接在程序中调用 `stop()` 方法时, 也会发生该事件。

17.1.4 销毁 applet

`destroy()` 方法与 `init()` 方法相反, 仅当 applet 被完全关闭并结束运行前被调用。

17.2 将 applet 放到 Web 页面中

在 Web 页面中放置 applet 时, 需要使用 HTML, 这是一种用来创建 Web 页面的标记语言。HTML 可以将格式化文本、图像、声音和其他元素组合起来, 并显示在 Web 浏览器中。HTML 使用称为标记 (tag) 的命令, 并将标记放在 “<” 和 “>” 之间, 其中 `` 用于显示图像, `<p>` 用于插入段落标记, `<h1>` 和 `</h1>` 用来表示位于它们之间的文本是标题 (heading)。

HTML 标记的性能受决定其功能的属性影响。例如, `src` 是 `img` 标记的一个属性, 它提供要显示的图形的文件名。下面是一个 HTML 标记的例子:

```

```


上述标记可以让 Web 页面显示 graduation.jpg 文件。在 Web 页面上放置 applet 的一个方法是使用 applet 标记和几个属性。下面的 HTML 标记将在一个页面上运行 applet:

```
<applet code="StripYahtzee.class" codebase="javadir" height="300"
width="400">
<p>Sorry, no dice ... this requires a Java-enabled browser.</p>
</applet>
```

code 属性指定 applet 类的文件名,如果 applet 使用了多个类文件,code 应指定作为 JApplet 子类的那个类。

codebase 属性包含 applet 和相关文件所在的文件夹或子文件夹的路径。如果没有 codebase 属性,与 applet 相关的所有文件必须与包含 applet 的 Web 页面位于同一个文件夹中。在上面的例子中,codebase 指出可在 javadir 子文件夹中找到 applet StripYahtzee。

属性 height 和 width 指定 applet 窗口在 Web 页面中的大小,其单位为像素。它必须足够大,能够处理要在 applet 中显示的内容。

在开始标记<applet>和结束标记</applet>之间,可以提供一些 HTML 标记,这样当用户的浏览器不支持 Java 或其 Java 功能已经关闭时,可以看到这些信息。

前面的例子中,当浏览器不支持 Java 时,原本显示 applet 的位置将显示文本“Sorry, no dice...this requires a Java-enabled browser”。可以在这里放置有关如何从 Oracle 下载支持 Java 的浏览器的说明,也可以包含超链接和其他 HTML 元素。

另一个很有用的属性是 align,它指定 applet 同页面中其他元素(包括文本和图像)的相对关系。其中 align="left"将 applet 放在相邻页面元素的左边,align="right"将 applet 放在相邻页面元素的右边。

17.3 创建 applet

本章创建的第一个 applet 程序用来显示字符串“Saluton mondo!”,这是传统的世界语问候语。这里将重新创建第 2 章的应用程序 Saluton,将其作为运行在 Web 页面中的程序,以了解 applet 的组织结构。

创建一个名为 SalutonApplet 的 Java 空文件,然后输入程序清单 17.1 中的全部文本,并保存。

程序清单 17.1 SalutonApplet.java 的完整源代码

```
1: import java.awt.*;
2:
3: public class SalutonApplet extends javax.swing.JApplet {
4:     String greeting;
5:
6:     public void init() {
7:         greeting = "Saluton mondo!";
8:     }
9:
10:    public void paint(Graphics screen) {
11:        Graphics2D screen2D = (Graphics2D) screen;
12:        screen2D.drawString(greeting, 25, 50);
```

```
13:     }
14: }
```

这个 SalutonApplet 程序将字符串 “Saluton mondo!” 存放在代码第 6~8 行的 `init()` 方法中，然后在代码第 12 行的 applet 窗口中显示出来。该 applet 不需要使用方法 `start()`、`stop()` 和 `destroy()`，因此没有包含它们。在获悉它是如何编码之后，运行该 applet。

17.3.1 在 applet 窗口中绘画

使用 `Graphics2D` 类的 `drawString()` 方法在 applet 窗口中显示文本，该方法是在图形用户界面组件中绘制文本。

`drawString()` 方法与用来在应用程序中显示文本的 `System.out.println()` 方法相似。

向 `drawString()` 传递下面 3 个参数：

- 要显示的文本，这可以是使用运算符 “+” 拼接起来的多个字符串和变量；
- 要显示字符串的 x 位置（在 (x, y) 坐标系中）；
- 要显示字符串的 y 位置。

applet 的多个方法都使用了 (x, y) 坐标系，它以 applet 窗口的左上角为原点。当向右移动时，x 值增大；当向下移动时，y 值增大。

17.3.2 测试 SalutonApplet 程序

由于 Java applet 没有 `main()` 方法，因此不能向应用程序那样运行。

要想运行 applet，必须在包含 applet 的 Web 页面中添加标记。为了创建 SalutonApplet 的 Web 页面示例，请按照下面的步骤在 NetBeans 中创建一个新的 Web 页面。

1. 选择 `File->New File`，打开 The New File 对话框。
2. 从 Categories 面板中选择 Other，从 File Type 中选择 HTML File，然后单击 Next 按钮，打开 The New HTML File 对话框。
3. 将该文件命名为 SalutonApplet，然后单击 Finish 按钮。

NetBeans 打开源代码编辑器，而且其中带有一些默认的 HTML 标记。删除所有的这些标记，然后输入程序清单 17.2 中的所有文本，并保存。

程序清单 17.2 SalutonApplet.html 的完整源代码

```
1: <html>
2: <head>
3: <title>Saluton Mondo!</title>
4: </head>
5: <body bgcolor="#000000" text="#FF00FF">
6: <p>This is a Java applet.</p>
7: <applet
8:     code="SalutonApplet.class"
9:     codebase="..\..\..\build\classes"
10:     height="150"
```

```

11:     width="300"
12: >
13: <p>You need a Java-enabled browser to see this.</p>
14: </applet>
15: </body>
16: </html>

```

<applet>标记在第 7~14 行定义，但是第 13 行将会被任何支持 Java 的浏览器所忽略。

在保存文件之后，可以在 Web 浏览器中进行查看：在源代码编辑器左边的 Project 面板中，右键单击 SalutonApplet.html 文件名，然后选择 View。该 Web 页面将在计算机默认的浏览器中打开，如图 17.1 所示。



图 17.1

SalutonApplet
applet 载入到
Microsoft IE 中

在浏览器中运行该 applet 时，需要确定浏览器是否可以运行该程序。原因是很多 Web 浏览器在运行 Java applet 之前，必须先行配置，然后才能支持 Java 程序的运行。

Java applet 现在是借助 Java 插件在当前的浏览器中运行的，Java 插件是 Oracle 开发的一个解释器，它可以支持 Java 语言的最新版本。

插件是一个与 Web 浏览器协同工作的程序，它扩展了浏览器的功能。插件可以处理浏览器无法处理的数据类型。Apple 提供了用来显示 QuickTime 电影的插件，Macromedia 提供了运行 Flash 动画文件的插件，还有众多其他的特殊内容也是通过这种方式支持的。

提示：

NetBeans 可以在不需要创建 Web 页面的情况下来测试 applet。当 applet 的类文件在源代码编辑器中打开时，选择菜单命令 Run->Run File 后，applet 就被 appletviewer (JDK 的一部分) 工具载入进去。

**Did you
Know?**

17.4 从 Web 页面传递参数

使用存储在 HTML 标记中的参数，可以自定义 applet 的功能，而且该参数的目的与应用程序中作为命令行输入的参数相同。

参数存储在包含 applet 的 Web 页面中，是通过 HTML 标记 param 及其两个属性 name 和

value 创建的。在 applet 中可以有多多个 param 标记,但它们都必须位于标记<applet>和</applet>之间。下面是一个包含多个参数的 applet 标记:

```
<applet code="ScrollingHeadline" height="50" width="400">
  <param name="headline1" value="Dewey defeats Truman">
  <param name="headline2" value="Stix nix hix pix">
  <param name="headline3" value="Man bites dog">
</applet>
```

上述代码可用于向 applet 发送新闻提要,该 applet 在屏幕上滚动显示这些新闻提要。由于新闻在不断变化,创建这种程序的唯一办法是使用参数。

name 属性用于给每个参数命名, value 属性用于指定参数的值。

17.5 在 applet 中接收参数

通过调用 applet 的 getParameter(String)方法(继承自 JApplet),可以访问 applet 中的参数,其中 name 属性指定的参数名作为上述方法中的参数,如下面的语句所示:

```
String display1 = getParameter("headline1");
```

getParameter()方法将参数作为字符串返回,因此必要时必须将其转换为其他类型。如果要将返回的参数作为整数使用,可使用下面的语句:

```
int speed;
String speedParam = getParameter("speed");
if (speedParam != null) {
    speed = Integer.parseInt(speedParam);
}
```

这里使用 speedParam 字符串来设置整型变量 speed 的值。使用 getParameter()方法检索参数,而该参数没有在网页中使用 param 标记指定时,getParameter()方法将返回 null,即空字符串。

**Watch
Out!**

警告:

Integer.parseInt(String)方法需要使用异常,该技术将在第 18 章讲解。

17.6 在 applet 中处理参数

接下来要编写的 applet 程序将接受一个人的体重,并以不同的单位进行显示。该 applet 将接受两个参数:以磅为单位的体重,以及该体重对应的人名。体重用于计算用盎司、千克和吨为单位表示的体重,并将它们显示出来。

创建一个名为 WeightScale 的新 Java 空文件,输入程序清单 17.3 中的所有文本,并保存。

程序清单 17.3 WeightScale.java 的完整源代码

```
1: import java.awt.*;
2:
3: public class WeightScale extends javax.swing.JApplet {
```

```

4:    float lbs = 0F;
5:    float ozs;
6:    float kgs;
7:    float metricTons;
8:    String name = "somebody";
9:
10:   public void init() {
11:       String lbsValue = getParameter("weight");
12:       if (lbsValue != null) {
13:           lbs = Float.valueOf(lbsValue);
14:       }
15:       String personValue = getParameter("person");
16:       if (personValue != null) {
17:           name = personValue;
18:       }
19:       ozs = (float) (lbs * 16);
20:       kgs = (float) (lbs / 2.204623);
21:       metricTons = (float) (lbs / 2204.623);
22:   }
23:
24:   public void paint(Graphics screen) {
25:       Graphics2D screen2D = (Graphics2D) screen;
26:       screen2D.drawString("Studying the weight of " + name, 5, 30);
27:       screen2D.drawString("In pounds: " + lbs, 55, 50);
28:       screen2D.drawString("In ounces: " + ozs, 55, 70);
29:       screen2D.drawString("In kilograms: " + kgs, 55, 90);
30:       screen2D.drawString("In metric tons: " + metricTons, 55, 110);
31:   }
32: }

```

init()方法将两个参数载入到 applet 中。由于参数是来自 Web 页面的字符串，因此 weight 参数必须转为浮点型参数，以便于在数学表达式中使用。Float 对象类包含 valueOf(String)方法，它可以将字符串的值作为 Float 返回。在第 13 行代码中，该值将自动拆包 (unbox) 为 float 变量。

第 19~22 行将 lbs 变量转换为不同度量单位的值。每一个语句在转换语句前面都有一个 (float)，用于将等式的结果转换为浮点型数值。

applet 的 paint()方法使用 drawString()来显示一行文本。paint()方法有 3 个参数：要显示的文本、文本的 x 位置和 y 位置。

在测试 WeightScale applet 之前，需要创建一个包含该 applet 的 Web 页面。在 NetBeans 中创建一个新 HTML 文件，将其命名为 WeightScale。在该文件中输入程序清单 17.4 中的所有文本，然后在浏览器中打开这个新创建的 Web 页面，其打开方式为在 Project 面板中右键单击 WeightScale.html 文件名，然后选择 View。

程序清单 17.4 WeightScale.html 的完整源代码

```

1: <applet code="WeightScale.class" codebase="..\..\..\build\classes"
2:     height="170" width="210">
3:     <param name="person" value="Konishiki">
4:     <param name="weight" value="605">
5: </applet>

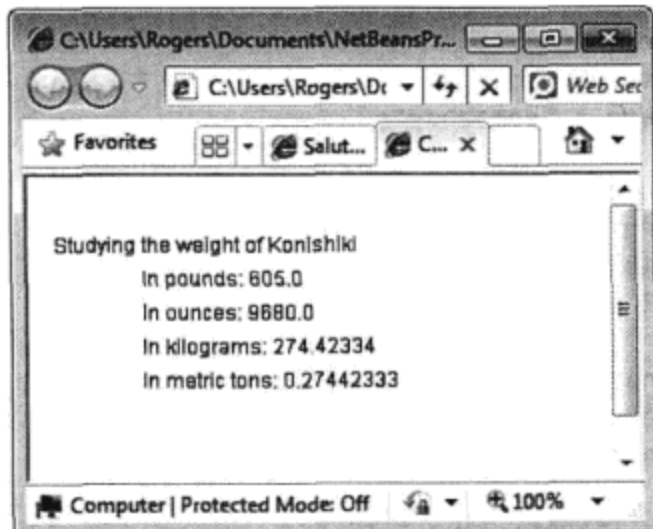
```

这里使用的是 Konishiki 的体重，这位美国出生的相扑冠军体重超过 605 磅，是这些穿比

基尼的庞然大物中最重的。读者可以将其替换任何人的体重。图 17.2 所示为该 applet 的输出。

图 17.2

WeightScale
applet 装载到
Internet Explorer 中



要让该 applet 显示不同的人名和体重，只需修改文件 WeightScale.html，applet 仍将正确运行。

17.7 使用 object 标记

在 HTML5 中载入 Java applet、Flash 程序以及其他格式的交互式内容时，使用的不再是 `<applet>` 标记，而是 `<object>` 标记。与 `<applet>` 相同，该标记也有 `height` 和 `width` 属性，不过它还有另外一个类型属性——`application/x-java-applet`，这是 MIME 为 Java applet 设定的类型。（MIME 类型对可以经由 Internet 传输的文件格式进行了分类）。下面是一个对象的格式示例：

```
<object type="application/x-java-applet" height="300" width="400">
</object>
```

applet 的 `code` 和 `codebase` 没有作为参数指定，相反，这两个参数被放置在 `<object>` 和 `</object>` 标记之间。

下面的 HTML5 标记可以显示一个 applet：

```
<object type="application/x-java-applet" height="300" width="400">
  <param name="code" value="StripYahtzee" />
  <param name="codebase" value="javadir" />
  <p>Sorry, no dice ... this requires a Java-enabled browser.</p>
</object>
```

17.8 总结

本书的大部分章介绍的都是应用程序，主要是因为当今的大多数 Java 程序员不设计大量用于 Web 的 applet。

通过一组默认的安全限制可以令 Web 页面中的 applet 安全地在用户计算机上运行。它们不能将文件存放到用户计算机上，也不能读取计算机中的文件、列出文件夹中的文件，以及创建被当做 Java applet 的弹出窗口。

通过为在 applet 中使用数字签名，并让用户批准该 applet 的方法，可以克服这些限制。将 Java 程序部署为 applet 的另外一个方法是使用 Java Web Start，这是一种可以从 Web 浏览器启用 Java 程序的技术。

17.9 问与答

问：为何要在 applet 标记中使用 codebase 属性？

答：如果所有的 Java 程序都存放在 codebase 指定的子文件夹中，可能改善网站的组织结构，但没有其他理由表明使用 codebase 优于省略它。如何做因人而异。

问：为什么 applet 没有 main() 方法？

答：applet 不适用 main() 的原因是，它的生命周期要比应用程序复杂。应用程序的生命周期是：开始运行，处理其工作，处理完毕之后退出。而当包含 applet 的 Web 页面在显示时，applet 可能需要运行和停用多次。

如果用户使用回退按钮离开当前页面，然后再使用前进按钮返回到该页面，则 applet 的 start() 方法将再次调用。如果隐藏 applet 的弹出窗口被关闭，则会调用 applet 的 paint() 方法。

JApplet 类使得这些复杂的交互可以在浏览器中运行。

17.10 测验

通过下列问题测试对 applet 的理解程度。

17.10.1 问题

1. paint() 方法接受什么类型的参数？
 - a. 一个 Graphics 对象。
 - b. 一个 Graphics2D 对象。
 - c. 不接受任何参数。
2. 在 applet 结束运行前将调用下面哪个方法？
 - a. decline()。
 - b. destroy()。
 - c. defenestrate()。
3. 为什么不能在 init() 方法中创建 applet 需要的所有变量？
 - a. 这样变量的作用域将为该方法。
 - b. 法律禁止这样做。
 - c. 可以在这里创建，而不会有任何问题。

17.10.2 答案

1. a. Graphics 对象跟踪在 applet 窗口中显示内容所需的属性和行为，可以在该方法中创建 Graphics2D 对象，但它将不是作为参数传入的。

2. b. `destroy()` 方法可以释放 applet 占用的资源。
3. a. 对于在类的多个方法中使用的变量，应在类语句后和任何方法前创建。

17.11 练习

完成下面的练习，以应用 applet 编程知识：

- 编写一个 applet，每当 applet 窗口被重绘时都移动显示的文本。
- 在浏览器中安装 Java 插件，然后尝试运行 www.javaonthebrain.com 上的 applet。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 18 章

处理程序中的错误

本章介绍如下内容：

- 如何响应 Java 程序中的异常；
- 如何创建忽略异常，让其他类去处理的方法；
- 如何使用引起异常的方法；
- 如何创建异常。

错误、bug、严重错误、输入错误及其他导致程序无法正确运行的问题是软件开发过程的正常组成部分。“正常”可能是用于描述错误的最善意的用词。我在编程时，当不能找出导致程序不能正确运行的难懂错误时，我会用让强盗都脸红的词。

有些错误将被编译器捕获，进而阻止你创建类；导致程序无法成功运行的其他错误将被解释器发现。Java 将错误划分为如下两类。

- 异常 (Exception)：表明程序运行时发生了异常情况的事件情况。
- 错误 (Errors)：表明解释器遇到问题，但可能与程序无关。

Java 程序不能从错误中恢复过来继续运行，所以它不是本章关注的重点。在进行 Java 编程时，读者可能遇到过 `OutOfMemoryError` 的错误，程序对这种类型的错误无能为力，只能退出运行。

而异常可以采用某种方式来处理，而且程序也会继续运行。

18.1 异常

虽然刚开始学习它，但通过本书前 17 章，读者可能已经对异常非常熟悉了。编写的 Java 程序编译成功但运行时遇到问题时，出现的错误就是异常。

例如，一种常见的编程错误是引用不存在的数组元素，如下面的语句所示：

```
String[] greek = { "Alpha", "Beta", "Gamma" };
System.out.println(greek[3]);
```

在这个例子中，String 数组 `greek` 有 3 个元素。数组的第一个元素编号为 0 而不是 1，因此第一个元素是 `greek[0]`，第二个元素是 `greek[1]`，第三个元素是 `greek[2]`，所以试图显示 `greek[3]` 的语句是错误的，因为该元素不存在。上述语句可以成功编译，但运行程序时，Java 解释器将显示下面的消息并停止运行：

Output ▼

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
    at SampleProgram.main(SampleProgram.java:4)
```

这条消息表明应用程序引发了异常，解释器通过显示错误消息并停止运行程序来指出这一点。

这条错误消息引用了 `java.lang` 包中的 `ArrayIndexOutOfBoundsException` 类，这是一个异常，该对象用于指出 Java 程序中发生了异常情况。

Java 类遇到异常后，它向类的用户指出错误类型。在这个例子中，类的用户是 Java 解释器。

By the Way

注意：

有两个术语可用于描述该过程：引发和捕获。对象抛出异常，以指出发生了异常。这些异常可被其他对象或 Java 解释器捕获。

所有异常都是 `Exception` 的子类，`Exception` 位于 `java.lang` 包中。正如读者预期的，`ArrayIndexOutOfBoundsException` 类指出使用了数组边界外的元素。

Java 中有数百种异常，其中很多表明问题可以通过修改程序得到解决，如数组异常，这些异常类似于编译错误，纠正后就不用担心它会再出现。

其他异常必须使用 5 个新关键词在程序运行时进行处理：`try`、`catch`、`finally`、`throw` 和 `throws`。

18.1.1 在 try-catch 块中捕获异常

到目前为止，读者通过纠正导致异常的问题来处理异常。有时这样无法解决异常，因此必须在 Java 类中处理异常。

为说明为何这很有用，在新的 Java 空文件中输入程序清单 18.1 中的 Java 应用程序，将其命名为 `Calculator`，然后保存。

程序清单 18.1 `Calculator.java` 的源代码

```
1: public class Calculator {
2:     public static void main(String[] arguments) {
3:         float sum = 0;
4:         for (int i = 0; i < arguments.length; i++) {
5:             sum = sum + Float.parseFloat(arguments[i]);
```

```

6:      }
7:      System.out.println("Those numbers add up to " + sum);
8:  }
9: }

```

该应用程序通过命令行参数接受一个或多个数字，然后将它们相加并显示结果。

在 Java 应用程序中，所有命令行参数都用字符串表示，将它们相加前，程序必须将其转换为浮点数。第 5 行的 `Float.parseFloat()` 方法完成这项任务，并将转换得到的数加到变量 `sum` 中。

在运行该应用程序之前，先对命令行参数进行设置：在 NetBeans 中选择 `Run->Set Project Configuration->Customize` 命令，然后输入 `8 6 7 5 3 0 9`。然后选择 `Run->Run Main Project` 来运行该程序，其输出如图 18.1 所示。

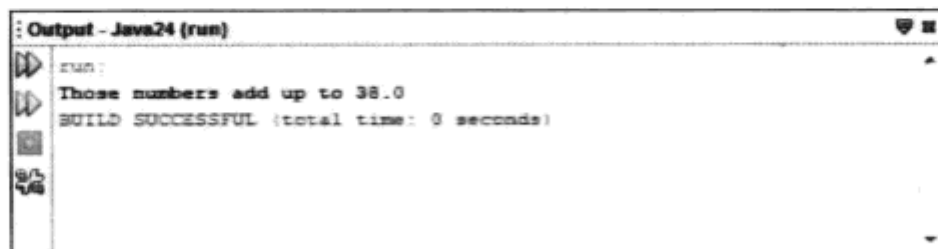


图 18.1

Calculator 应用程序的输出

使用不同的数字作为参数运行该应用程序多次，程序都将成功处理，这让读者产生疑惑，这与异常有什么关系？

要看到异常，将 Calculator 应用程序的命令行参数修改为 `1 3 5x`。

这里的第三个参数包含输入错误：数字 5 后不应是字符 `x`。Calculator 应用程序不知道这是错误，所以将 `5x` 同其他数字相加，导致下面的异常：

Output ▼

```

Exception in thread "main" java.lang.NumberFormatException: For input
string: "5x" at sun.misc.FloatingDecimal.readJavaFormatString
(FloatingDecimal.java:1224)
    at java.lang.Float.parseFloat(Float.java:422)
    at Calculator.main(Calculator.java:5)

```

尽管该消息对程序员来说很有用，但他们不希望用户看到。Java 程序可以使用 `try-catch` 块语句来处理异常，其格式如下：

```

try {
    // statements that might cause the exception
} catch (Exception e) {
    // what to do when the exception occurs
}

```

对于希望类方法来处理的任何异常，都必须使用一个 `try-catch` 块。在 `catch` 语句中的 `Exception` 对象应是下面 3 个中的一个：

- 可能发生的异常类；
- 多个异常类，期间使用 `|` 字符隔开；
- 可能发生的多种异常的超类。

try-catch 块的 try 部分应包含可能引发异常的语句。在应用程序 Calculator 中，第 5 行调用了 Float.parseFloat(String)方法，当该方法中的参数是不能转换为浮点数的字符串时，将抛出 NumberFormatException。

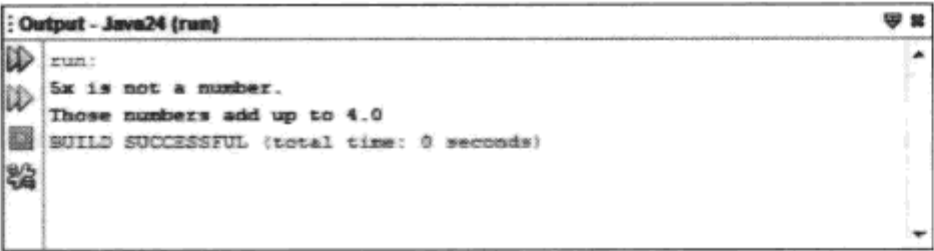
为了改善应用程序 Calculators，使其遇到这种错误时不停止运行，可使用一个 trycatch 块。创建一个名为 NewCalculator 的 Java 空文件，然后输入程序清单 18.2 中的所有文本。

程序清单 18.2 NewCalculator.java 程序

```
1: public class NewCalculator {
2:     public static void main(String[] arguments) {
3:         float sum = 0;
4:         for (int i = 0; i < arguments.length; i++) {
5:             try {
6:                 sum = sum + Float.parseFloat(arguments[i]);
7:             } catch (NumberFormatException e) {
8:                 System.out.println(arguments[i] + " is not a
                ↪number.");
9:             }
10:        }
11:        System.out.println("Those numbers add up to " + sum);
12:    }
13: }
```

保存该应用程序之后，使用命令行参数 1 3 5x 运行该程序，将会看到如图 18.2 所示的输出。

图 18.2
NewCalculator
应用程序的输出



第 5~9 行的 try-catch 块处理 Float.parseFloat()方法抛出的 NumberFormatException 错误。这些异常是在 NewCalculator 类中捕获的，如果某个参数不是数字，该类将显示一条错误消息。由于在这个类中处理了异常，因此 Java 解释器不会显示错误消息。可以使用 try-catch 块来处理与用户输入和其他意外数据相关的问题。

18.1.2 捕获多种不同的异常

try-catch 块可用来处理几种不同类型的异常，即使这些异常是由不同的语句抛出的。处理多种异常类的一种方法是在每一个异常中使用一个 catch 语句块，如下面的代码所示：

```
String textValue = "35";
int value;
try {
    value = Integer.parseInt(textValue);
} catch (NumberFormatException exc) {
    // code to handle exception
}
```

```

} catch (Arithmetic Exception exc) {
    // code to handle exception
}

```

在 Java 7 中，可以在同一个 `catch` 块中处理多个异常，方法如下：多个异常之间使用管道字符 (`|`) 隔开，并在最后使用异常变量来结尾。其示例如下：

```

try {
    value = Integer.parseInt(textValue);
} catch (NumberFormatException | Arithmetic Exception exc) {
    // code to handle exceptions
}

```

如果该代码捕获了 `NumberFormatException` 或 `ArithmeticException` 异常，则将其赋值给 `exc` 变量。

程序清单 18.3 包含了名为 `NumberDivider` 的应用程序，该应用程序从命令行接受两个整型参数，然后将它们用于除法表达式中。

该应用程序必须能够处理两个潜在的用户输入问题：

- 非数字型参数；
- 除数为 0。

创建一个名为 `NumberDivider` 的新 Java 空文件，然后输入程序清单 18.3 中的所有文本。

程序清单 18.3 `NumberDivider.java` 的源代码

```

1: public class NumberDivider {
2:     public static void main(String[] arguments) {
3:         if (arguments.length == 2) {
4:             int result = 0;
5:             try {
6:                 result = Integer.parseInt(arguments[0]) /
7:                     Integer.parseInt(arguments[1]);
8:                 System.out.println(arguments[0] + " divided by " +
9:                     arguments[1] + " equals " + result);
10:            } catch (NumberFormatException e) {
11:                System.out.println("Both arguments must be numbers.");
12:            } catch (ArithmeticException e) {
13:                System.out.println("You cannot divide by zero.");
14:            }
15:        }
16:    }
17: }

```

使用命令行参数来指定该应用程序的两个参数。在运行该程序时，你可以将其参数指定为整数、浮点数和非数字型参数。

第 3 行的 `if` 语句检查是否给应用程序传递了两个参数，如果不是，程序将退出且不显示任何信息。

应用程序 `NumberDivider` 执行整数除法运算，结果也是整数。在整数除法中，5 除以 2 等于 2，而不是 2.5。

如果使用浮点数或非数字作为参数，第 6~7 行将抛出 `NumberFormatException` 异常，并

被第 10~11 行捕获。

如果使用整数作为第一个参数，使用 0 作为第二个参数，第 6~7 行将抛出 `ArithmeticException` 异常，并被第 12~13 行捕获。

18.1.3 出现异常后进行处理

使用 `try-catch` 块处理多种异常时，有时希望不管是否发生异常，程序都在块的后面执行某种操作。

为此，可以使用 `try-catch-finally` 块，其格式如下：

```
try {
    // statements that might cause the exception
} catch (Exception e) {
    // what to do when the exception occurs
} finally {
    // statements to execute no matter what
}
```

其中，`finally` 部分的语句将在其他语句执行后执行，而不管是否发生异常。

其用途之一是在从磁盘文件中读取数据的程序中，这将在第 20 章介绍。访问数据时可能发生多种异常：文件不存在、磁盘错误等。如果读取磁盘的语句在 `try` 部分，并在 `catch` 部分处理错误，可以在 `finally` 部分关闭文件。这可以确保无论读取文件是否发生异常，都将关闭文件。

18.1.4 抛出异常

调用另一个类的方法时，那个类可以通过抛出异常来控制如何使用该方法。

使用 Java 类库中的类时，编译器经常会显示下面这样的消息：

Output ▼

```
NetReader.java:14: unreported exception java.net.MalformedURLException;
must be caught or declared to be thrown
```

看到包含 “`must be caught or declared to be thrown`” 这样的错误消息时，表明你试图使用的方法抛出了异常。

调用这些方法的任何类，例如你编写的应用程序，必须做下面的工作之一：

- 使用 `try-catch` 块处理异常；
- 抛出异常；
- 先用 `try-catch` 块处理异常，然后再抛出异常。

至此，读者看到了如何处理异常。如果想先处理异常再抛出它，可使用关键字 `throw` 和要抛出的异常对象。

下面的语句在 `catch` 块中处理错误 `NumberFormatException`，然后再抛出它：


```
try {
    principal = Float.parseFloat(loanText) * 1.1F;
} catch (NumberFormatException e) {
    System.out.println(arguments[i] + " is not a number.");
    throw e;
}
```

下面这种改写的代码可以处理 try 块语句中生成的所有异常，并在处理完后再抛出它：

```
try {
    principal = Float.parseFloat(loanText) * 1.1F;
} catch (Exception e) {
    System.out.println("Error " + e.getMessage());
    throw e;
}
```

Exception 是所有异常子类的父类。这个 **catch** 语句可以捕获 **Exception** 类，以及类层次结构中位于 **Exception** 类下的所有子类。

当使用 **throw** 抛出一个异常时，通常意味着没有完成处理异常需要完成的所有工作。

一个采用这种方式很有用的情形是：一个 **CreditCardChecker** 应用程序验证信用卡支付，它使用了一个名为 **CheckDatabase** 的类，该类完成如下工作：

- 连接到信用卡贷方的计算机；
- 询问该计算机，消费者的信用卡号是否有效；
- 询问该计算机，消费者是否有足够的信用。

当 **CheckDatabase** 类执行其工作时，如果信用卡贷方的计算机根本不应答电话请求将如何呢？这种错误正是 try-catch 块要解决的，在 **CheckDatabase** 类使用它来处理连接错误。

如果 **CheckDatabase** 类自己处理这种错误，**CreditCardChecker** 应用程序将根本不知道发生了异常。这不是好主意——应用程序应知道不能建立连接，以便向使用应用程序的用户报告。

通知应用程序 **CreditCardChecker** 的一种方法是，在 **CheckDatabase** 类中使用 **catch** 块捕获异常，然后使用 **throw** 语句抛出它。异常将在 **CheckDatabase** 中抛出，**CheckDatabase** 必须像处理其他异常一样处理它。

出现错误或其他不正常情况时，异常处理是不同类相互进行通信的一种方式。

当在捕获父类异常（比如 **Exception**）的 **catch** 块中使用 **throw** 时，将会抛出该父类的异常。这样将无法获悉所发生错误的详情，因为子类异常（比如 **NumberFormatException**）提供的信息要比 **Exception** 类更为详细。

Java 7 提供了新的方法来记录异常的详情：**catch** 语句中的 **final** 关键字。

```
try {
    principal = Float.parseFloat(loanText) * 1.1F;
} catch (final Exception e) {
    System.out.println("Error " + e.getMessage());
    throw e;
}
```

`catch` 语句中的 `final` 关键字会导致 `throw` 语句在执行时，表现出不同的行为，即抛出所捕获的特定异常类。

18.1.5 忽略异常

本章将介绍的最后一种技术是如何完全忽略异常。在类中，可以在方法的定义中使用关键字 `throw`，让方法忽略异常。

下面的方法抛出 `MalformedURLException`，这是在 Java 程序中使用网络地址时可能发生的一种错误：

```
public loadURL(String address) throws MalformedURLException {  
    URL page = new URL(address);  
    loadWebPage(page);  
}
```

其中的第二条语句 `URL page = new URL(address);` 创建一个 `URL` 对象，该对象表示一个网络地址。`URL` 类的构造函数抛出 `MalformedURLException`，指出使用的地址无效，因此无法创建这样的对象。下面的语句将抛出这样的异常：

```
URL source = new URL("http:www.java24hours.com");
```

字符串 `http:www.java24hours.com` 不是有效的 `URL`，因为冒号后少一些符号：两个反斜线 (`//`)。

由于 `loadURL()` 方法被声明为抛出 `MalformedURLException` 错误，因此不用在方法中处理它们。这种异常将由调用 `loadURL()` 方法的方法负责处理。

18.2 抛出和捕获异常

接下来创建一个类，该类使用异常将发生的错误告诉另一个类。

该类为 `HomePage`，代表网上的个人网站。`PageCatalog` 是一个对个人网站进行分类的应用程序。

创建一个新的 Java 空文件，将其命名为 `HomePage`，然后输入程序清单 18.4 中的完整文本。

程序清单 18.4 `HomePage.java` 的完整源代码

```
1: import java.net.*;  
2:  
3: public class HomePage {  
4:     String owner;  
5:     URL address;  
6:     String category = "none";  
7:  
8:     public HomePage(String inOwner, String inAddress)  
9:         throws MalformedURLException {  
10:  
11:         owner = inOwner;
```

```

12:         address = new URL(inAddress);
13:     }
14:
15:     public HomePage(String inOwner, String inAddress, String inCategory)
16:         throws MalformedURLException {
17:
18:         this(inOwner, inAddress);
19:         category = inCategory;
20:     }
21: }

```

你可以在其他程序中使用这个编译后的 `HomePage` 类。这个类代表网上的个人网站，它包含 3 个实例变量：`address`（代表网站地址的 `URL` 对象）、`owner`（代表网站的主人）和 `category`（描述网站主要内容的注释）。

与创建 `RUL` 对象的其他类一样，`HonePage` 也必须在 `try-catch` 块中处理 `MalformedURLException` 错误或声明忽略这些错误。

这个类采用后一种方法，如第 8~9 行和第 15~16 行所示。通过在两个构造函数中使用 `throw` 语句，`HomePage` 避免了处理 `MalformedURLException` 错误。

为了创建一个使用 `HomePage` 类的应用程序，回到 `NetBeans` 并创建一个名为 `PageCatalog` 的 Java 空文件，然后输入程序清单 18.5 中的文本。

程序清单 18.5 PageCatalog.java 的完整源代码

```

1: import java.net.*;
2:
3: public class PageCatalog {
4:     public static void main(String[] arguments) {
5:         HomePage[] catalog = new HomePage[5];
6:         try {
7:             catalog[0] = new HomePage("Mark Evanier",
8:                 "http://www.newsfromme.com", "comic books");
9:             catalog[1] = new HomePage("Todd Smith",
10:                 "http://www.sharkbitten.com", "music");
11:             catalog[2] = new HomePage("Rogers Cadenhead",
12:                 "http://workbench.cadenhead.org", "programming");
13:             catalog[3] = new HomePage("Juan Cole",
14:                 "http://www.juancole.com", "politics");
15:             catalog[4] = new HomePage("Rafe Colburn",
16:                 "www.rc3.org");
17:             for (int i = 0; i < catalog.length; i++) {
18:                 System.out.println(catalog[i].owner + ": " +
19:                     catalog[i].address + " - " +
20:                     catalog[i].category);
21:             }
22:         } catch (MalformedURLException e) {
23:             System.out.println("Error: " + e.getMessage());
24:         }
25:     }
26: }

```

运行编译后的程序时，将显示如下输出：

Output ▼

Error: no protocol: www.rc3.org

应用程序 PagaCatalog 创建一个 HomePage 对象数组，然后显示该数组的内容。创建每个 HomePage 对象时最多使用 3 个参数：

- 网站主人的姓名；
- 网站地址（String 而不是 URL）；
- 页面分类。

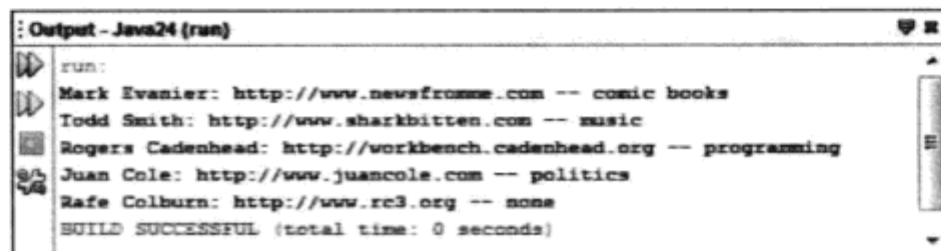
第 3 个参数是可选的，第 15~16 行没有使用。

HomePage 类的构造函数收到不能转换为有效 URL 对象的字符串时，将抛出 MalformedURLException 错误。这些异常在 PagaCatalog 应用程序中使用 try-catch 块来处理。

要修复导致“no protocol”错误的问题，编辑第 16 行，使字符串以 http://打头，就像第 7~14 行的网络地址一样。当再次运行该程序时，其输出如图 18.3 所示。

图 18.3

PageCatalog 应用程序的输出



18.3 总结

使用 Java 异常处理技术后，希望有关错误的主题要更受读者的欢迎。

可以使用这种技术做很多工作：

- 捕获异常并处理它；
- 忽略异常，将其留给另一个类或解释器去处理；
- 在同一个 try-catch 块中捕获多种异常；
- 抛出自己的异常。

通过在 Java 程序中管理异常，可使程序更可靠、更通用、更易于使用，因为别人在使用你的软件时，不会显示晦涩的错误消息。

18.4 问与答

问：能够创建自己的异常吗？

答：可以通过创建现有异常（如 Exception，它是所有异常的超类）的子类来创建自己的异常。在 Exception 的子类中，可能需要覆盖两个方法：不接受任何参数的 Exception() 方法和接受一个 String 参数的 Exception() 方法。在后一个方法中，字符串参数应是一条消息，它描述发生的错误。

问：除异常外，本章为什么没有描述如何抛出和捕获错误？

答：Java 将问题分为错误和异常两类，因为它们的严重程度不同。异常不那么严重，因此应在程序中使用 `try-catch` 或 `throw` 来处理它们；而错误更严重，仅在程序中处理它们是不够的。

有关错误的两个例子是栈溢出和内存耗尽，这些错误可能导致 Java 解释器崩溃，而且在解释器运行时，没有办法在程序中修复这种错误。

18.5 测验

本章充斥“错误”一词，看看能否不出任何错误地回答下列问题。

18.5.1 问题

1. 单条 `catch` 语句能够处理多少种异常？
 - a. 一种。
 - b. 多种异常。
 - c. 我不想回答这个问题。
2. `finally` 部分中的语句在什么情况下运行？
 - a. 出现异常且 `try-catch` 块结束后。
 - b. 不出现异常且 `try-catch` 块结束后。
 - c. 两者都是。

18.5.2 答案

1. b. `catch` 语句中的 `Exception` 对象能够各种异常。
2. c. `finally` 部分中的语句总是在 `try-catch` 块执行完毕后执行，而不管是否发生异常。

18.6 练习

要检测你是否是一名优秀的 Java 程序员，尽量少犯错误地完成下列练习。

- 修改应用程序 `NumberDivider`，使其抛出它能够捕获的任何异常。然后运行程序，看看发生的情况。
- 第 15 章创建的 `LottoEvent` 类有一个 `try-catch` 块，根据这个块创建自己的 `Sleep` 类，它处理 `InterruptedException` 异常，这样其他类（如 `LottoEvent`）就不需要处理这种异常了。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。

第 19 章

创建线程程序

本章介绍如下内容：

- 在程序中使用接口；
- 创建线程；
- 启动、终止和暂停线程；
- 捕获错误。

一个常用于描述每天紧张忙碌的生活节奏的计算机术语是“多任务”，意思是同时做多件事，如一边浏览 Web 站点，一边参加电话会议，同时还做压腿练习。多线程计算机是同时可以运行多个程序的计算机。

Java 语句的一个复杂特性是，能够使用线程编写具有多任务功能的程序，Java 的该功能是通过称为线程的对象类实现的。

19.1 线程

在 Java 程序中，计算机同时处理的每一个任务称为“线程”，所有的任务称为“多线程”。线程在动画和很多其他程序中很有用。

线程是一种组织程序的方式，使其能够同时做多项工作。需要同时执行的每项任务都运行在自己的线程中，这通常是通过将每项任务作为独立的类来实现的。

线程用 Thread 类和 Runnable 接口表示，它们都位于 java.lang 包中。由于它们都位于这个包中，因此在程序不用使用 import 语句就能使用它们。

Thread 类的一种最简单的用途是降低程序完成工作的速度。

19.1.1 降低程序的速度

Thread 类有一个 sleep() 方法，可在任何程序中调用它让程序停止运行一段时间。在动画

程序中经常使用这种技术，以防止图像显示的速度超过 Java 解释器的处理能力。

要使用 `sleep()` 方法，可调用 `Thread.sleep()` 方法并使用要暂停的毫秒数作参数，如下面的语句所示：

```
Thread.sleep(5000);
```

上述语句让 Java 解释器暂停 5 秒再继续运行。如果由于某种原因解释器不能停止那么长时间，`sleep()` 方法将抛出 `InterruptedException` 异常。

由于可能会抛出上面提到的异常，因此，在使用 `sleep()` 方法时必须采取某种方式处理这种异常。一种解决办法是将 `Thread.sleep()` 语句放在 `try-catch` 块中：

```
try {
    Thread.sleep(5000);
} catch (InterruptedException e) {
    // wake up early
}
```

要让 Java 程序同时处理多项任务，必须将程序组织成线程。根据需要，程序可以有任意多个线程，它们将同时运行，而且不会相互影响。

19.1.2 创建线程

可作为线程运行的 Java 类常被称为“线程化类”。虽然使用线程可让程序暂停几秒再执行，但通常因为相反的原因而使用它们：提高程序的运行速度。如果将耗时的任务放在独立的线程中，程序的其他部分运行起来将更快，这通常用于防止任务降低程序的图形用户界面的响应速度。

例如，如果编写了一个应用程序，它从磁盘加载股价数据并生成统计数据，则最耗时的任务是从磁盘加载数据。如果不在应用程序中使用线程，加载数据时程序的界面响应将很慢。这可能让用户失去耐心。

有两种方法可将任务放在线程中：

- 将任务放在实现了 `Runnable` 接口的类中；
- 将任务放在 `Thread` 的子类中。

要支持 `Runnable` 接口，可在创建类时使用关键字 `implements`，如下例所示：

```
public class LoadStocks implements Runnable {
    // body of the class
}
```

当类实现了接口后，除了自己的方法外，它还将支持其他的行为。

实现了 `Runnable` 接口的类必须包含 `run()` 方法，该方法的结构如下：

```
public void run() {
    // body of the method
}
```

`run()` 方法应完成线程要完成的任务。在股票分析示例中，`run()` 方法应包含从磁盘加载

数据以及根据这些数据生成统计信息的语句。

当线程应用程序运行时，不会自动执行 `run()` 方法中的语句。在 Java 中可以启动和终止线程，要运行线程，必须做下面两项工作：

- 通过调用构造函数 `Thread` 创建线程化类的对象；
- 通过调用 `start()` 方法启动线程。

构造函数 `Thread` 接受一个参数——包含线程的 `run()` 方法的对象。通常使用 `this` 作为参数，`this` 表明当前类包含 `run()` 方法。

程序清单 19.1 所示的 Java 应用程序在文本区域显示一系列素数。创建一个名为 `Prime Finder` 的 Java 空文件，然后输入程序清单 19.1 中的所有文本，并保存。

程序清单 19.1 PrimeFinder.java 程序的源代码

```

1: import java.awt.*;
2: import javax.swing.*;
3: import java.awt.event.*;
4:
5: class PrimeFinder extends JFrame implements Runnable, ActionListener {
6:     Thread go;
7:     JLabel howManyLabel;
8:     JTextField howMany;
9:     JButton display;
10:    JTextArea primes;
11:
12:    PrimeFinder() {
13:        super("Find Prime Numbers");
14:        setLookAndFeel();
15:        setSize(400, 300);
16:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17:        BorderLayout bord = new BorderLayout();
18:        setLayout(bord);
19:
20:        howManyLabel = new JLabel("Quantity: ");
21:        howMany = new JTextField("400", 10);
22:        display = new JButton("Display primes");
23:        primes = new JTextArea(8, 40);
24:
25:        display.addActionListener(this);
26:        JPanel topPanel = new JPanel();
27:        topPanel.add(howManyLabel);
28:        topPanel.add(howMany);
29:        topPanel.add(display);
30:        add(topPanel, BorderLayout.NORTH);
31:
32:        primes.setLineWrap(true);
33:        JScrollPane textPane = new JScrollPane(primes);
34:        add(textPane, BorderLayout.CENTER);
35:
36:        setVisible(true);
37:    }
38:
39:    public void actionPerformed(ActionEvent event) {
40:        display.setEnabled(false);
41:        if (go == null) {
42:            go = new Thread(this);
43:            go.start();

```

```

44:     }
45: }
46:
47: public void run() {
48:     int quantity = Integer.parseInt(howMany.getText());
49:     int numPrimes = 0;
50:     // candidate: the number that might be prime
51:     int candidate = 2;
52:     primes.append("First " + quantity + " primes:");
53:     while (numPrimes < quantity) {
54:         if (isPrime(candidate)) {
55:             primes.append(candidate + " ");
56:             numPrimes++;
57:         }
58:         candidate++;
59:     }
60: }
61:
62: public static boolean isPrime(int checkNumber) {
63:     double root = Math.sqrt(checkNumber);
64:     for (int i = 2; i <= root; i++) {
65:         if (checkNumber % i == 0) {
66:             return false;
67:         }
68:     }
69:     return true;
70: }
71:
72: private void setLookAndFeel() {
73:     try {
74:         UIManager.setLookAndFeel(
75:             "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
76:         );
77:     } catch (Exception exc) {
78:         // ignore error
79:     }
80: }
81: public static void main(String[] arguments) {
82:     PrimeFinder fp = new PrimeFinder();
83: }
84: }

```

应用程序 PrimeFinder 显示一个文本框、一个 Display primes 按钮和一个文本区域，如图 19.1 所示。

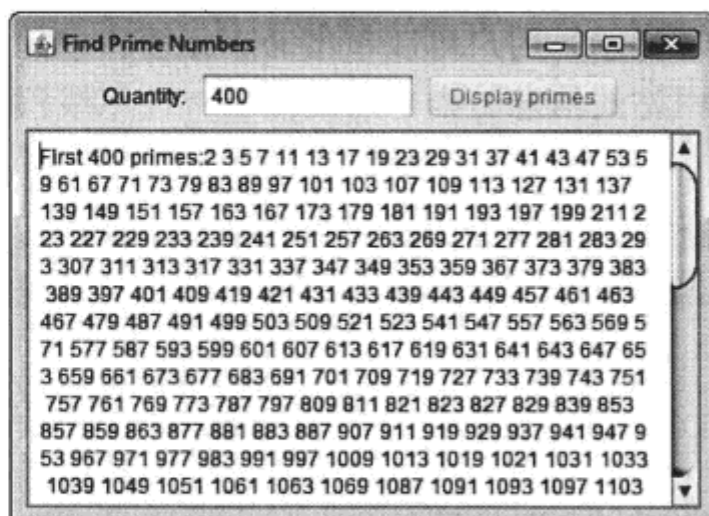


图 19.1

运行 PrimeFinder
应用程序

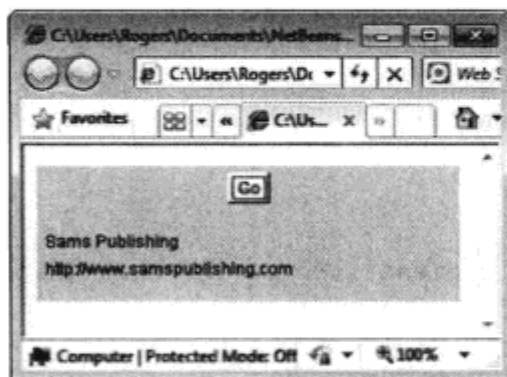
该应用程序中的大部分语句用于创建图形用户界面和显示一系列素数。下面是该程序中

用于实现线程的语句。

- 第 5 行：将 `Runnable` 接口应用到 `PrimeFinder` 类。
- 第 6 行：创建一个 `Thread` 对象变量，名字为 `go`，但是没有为其赋值。
- 第 41~44 行：如果对象变量 `go` 的值为 `null`，表示该线程还没有创建，因此创建一个新的 `Thread` 对象并将其存储到 `go` 变量中。通过调用线程的 `start()` 方法启动线程，这导致 `PrimeFinder` 类中的 `run()` 方法被调用。

图 19.2

运行应用程序
FindPrimes



- 第 47~60 行：`run()` 方法从 2 开始寻找一系列素数，并通过调用文本区域组件 `primes` 的 `append()` 方法显示每个素数。素数个数由 `howMany` 文本框中的值决定。

19.2 使用线程

可以通过调用 `start()` 方法启动线程，因此读者可能认为也有一个 `stop()` 方法用于终止线程。

尽管在 Java 的 `Thread` 类中包含 `stop()` 方法，但不提倡使用该方法。在 Java 中，不提倡使用的元素是指已被更好的东西取代的类、接口、方法或变量。

Watch Out!

警告：

读者应该重视有关不提倡使用的警告。Oracle 公司不提倡使用 `stop()` 方法，因为它可能给 Java 解释器中运行的其他线程带来问题。线程类的 `resume()` 方法和 `suspend()` 方法也不提倡使用。

下一个项目将演示如何终止线程。读者将编写的程序将在一个列表中循环，该列表包含网站名称及其网址。

每一个页面的标题和地址将循环显示。用户可以单击 applet 窗口中的按钮来访问当前显示的网站。该程序按顺序显示每个网站的信息。由于时间因素，使用线程来控制程序是最好的选择。

这次不是首先在 NetBeans 的源代码编辑器中输入程序然后再介绍它。本章结束时读者将有机会输入 `LinkRotator` applet 的全部代码，在此之前，将描述该程序的每个组成部分。

19.2.1 声明类

在这个 applet 中，首先需要使用 `import` 语句导入 `java.awt`、`java.net`、`java.applet`、`java.awt.event` 和 `javax.swing` 中的一些类。

使用 `import` 语句导入一些类后，就可以使用下面的语句创建 `applet`：

```
public class LinkRotator extends JApplet
    implements Runnable, ActionListener {
```

该语句将 `LinkRotator` 类声明为 `JApplet` 的子类，并指出它支持两个接口：`Runnable` 和 `ActionListener`。通过实现 `Runnable` 接口，将能够在该 `applet` 中使用 `run()` 方法来启动线程；`ActionListener` 接口让该 `applet` 能够响应用户使用鼠标执行的操作。

19.2.2 创建变量

在 `LinkRotator` 类中，首先要创建该类的变量和对象。创建两个包含 6 个元素的数组：一个名为 `pageTitle` 的 `String` 对象数组和一个名为 `pageLink` 的 `URL` 对象数组：

```
String[] pageTitle = new String[6];
URL[] pageLink = new URL[6];
```

`pageTitle` 数组用于存储要显示的 6 个网站的标题。`URL` 对象存储网站地址，`URL` 有记录网站地址以及在 Web 浏览器中根据地址加载网页所需的行为和属性。

最后 3 项工作是创建一个名为 `butterscotch` 的 `Color` 对象、一个名为 `current` 整型变量和一个名为 `runner` 的 `Thread` 对象：

```
Color butterscotch = new Color(255, 204, 158);
int current = 0;
Thread runner;
```

`Color` 对象表示在用于字体、用户界面组件和其他 Swing 可视对象的颜色，第 23 章将介绍如何使用 `Color` 对象。

变量 `current` 用于记录当前显示的网址，以便能够在网站之间循环。`Thread` 对象 `runner` 是该程序运行的线程。可调用 `runner` 对象的方法来启动、终止和暂停 `applet` 的运行。

19.3 从 init() 开始

`applet` 第一次运行时，将自动调用 `applet` 的 `init()` 方法。该方法用于给 `applet` 的两个数组赋值：`pageTitle` 和 `pageLink`。在这里还将创建一个显示在 `applet` 窗口中的可单击按钮。该方法由下面的语句组成：

```
public void init() {
    pageTitle = new String[] {
        "Sun's Java site",
        "Cafe au Lait",
        "JavaWorld",
        "Java in 24 Hours",
        "Sams Publishing",
        "Workbench"
    };
    pageLink[0] = getURL("http://java.sun.com");
    pageLink[1] = getURL("http://www.ibiblio.org/javafaq");
```

```

    pageLink[2] = getURL("http://www.javaworld.com");
    pageLink[3] = getURL("http://www.java24hours.com");
    pageLink[4] = getURL("http://www.sampublishing.com");
    pageLink[5] = getURL("http://workbench.cadenhead.org");
    Button goButton = new Button("Go");
    goButton.addActionListener(this);
    FlowLayout flow = new FlowLayout();
    setLayout(flow);
    add(goButton);
}

```

每一个网站的标题存储到 `pageTitle` 数组的 6 个元素中，这是用 6 个字符串来实现的。`PageLink` 数组的元素由 `getURL()` 方法返回的值来确定，`getURL()` 方法此时还没有创建。

`init()` 方法中的最后 7 条语句用于创建一个按钮并将其放在 `applet` 窗口中。该按钮的标签为文本 `Go`。

19.4 在创建 URL 时捕获错误

创建 `URL` 对象时，必须确保用于设置网址的文本为有效格式。`http://workbench.cadenhead.org` 和 `http://www.sampublishing.com` 都是有效的，但是 `http:www.javaworld.com` 不是有效的，因为少了 `//`。

`getURL(String)` 方法接受一个网络地址作为参数，然后返回一个表示该地址的 `URL` 对象。如果作为参数的字符串不是有效的地址，则返回 `null`。

```

URL getURL(String urlText) {
    URL pageURL = null;
    try {
        pageURL = new URL(getDocumentBase(), urlText);
    } catch (MalformedURLException m) {
        // do nothing
    }
    return pageURL;
}

```

在创建 `URL` 对象时，`try-catch` 语句块用来处理发生的任何 `MalformedURLException` 错误。由于在该错误抛出时，什么都没有发生，因此 `catch` 语句块只包含注释行。

19.5 在 `paint()` 方法中处理屏幕更新

当 `applet` 窗口需要更新时，将执行 `applet` 的 `paint()` 方法。可以在 `applet` 内手动调用 `paint()` 方法。

调用 `repaint()` 方法可以强制调用 `paint()` 方法。该方法告诉图形用户界面有变动发生，有必要显示更新。

`LinkRotator applet` 的 `paint()` 方法如下：

```

public void paint(Graphics screen) {
    Graphics2D screen2D = (Graphics2D) screen;
    screen2D.setColor(butterscotch);
}

```

```

screen2D.fillRect(0, 0, getSize().width, getSize().height);
screen2D.setColor(Color.black);
screen2D.drawString(pageTitle[current], 5, 60);
screen2D.drawString("" + pageLink[current], 5, 80);
}

```

该方法中的第一条语句创建一个 Graphics2D 对象，它表示 applet 窗口的可绘制区域。所有绘制操作都将通过调用该方法来完成。

Graphics2D 的 setColor() 方法用于选择随后的绘制操作使用的颜色。绘制填充整个 applet 窗口的矩形前，将颜色设置为奶油糖色。接下来将颜色设置为黑色，并在屏幕的(5, 60)和(5, 80)处分别显示一行文本。第一行是 pageTitle 数组的一个元素，第二行是 URL 对象的网址，该对象存储在 pageLink 数组中。current 变量用于决定显示这些数组中的哪个元素。

19.6 启动线程

在这个 applet 中，runner 线程将在 start() 方法被调用时启动，并在 stop() 方法被调用时停止。

start() 方法在 init() 方法执行后以及程序重新启动时运行。下面是 start() 方法：

```

public void start() {
    if (runner == null) {
        runner = new Thread(this);
        runner.start();
    }
}

```

如果 runner 线程还没有启动，该方法将创建并启动它。

语句 runner = new Thread(this); 使用一个参数（关键字 this）创建一个新的 Thread 对象，使用 this 使该 applet 运行在 runner 线程中。

语句 runner.start(); 导致线程开始运行。线程开始后，将调用线程的 run() 方法。由于 runner 线程是 applet 本身，因此将调用 applet 的 run() 方法。

19.6.1 运行线程

线程的主要工作是在 run() 方法中完成的。在 LinkRotator applet 中，run() 方法如下：

```

public void run() {
    Thread thisThread = Thread.currentThread();
    while (runner == thisThread) {
        current++;
        if (current > 5) {
            current = 0;
        }
        repaint();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            // do nothing
        }
    }
}

```

```

    }
}

```

`run()`方法做的第一件事情是创建一个名为 `thisThread` 的 `Thread` 对象，使用 `Thread` 类的类方法 `currentThread()` 给 `thisThread` 对象赋值，`currentThread()` 方法跟踪当前运行的线程。

该方法的所有语句都位于 `while` 循环中，该循环比较 `runner` 对象和 `thisThread` 对象，这两个对象都是线程，而且只要它们的值相同，`while` 循环将一直执行下去。该循环中没有导致 `runner` 和 `thisThread` 对象的值不同的语句，因此循环将不断进行下去，直到在循环外修改了其中一个线程对象。

`run()`方法首先使用 `repaint()` 语句，接下来将 `current` 变量的值加 1，如果 `current` 大于 5，则将其重置为 0。`paint()` 方法中的 `current` 变量用来决定显示哪个网站的信息。修改变量 `current` 将导致下次调用 `paint()` 方法时，显示不同网站的信息。

该方法包括一个 `try-catch` 语句，负责处理可能发生的错误。语句 `Thread.sleep(10000)`；让线程暂停 10 秒。该语句导致线程等待一段时间，让用户就有足够的时间阅读网站的名称和网址。`catch` 语句处理调用 `Thread.sleep()` 方法时可能发生的 `InterruptedException` 错误，如果在线程运行时将其中断，将发生这种错误。

19.6.2 停止线程

每当因为退出 applet 所在的页面而导致 applet 停止时，都将调用 `stop()` 方法，因此在该方法中停止运行的线程是最好的选择。`LinkRotator applet` 的 `stop()` 方法包含下面的语句：

```

public void stop() {
    if (runner != null) {
        runner = null;
    }
}

```

`if` 语句检测 `runner` 对象是否等于 `null`。如果是，表明没有需要停止的活动线程；否则，将 `runner` 设置为 `null`。

通过将 `runner` 对象设置为 `null`，`runner` 对象的值就与 `thisThread` 对象不同，进而导致 `run()` 方法中的 `while` 循环停止运行。

19.7 处理鼠标单击

`LinkRotator applet` 需要完成的最后一项工作是事件处理：每当用户单击 `Go` 按钮，`Web` 浏览器应打开列出的网站。这是使用 `actionPerformed()` 方法实现的，该方法在用户单击按钮时被调用。

下面是 `LinkRotator applet` 的 `actionPerformed()` 方法：

```

public void actionPerformed(ActionEvent event) {
    if (runner != null) {
        runner = null;
    }
    AppletContext browser = getAppletContext();
}

```



```

        if (pageLink[current] != null) {
            browser.showDocument(pageLink[current]);
        }
    }
}

```

该方法首先停止 `runner` 线程。下一条语句创建一个名为 `browser` 的 `AppletContext` 对象。`AppletContext` 对象表示 applet 所处的环境,即它所属的页面和加载该页面的 Web 浏览器。

`showDocument()` 方法将指定的网络地址载入到浏览器。如果 `pageLink[current]` 是有效的网址, `showDocument()` 方法将请求浏览器加载相应的网页。

19.8 循环显示链接

现在准备创建 `LinkRotator` 程序并进行测试。创建一个名为 `LinkRotator` 的 Java 空文件,然后输入程序清单 19.2 所示的内容。

程序清单 19.2 `LinkRotator.java` 程序的完整源代码

```

1: import java.applet.*;
2: import java.awt.*;
3: import java.awt.event.*;
4: import javax.swing.*;
5: import java.net.*;
6:
7: public class LinkRotator extends JApplet
8:     implements Runnable, ActionListener {
9:
10:    String[] pageTitle = new String[6];
11:    URL[] pageLink = new URL[6];
12:    Color butterscotch = new Color(255, 204, 158);
13:    int current = 0;
14:    Thread runner;
15:
16:    public void init() {
17:        pageTitle = new String[] {
18:            "Sun's Java site",
19:            "Cafe au Lait",
20:            "JavaWorld",
21:            "Java in 24 Hours",
22:            "Sams Publishing",
23:            "Workbench"
24:        };
25:        pageLink[0] = getURL("http://java.sun.com");
26:        pageLink[1] = getURL("http://www.ibiblio.org/javafaq");
27:        pageLink[2] = getURL("http://www.javaworld.com");
28:        pageLink[3] = getURL("http://www.java24hours.com");
29:        pageLink[4] = getURL("http://www.sampublishing.com");
30:        pageLink[5] = getURL("http://workbench.cadenhead.org");
31:        Button goButton = new Button("Go");
32:        goButton.addActionListener(this);
33:        FlowLayout flow = new FlowLayout();
34:        setLayout(flow);
35:        add(goButton);
36:    }
37:
38:    URL getURL(String urlText) {
39:        URL pageURL = null;
40:        try {

```

```

41:         pageURL = new URL(getDocumentBase(), urlText);
42:     } catch (MalformedURLException m) { }
43:     return pageURL;
44: }
45:
46: public void paint(Graphics screen) {
47:     Graphics2D screen2D = (Graphics2D) screen;
48:     screen2D.setColor(butterscotch);
49:     screen2D.fillRect(0, 0, getSize().width, getSize().height);
50:     screen2D.setColor(Color.black);
51:     screen2D.drawString(pageTitle[current], 5, 60);
52:     screen2D.drawString("" + pageLink[current], 5, 80);
53: }
54:
55: public void start() {
56:     if (runner == null) {
57:         runner = new Thread(this);
58:         runner.start();
59:     }
60: }
61:
62: public void run() {
63:     Thread thisThread = Thread.currentThread();
64:     while (runner == thisThread) {
65:         current++;
66:         if (current > 5) {
67:             current = 0;
68:         }
69:         repaint();
70:         try {
71:             Thread.sleep(10000);
72:         } catch (InterruptedException e) {
73:             // do nothing
74:         }
75:     }
76: }
77:
78: public void stop() {
79:     if (runner != null) {
80:         runner = null;
81:     }
82: }
83:
84: public void actionPerformed(ActionEvent event) {
85:     if (runner != null) {
86:         runner = null;
87:     }
88:     AppletContext browser = getAppletContext();
89:     if (pageLink[current] != null) {
90:         browser.showDocument(pageLink[current]);
91:     }
92: }
93: }

```

保存该程序之后，需要创建一个 Web 页面，将该 applet 放到其中。如果你在 NetBeans 中运行 Run->Run File 对其测试，它无法正常运行，原因是链接无法打开。创建新的 Web 页面：选择 File->New File，然后单击“Other”，在弹出的“Choose File Type”对话框中的 File Type 面板中找到 HTML 文件选项。将该 Web 页面命名为 LinkRotator，该文件将被 NetBeans 存储

为 LinkRotator.html，然后再输入程序清单 19.3 中的内容。

程序清单 19.3 LinkRotator.html 的完整源代码

```
1: <applet
2:     code="LinkRotator.class"
3:     codebase="..\build\classes"
4:     width="300"
5:     height="100"
6: >
7: </applet>
```

完毕之后，在 Project 面板中右键单击 LinkRotator.html 名字，然后选择 View。该页面将在 Web 浏览器中打开，而且该 applet 循环显示每一个链接。单击 Go 按钮可以浏览相应的站点。图 19.3 所示为 applet 在 IE 中运行的结果。

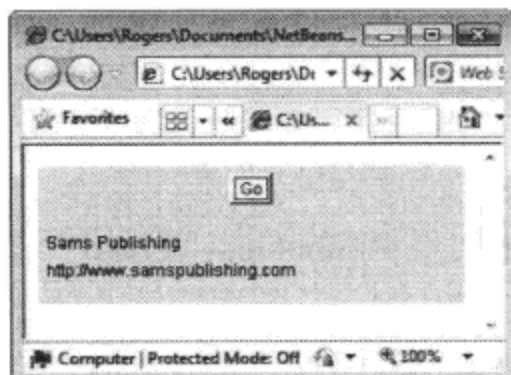


图 19.3

在 applet 窗口中
循环显示链接

19.9 总结

线程是一个功能强大的概念，在 Java 中通过几个类和接口来实现。通过在应用程序中支持多线程，可提高程序的响应速度及其执行任务的速度。

即使读者在本章中什么也没学到，但至少现在知道了一个用于描述狂热生活方式的新术语，那就是多线程（multithreading）。

19.10 问与答

问：在 LinkRotator applet 中的 catch 语句中不执行任何操作，这是为什么呢？

答：这取决于所捕获的错误或异常的类型。在 LinkRotator applet 中，由于你已经知道引发异常的原因是什么，因此 catch 语句中可以不执行任何操作，这都不会有问题。在 getURL() 方法中，只有当发送给该方法的 URL 无效，才会引发 MalformedURLException 异常。

19.11 测验

将你的线程放在一边，回答下列关于 Java 多线程的问题。

19.11.1 问题

1. 程序要使用线程，必须实现哪个接口？

a. Runnable。

- b. Thread。
 - c. Applet。
2. 如果接口包含 3 个不同的方法，在实现了该接口的类中必须包含其中的几个方法？
- a. 一个也不需要。
 - b. 所有的方法。
 - c. 我知道，但是我不说。
3. 你非常羡慕另一名程序员，他创建了一个同时处理 4 项任务的程序。你应对他说什么？
- a. 这不如我从网上下载的画面保护程序 Anna Kournikova 令人激动。
 - b. 你是我前进的动力。
 - c. 不错的线程（nice threads，也可以理解为衣服不错）。

19.11.2 答案

1. a. 必须使用关键字 `implements` 来实现 `Runnable`。`Thread` 用于多线程程序中，但不需要在程序开头的 `class` 语句中使用。
2. b. 接口要求实现它的类必须包括其所有方法。
3. c. 如果程序员穿着考究，这种恭维容易引起混淆，但说实在的，这种可能性有多大呢？

19.12 练习

如果本章内容还没有让读者感觉到劳累，请完成下面的练习以提高技能：

- 如果你熟悉 HTML，创建自己的主页，其中包含 `LinkRotator` applet 和 6 个最喜欢的网站。在页面中使用 applet 以及其他图形和文本。
- 在应用程序 `PrimeFinder` 中添加一个按钮，以便在计算素数时停止线程。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 20 章

读写文件

本章介绍如下内容：

- 将文件中的字节读取到程序中；
- 在计算机上创建新文件；
- 将字节数组存储到文件中；
- 更改存储在文件中的数据。

在计算机中有很多表示数据的方式，读者使用过的一种是创建并引用对象，对象包含以变量方式存储的数据，还包含使用数据来完成的方法。

要使用其他类型的数据，如存储在硬盘中的文件和 Web 服务器中的文档，可使用 `java.io` 包中的类。其中的 `io` 表示 `input/output`，这些类用于访问数据源，如硬盘、CD-ROM 或计算机内存。

可以使用称为流（将信息从一个地方传到另一个地方的对象）的通信系统，将数据传入程序或从程序中传出数据。

20.1 流

要在 Java 程序中永久地保存数据或要检索已存储的数据，至少必须有一个流。

“流”是一种对象，将信息从一个地方发送到另一个地方。其名称来自溪流，溪流可以将鱼、船和工业污染都从一个地方带到另一个地方。

流可以连接到各种数据源，包括计算机程序、硬盘、Internet 服务器、计算机内存和 DVD-ROM。由于它们都使用流，因此学会如何使用其中的一种数据后，就能够以相同的方式处理其他类型的数据。

本章将使用流来读写计算机上文件中的数据。

有如下两种类型的流。

- 输入流：从数据源读取数据。
- 输出流：将数据写入数据源。

所有输入和输出流都由字节组成，字节是 0~255 的整数。可以用这种格式来表示数据，如可执行程序、字处理文档和 MP3 音乐文件，但这只是字节可表示的一小部分数据。字节流用于读写这种数据。

By the Way

注意：

Java 类文件以字节码（bytecode）的形式存储为字节。Java 解释器运行字节码，而字节码并不一定是由 Java 语言生成。Java 解释器也能运行其他语言生成的字节码，包括 NetRexx 和 Jython 语言。Java 解释器也被称为“字节码解释器”。

一种更专用的数据表示方式是字符——字母、数字、标点符号等。读写文本数据源时，可以使用字符流。

无论是使用字节流、字符流还是其他类型的信息，整体过程是相同的：

- 创建一个与数据相关联的流对象；
- 调用流的方法，将信息加入流中或从流中取出信息；
- 调用流对象的 `close()` 方法关闭流。

20.1.1 文件

在 Java 中，文件用 `File` 类表示，它也位于 `java.io` 包中。可以读取硬盘、CD-ROM 或其他存储设备中的文件。

`File` 对象可以表示已有的文件或要创建的文件。要创建 `File` 对象，使用文件名作为构造函数的参数，如下例所示：

```
File bookName = new File("address.dat");
```

这条语句创建一个位于当前文件夹中的 `address.dat` 文件的对象，也可以在文件名中包括路径：

```
File bookName = new File("data\\address.dat");
```

By the Way

注意：

上述代码适用于 Windows 系统，它使用反斜线（\）字符作为文件名和路径的分隔符。Linux 和其他基于 UNIX 的系统使用斜线（/）字符作为分隔符。编写 Java 程序时，要以适用于任何操作系统的方式引用文件，可使用类变量 `File.pathSeparator` 而不是反斜线和斜线，如下面的语句所示：

```
File bookName = new File("data" + File.pathSeparator  
+ "address.dat");
```

有了 File 对象后，就可以调用该对象的几个有用的方法。

- `exist()`: 如果文件存在，返回 `true`；否则返回 `false`。
- `getName()`: 将文件名作为 `String` 返回。
- `length()`: 将文件长度作为 `long` 值返回。
- `createNewFile()`: 如果文件不存在，创建它。
- `delete()`: 如果文件存在，将其删除。
- `renameTo(File)`: 使用通过参数指定的 File 对象的名称来重命名文件。

也可以使用 File 对象表示系统中的文件夹而不是文件。为此，在构造函数 `File` 中指定文件夹名，这可以是绝对路径（如 `C:\\MyDocuments\\`），也可以是相对路径（如 `java\\database`）。

有了代表文件夹的对象后，就可以调用其 `listFiles()` 方法来看文件夹的内容。该方法返回一个 File 对象数组，表示文件夹包含的每个文件和子文件夹。

20.1.2 从流中读取数据

本章的第一个项目将使用输入流从文件中读取数据。为此，可使用 `FileInputStream` 类，它代表可通过它从文件中读取字节的输入流。

要创建文件输入流，可调用 `FileInputStream()` 构造函数并将一个文件名或 File 对象作为参数。

创建文件输入流时对应的文件必须存在，否则在创建该流时将引发 `IOException` 异常。很多与文件读写相关的方法都会引发这种异常，因此最好将与文件相关的所有语句放到 `try-catch` 块中，如下例所示：

```
try {
    File cookie = new File("cookie.web");
    FileInputStream stream = new FileInputStream(cookie);
    System.out.println("Length of file: " + cookie.length());
} catch (IOException e) {
    System.out.println("Could not read file.");
}
```

文件输入流以字节方式读取数据。要读取单个字节，可调用流的 `read()` 方法且不指定任何参数。如果因已到达文件尾导致流中没有可读取的字节，将返回字节值 -1。

读取输入流时，从流中的第一个字节开始读取，如文件的第一个字节。可以调用带有一个参数的 `skip()` 方法来跳过一些字节，该参数是一个 `int` 参数，表示要跳过的字节数。下面的语句跳过 `scanData` 流中接下来的 1024 个字节：

```
scanData.skip(1024);
```

如果要一次读取多个字节，可以这样做：

- 创建一个字节数组，其大小等于要读取的字节数；
- 使用该数组作为参数调用 `read()` 方法，将使用从流中读取的字节填充该数组。

下面创建一个从 MP3 音频文件读取 ID3 数据的应用程序。MP3 是一种非常流行的音乐文件格式，因此经常在 ID3 文件的末尾添加 128 个字节，用于存储与歌曲相关的信息，如歌名、创作者及其所属的唱片。

应用程序 ID3Reader 使用文件输入流来读取 MP3 文件，它跳过最后 128 个字节前的所有内容。然后检查余下的字节是否包含 ID3 数据，如果包含，前 3 个字节应为 84、65 和 71。

By the Way

注意：

Java 支持的 Unicode 标准字符集包含 ASCII 字符集，在 ASCII 字符集中，这 3 个数字分别代表大写字母 T、A 和 G。

创建一个名为 ID3Reader 的 Java 空文件，然后输入程序清单 20.1 中的所有文本。

程序清单 20.1 ID3Reader.java 的源代码

```

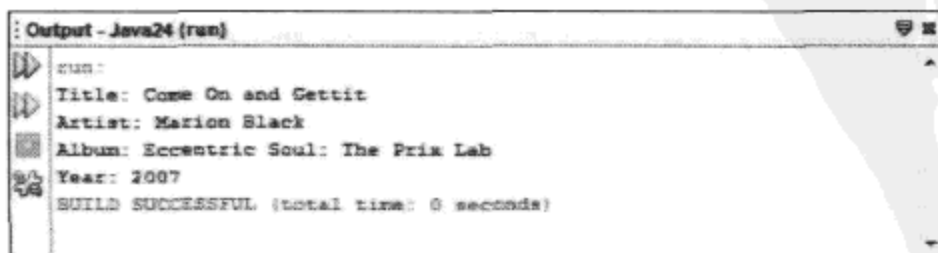
1: import java.io.*;
2:
3: public class ID3Reader {
4:     public static void main(String[] arguments) {
5:         try {
6:             File song = new File(arguments[0]);
7:             FileInputStream file = new FileInputStream(song);
8:             int size = (int) song.length();
9:             file.skip(size - 128);
10:            byte[] last128 = new byte[128];
11:            file.read(last128);
12:            String id3 = new String(last128);
13:            String tag = id3.substring(0, 3);
14:            if (tag.equals("TAG")) {
15:                System.out.println("Title: " + id3.substring(3, 32));
16:                System.out.println("Artist: " + id3.substring(33, 62));
17:                System.out.println("Album: " + id3.substring(63, 91));
18:                System.out.println("Year: " + id3.substring(93, 97));
19:            } else {
20:                System.out.println(arguments[0] + " does not contain"
21:                    + " ID3 info.");
22:            }
23:            file.close();
24:        } catch (Exception e) {
25:            System.out.println("Error - " + e.toString());
26:        }
27:    }
28: }

```

在作为应用程序运行该类之前，必须将一个 MP3 文件指定为命令行参数。该程序可以对任何 MP3 文件都生效，比如 Come on and Getit.mp3（这是由 Marion Black 演唱的一首歌曲）。如果你的系统中有这首歌曲（应该有），则图 20.1 就是 ID3Reader 应用程序运行时的输出。

图 20.1

运行 ID3Reader
应用程序



提示:

如果你的计算机中没有 Come on and Getit.mp3 文件 (在我看来, 这是严重的错误), 可在 Creative Commons 查找 MP3 歌曲, Creative Commons 可通过 Yahoo! Search (<http://search.yahoo.com/cc>) 搜索到。

Creative Commons 是一组版权许可协议, 规定了可如何分发、编辑或复制歌曲和书籍等作品。Rock Proper 站点 (www.rockproper.com) 提供了大量 MP3 专辑, 这些专辑可以在 Creative Commons 的许可下共享。

Did you know?

在程序清单 20.1 的第 10~11 行, 应用程序读取 MP3 文件的最后 128 个字节, 并将其存储到一个字节数组中。该数组在第 12 行被用于创建一个 String 对象, 后者包含这些字节表示的字符。

如果字符串中的前 3 个字符为 “TAG”, 表明该 MP3 文件包含的 ID3 信息是应用程序能够识别的格式。

在第 15~18 行, 调用字符串的 substring() 方法来显示字符串的各个部分。要显示的字符为 ID3 格式, 这种格式总是将艺术家、歌曲、曲名和年度等信息存储在 MP3 文件最后 128 字节的相同位置。

有些 MP3 文件不包含 ID3 信息或包含的 ID3 信息为应用程序不能识别的格式。

如果 Come on and Getit.mp3 文件来自你购买的 Eccentric Soul CD, 它将包含可读取的 ID3 信息, 因为使用音频 CD 创建 MP3 文件的程序从音乐行业数据库 Cddb 中读取歌曲信息。

从 MP3 文件的输入流中读取与 ID3 相关的信息后, 第 23 行关闭这个流。使用完流后, 应关闭它, 这样可以节省 Java 解释器的资源。

注意:

读者可能想从 BitTorrent 服务 (这是一种非常流行的文件共享服务) 中查找 Come on and Getit.mp3 文件的拷贝, 就这首 MP3 而言, 我非常理解这种做法。然而, 根据美国唱片行业协会的规定, 任何人下载不属于自己的 CD 的 MP3 文件, 将被起诉。因此, 最好先从 Amazon.com、eBay、Applet iTunes 以及其他主流零售商那里购买 Eccentric Soul CD。

By the way

20.1.3 缓冲输入流

对于读取输入流的程序, 提高其性能的一个方法是将输入放到缓冲区中。缓冲 (buffering) 是将数据放到内存中供程序需要时使用的一个过程。当 Java 程序需要缓冲输入流中的数据时, 将首先在缓冲区中查找, 这比从文件等数据源读取数据快。

要使用缓冲输入流, 需要创建一个输入流, 如 FileInputStream 对象, 然后使用该对象创建缓冲的流。为此, 将输入流作为唯一的参数调用 BufferedInputStream(InputStream) 构造函数, 这样, 从输入流中读取数据时, 数据将被存储到缓冲区中。

要从缓冲的流中读取数据, 可调用其 read() 方法且不指定任何参数。这将返回一个 0~255 的整数, 表示该流中下一字节的数据。如果没有字节可读取, 将返回 -1。

为了演示缓冲的流，将创建的下一个程序将使用 Java 的一项功能：控制台输入，这项功能在很多其他语言中是没有的。

控制台输入指的是程序运行时从控制台（也叫命令行）读取字符。

System 类包含一个 out 变量，语句 `System.out.print()` 和 `System.out.println()` 使用了该变量；它还有一个名为 in 的类变量，表示一个 `InputStream` 对象，该对象从键盘接收输入并以流的方式提供它们。

可以像使用其他输入流那样使用这个输入流，下面的语句创建一个缓冲的输入流，它与输入流 `System.in` 相关联：

```
BufferedInputStream bin = new BufferedInputStream(System.in);
```

接下来的项目（`Console` 类）包含一个类方法，可在任何 Java 程序中用于接收控制台输入。在名为 `Console` 的 Java 空文件中输入程序清单 20.2 中的所有文本。

程序清单 20.2 Console.java 的完整源代码

```
1: import java.io.*;
2:
3: public class Console {
4:     public static String readLine() {
5:         StringBuffer response = new StringBuffer();
6:         try {
7:             BufferedInputStream bin = new
8:                 BufferedInputStream(System.in);
9:             int in = 0;
10:            char inChar;
11:            do {
12:                in = bin.read();
13:                inChar = (char) in;
14:                if (in != -1) {
15:                    response.append(inChar);
16:                }
17:            } while ((in != -1) & (inChar != '\n'));
18:            bin.close();
19:            return response.toString();
20:        } catch (IOException e) {
21:            System.out.println("Exception: " + e.getMessage());
22:            return null;
23:        }
24:    }
25:
26:    public static void main(String[] arguments) {
27:        System.out.print("You are standing at the end of the road ");
28:        System.out.print("before a small brick building. Around you ");
29:        System.out.print("is a forest. A small stream flows out of ");
30:        System.out.println("the building and down a gully.\n");
31:        System.out.print("> ");
32:        String input = Console.readLine();
33:        System.out.println("That's not a verb I recognize.");
34:    }
35: }
```

`Console` 类包含一个 `main()` 方法，用于演示如何使用它。当运行该程序时，输出结果如图 20.2 所示。

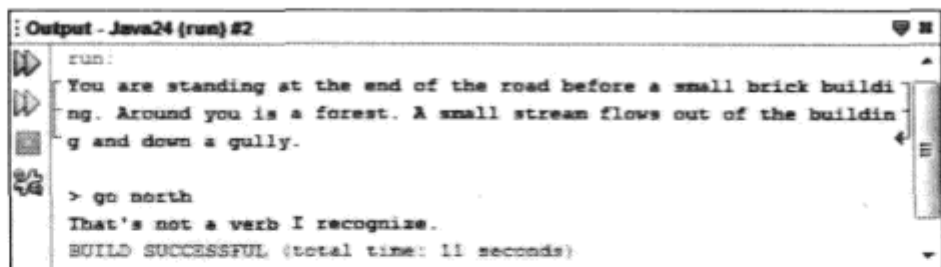


图 20.2
运行 Console 应用程序

Console 类包含一个类方法：`readLine()`，它从控制台接收字符。用户按回车键后，`readLine()`方法返回一个 `String` 对象，其中包含收到的所有字符。

如果将 Console 类放在 CLASSPATH 环境变量（在 Windows 上）列出的文件夹中，则可以从你编写的任何 Java 程序中调用 `Console.readLine()` 方法。

注意：

Console 类也是世界上最不能让人省心的一款文字冒险游戏，在该游戏中，你不能进入大楼，无法在溪流中涉水前进，也无法四处闲逛。该游戏也称为 Adventure，有关该游戏的完整版本，请访问互动小说文档(Interactive Fiction archive)，网址为 www.wurb.com/if/game/1。

By the Way

20.2 将数据写入流中

在 `java.io` 包中，与流相关的类都是成对出现的。对于字节流，有 `FileInputStream` 类和 `FileOutputStream` 类；对于字符流，有 `FileReader` 类和 `FileWriter` 类，还有很多其他成对的用于处理流数据的类。

要将数据写入字节流中，首先需要创建一个与输出流相关联的 `File` 对象。该文件不必是系统中现有的文件。

可以通过两种方式创建 `FileOutputStream`。如果要在现在的文件中追加字节，使用两个参数调用构造函数 `FileOutputStream()`：一个是代表文件的 `File` 对象，另一个是布尔值 `true`。这样，写入流中的字节就会追加到文件的末尾。

如果要将字节写入一个新文件中，只使用一个 `File` 对象作为参数调用构造函数 `FileOutputStream()`。

有了输出流后，就可以调用不同的 `write()` 方法来写入字节。

- 用一个字节作为参数调用 `write()` 方法时，将该字节写入流中。
- 用一个字节数组作为参数调用 `write()` 方法时，将数组的所有字节写入流中。
- 给 `write(byte[], int, int)` 方法指定 3 个参数：一个字节数组、一个表示要写入流中的数组的第一个元素的整数、要写入的字节总数。

下面的语句创建一个包含 10 个字节的字节数组，并将最后 5 个字节写入到输出流：

```
File dat = new File("data.dat");
FileOutputStream datStream = new FileOutputStream(dat);
byte[] data = new byte[] { 5, 12, 4, 13, 3, 15, 2, 17, 1, 18 };
datStream.write(data, 5, 5);
```

将数据写入到流中后，调用流的 `close()` 方法关闭它。

```
String name = "Puddin N. Tane";  
byte[] nameBytes = name.getBytes();
```

现在编写一个简单的应用程序，它通过将字节写入到文件输出流中的方式，将几行文本存储到一个文件中。创建一个名为 `ConfigWriter` 的 Java 空文件，然后输入程序清单 20.3 中的所有文本。

程序清单 20.3 `ConfigWriter.java` 的完整源代码

```
1: import java.io.*;  
2:  
3: class ConfigWriter {  
4:     String newline = System.getProperty("line.separator");  
5:  
6:     ConfigWriter() {  
7:         try {  
8:             File file = new File("program.properties");  
9:             FileOutputStream fileStream = new FileOutputStream(file);  
10:            write(fileStream, "username=max");  
11:            write(fileStream, "score=12550");  
12:            write(fileStream, "level=5");  
13:        } catch (IOException ioe) {  
14:            System.out.println("Could not write file");  
15:        }  
16:    }  
17:  
18:    void write(FileOutputStream stream, String output)  
19:        throws IOException {  
20:  
21:        output = output + newline;  
22:        byte[] data = output.getBytes();  
23:        stream.write(data, 0, data.length);  
24:    }  
25:  
26:    public static void main(String[] arguments) {  
27:        ConfigWriter cw = new ConfigWriter();  
28:    }  
29: }
```

运行该应用程序时，将创建一个名为 `program.properties` 的文件，该文件包含下面 3 行文本：

```
username=max  
score=12550  
level=5
```

20.3 读写配置属性

当使用命令行参数对 Java 程序进行配置之后（就像前面几章中创建的应用程序那样），Java 程序将更有用。`java.util` 包中有一个 `Properties` 类，可用于从源文件（文本文件）中载入配置设置。

在 Java 中，可以像其他文件那样读取属性文件：

- 创建一个代表该文件的 File 对象；
- 使用该 File 对象创建一个 FileInputStream 对象；
- 调用 load()方法从输入流中检索属性。

属性文件由一组属性名、等号和属性值组成，如下例所示：

```
username=lepton
lastCommand=open database
windowSize=32
```

每个属性占一行，因此上述内容将属性 username、lastCommand 和 windowSize 的值分别设置为 lepton、open database 和 32（ConfigWriter 类也使用相同的格式）。

下面的代码载入一个名为 config.dat 的属性文件：

```
File configFile = new File("config.dat");
FileInputStream inStream = new FileInputStream(configFile);
Properties config = new Properties();
config.load(inStream);
```

配置设置也称为属性，作为字符串存储在 Properties 对象中。每个属性都用一个类似于 applet 参数的键标识。方法 getProperty()根据键来检索属性，如下面的语句所示：

```
String username = config.getProperty("username");
```

由于属性被存储为字符串，要将其用作数字值，必须采用某种方式对其进行转换，如下面的代码所示：

```
String windowProp = config.getProperty("windowSize");
int windowSize = 24;
try {
    windowSize = Integer.parseInt(windowProp);
} catch (NumberFormatException exception) {
    // do nothing
}
```

要存储属性，可调用 setProperty()方法并指定两个参数——键和值：

```
config.setProperty("username", "max");
```

可以调用 Properties 对象的 list(PrintStream)方法来显示所有的属性。PrintStream 是 System 类中 out 类的变量。在本书前面，我们已经在 System.out.println()语句中使用 out 来显示输出。下面的代码将调用 list()方法来显示所有的属性。

```
config.list(System.out);
```

在对属性作出修改之后，可将其存回到文件中：

- 创建一个代表文件的 File 对象；
- 根据该 File 对象创建一个 FileOutputStream 对象；
- 调用方法 store(OutputStream, String)将属性存储到指定的输出流中，而且参数 String

是属性文件的描述。

接下来创建 `ConfigWriter` 应用程序，它将多个文件属性设置写入到一个文件中。而 `Configurator` 应用程序将这些属性设置读入到一个 Java 属性文件中，并添加一个名为 `runtime` 的新属性（具有当前日期和时间），然后保存该文件。

创建一个名为 `Configurator` 的 Java 空文件，然后输入程序清单 20.4 中的所有文本。

程序清单 20.4 `Configurator.java` 的完整源代码

```

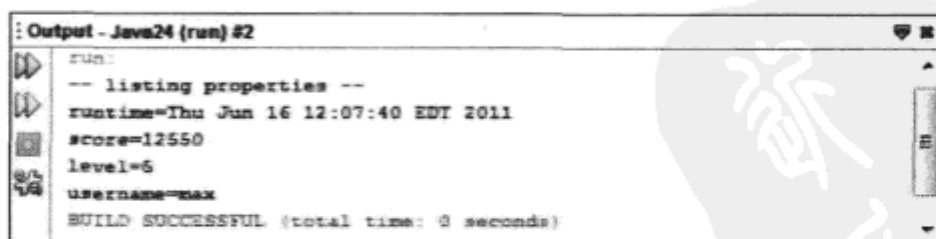
1: import java.io.*;
2: import java.util.*;
3:
4: class Configurator {
5:
6:     Configurator() {
7:         try {
8:             // load the properties file
9:             File configFile = new File("program.properties");
10:            FileInputStream inStream = new
11:            ↪FileInputStream(configFile);
12:            Properties config = new Properties();
13:            config.load(inStream);
14:            // create a new property
15:            Date current = new Date();
16:            config.setProperty("runtime", current.toString());
17:            // save the properties file
18:            FileOutputStream outStream = new
19:            ↪FileOutputStream(configFile);
20:            config.store(outStream, "Properties settings");
21:            inStream.close();
22:            config.list(System.out);
23:        } catch (IOException ioe) {
24:            System.out.println("IO error " + ioe.getMessage());
25:        }
26:    }
27:
28:    public static void main(String[] arguments) {
29:        Configurator con = new Configurator();
30:    }
31: }

```

运行该应用程序的输出结果如图 20.3 所示。

图 20.3

运行 `Configurator`
应用程序



文件 `program.properties` 现在应该包含如下文本：

Output ▼

```

#Properties settings
#Tue May 12 22:51:26 EDT 2009
runtime=Tue May 12 22\:51\:26 EDT 2009
score=12550
level=5
username=max

```


上面用到的反斜线字符 (\) 格式与应用程序的输出不同，可以确保属性文件被正确地存储。

20.4 总结

本章介绍了用于读写字节的输入流和输出流，这是通过流提供数据的最简单方式。

java.io 包有很多用于以其他方式处理流的类；还有一个名为 java.net 的包，它可以让你通过 Internet 连接读和写流。

字节流的用途广泛，因为可以轻松地将字节转换为其他数据类型，如整数、字符和字符串。

本章的第一个项目（应用程序 ID3Reader）从流中读取字节，然后将其转换为字符串，因为以这种格式从歌曲（如 Marian Black 在专辑 EccentricSoul 中演唱的 Come on and Getit）中读取 ID3 数据更加容易。

20.5 问与答

问：为什么本章的一些字节流方法使用整型参数？不是应使用字节型参数吗？

答：流中的字节与 byte 类表示的字节是有区别的，在 Java 中，byte 变量的取值范围为 -127~128，但流中的字节的取值范围为 0~255。因此，处理字节时，经常需要使用 int 类型，它可以存储 128~255 的值，而 byte 变量不能。

20.6 测验

为检验是否掌握了本章的大部分知识，请回答下面关于 Java 的输入流和输出流的问题。

20.6.1 问题

1. 下面哪种方法可用于将字节数组转换为字符串？
 - a. 调用数组的 toString() 方法。
 - b. 将每个字节转换为字符，再将每个字符赋给 String 数组的元素。
 - c. 用数组作为参数调用构造函数 String()。
2. 哪种流用于在 Java 程序中读取文件？
 - a. 输入流。
 - b. 输出流。
 - c. 都可以。
3. File 类的哪个方法可用于确定文件的长度？
 - a. getSize()。
 - b. read()。

c. `length()`。

20.6.2 答案

1. c. 可以像答案 b 那样单独处理每个字节，但是可以使用其他数据类型来更简单地创建字符串。
2. a. 使用 `File` 对象创建一个输入流，或调用输入流的构造函数并提供文件名作为参数来创建输入流。
3. c. 该方法返回一个 `long` 值，表示流中的字节数。

20.7 练习

为体验跋涉到另一条河流的新鲜感，通过下面练习来检测其水质：

- 编写一个应用程序，它读取一个文件夹中所有 MP3 文件的 ID3 标记，并使用创作者、曲名和唱片信息（如果有的话）来重命名这些文件。
- 编写一个应用程序，它读取一个 Java 程序的源文件，然后不加修改地写入到一个新文件中。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 21 章

读写 XML 数据

本章介绍如下内容：

- 从文件中读取 XML 数据；
- 提取 XML 元素；
- 收集一组子元素；
- 读取 XML 元素的属性值；
- 写入 XML 文件。

在 Java 于 20 世纪 90 年代崛起的同时，计算机软件开发也发生了重大变化，这就是可扩展标记语言（XML）的面世。XML 是一种用于组织和存储数据以便程序读取的格式，现在非常流行。

多亏了 XML，数据的读写可完全独立于用于创建它的软件。在以往，每个程序都有专用和特有的格式，因此这种改变受到了大家的欢迎。

XML 数据可以使用分析器（parser）来读取，分析器是一种能够识别 XML 格式并提取所需数据的程序。

本章将使用 XML 对象模型（XML Object Model, XOM）来读写 XML 数据，XOM 是一个 Java 类库，在 Java 程序中使用 XOM，可以方便地处理 XML 数据。

21.1 创建 XML 文件

在讲解 XOM 之前，应该学习一些有关 XML 及其如何存储数据的知识。XML 数据无处不在，可存储在文件中，可通过 Internet 网络传输，可以存储在内存中。

Java 类库中有多个可读写 XML 的类，其中包括 java.util 包中的 Properties 类，这个类在第 20 章介绍过。

Properties 对象可存储为 XML 格式，而不是前一章介绍的 name=value 格式。

当使用配置属性填充这种对象后，可以使用其 `storeToXML()` 方法将其保存到 XML 文件中。该方法接受下面两个参数。

- 一个 `FileOutputStream` 对象：通过它将数据存储到文件中。
- 一条注释：如果数据不需要注释，可以为空字符串“”。

本章的第一个项目是一个简单的应用程序 `PropertyFileCreator`，它将配置属性存储为 XML 格式。启动 `NetBeans`，在一个新的 Java 空文件中输入程序清单 21.1 中的文本，将其命名为 `PropertyFileCreator`，然后保存。

程序清单 21.1 `PropertyFileCreator.java` 的源代码

```

1: import java.io.*;
2: import java.util.*;
3:
4: public class PropertyFileCreator {
5:     public PropertyFileCreator() {
6:         Properties prop = new Properties();
7:         prop.setProperty("username", "rcade");
8:         prop.setProperty("browser", "Mozilla Firefox");
9:         prop.setProperty("showEmail", "no");
10:        try {
11:            File propFile = new File("properties.xml");
12:            FileOutputStream propStream = new
13:                FileOutputStream(propFile);
14:            Date now = new Date();
15:            prop.storeToXML(propStream, "Created on " + now);
16:        } catch (IOException exception) {
17:            System.out.println("Error: " + exception.getMessage());
18:        }
19:    }
20:    public static void main(String[] arguments) {
21:        PropertyFileCreator pfc = new PropertyFileCreator();
22:    }
23: }

```

运行该程序时，它创建一个包含 3 项设置的属性文件：username 为 rcade，browser 为 Mozilla Firefox，showEmail 为 no。

如果这些属性被存储为其他格式，则看起来会如下所示：

```

#Created on Wed Jun 15 20:56:33 EDT 2011
# Thu Wed Jun 15 20:56:33 EDT 2011
showEmail=no
browser=Mozilla Firefox
username=rcade

```

运行该应用程序时，它将创建如程序清单 21.2 所示的 XML 文件 `properties.xml`。

程序清单 21.2 `properties.xml` 文件

```

1: <?xml version="1.0" encoding="UTF-8"?>
2: <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3: <properties>
4: <comment>Created on Wed Jun 15 20:56:33 EDT 2011</comment>

```

```

5: <entry key="showEmail">no</entry>
6: <entry key="browser">Mozilla Firefox</entry>
7: <entry key="username">rcade</entry>
8: </properties>

```

XML 以自我记录 (self-documenting) 的方式组织数据, 这样只需阅读文件本身就能理解其中的数据。

通过观察程序清单 21.2, 就可以立即知道它是如何存储配置属性的, 标记?xml 和!DOCTYPE 可能不好理解, 但文件中的其他内容很容易理解。

XML 文件中的数据用标记括起, 很像用于编写网页的标记语言 HTML。

起始标记以 “<” 打头, 然后是元素名和字符 “>”, 如程序清单 21.2 中第 3 行的 <properties>。

结束标记以字符 “<” 打头, 然后是 “/”, 随后是元素名和 “>”, 如第 8 行的 </properties>。

位于起始标记和结束标记之间的内容是元素的值。

XML 数据必须有一个根元素, 它将所有数据括起。在程序清单 21.2 中, 根元素为第 3~8 行定义的 properties 元素。

元素可以包含文本以及一个或多个子元素。元素 properties 包含 4 个子元素: 1 个 comment 元素和 3 个 entry 元素。

下面是其中的 comment 元素:

```
<comment>Created on Wed Jun 15 20:56:33 EDT 2011</comment>
```

该元素包含文本值 Created on Wed Jun 15 20:56:33 EDT 2011。

XML 元素还可以有一个或多个属性, 它们是在元素的起始标记中以 name=“value”的方式定义的。属性之间必须用空格分开, 它们提供了与元素相关的补充信息。

每个 entry 元素都有一个属性和一个值:

```
<entry key="showEmail">no</entry>
```

该元素的值为 no, 它还有一个值为 showEmail 的 key 属性。

程序清单 21.2 中没有的一种 XML 元素是, 在单个标记内定义的元素。这些元素以字符 “<” 打头, 然后是元素名和字符 “/>”。

例如, 下面的元素可作为 properties 元素的子元素:

```
<inactive />
```

虽然 XML 被称为一种类似于 HTML 的格式, 但它本身并非一种语言。相反, XML 描述如何根据要使用计算机程序完成的任务创建数据格式。XML 格式被称为方言 (dialect)。

一个这样的例子是 Java 的 Properties 类创建的 XML 方言。Oracle 公司开发这种格式用于表示软件配置设置。

遵循 XML 格式化规则的数据被称为格式良好的 (well-formed), 读写 XML 的软件必须能够接受格式良好的数据。

数据也可以遵循更严格的标准, 这种标准被称为有效性 (validity)。有效的 XML 文件在正确的位置包含正确的元素, 这要求采取某种方法来定义哪些元素是有效的。

21.2 读取XML文件

在本书前 20 章中可以看到，很多 Java 代码都已经编写好了，因此你的工作也得以大大简化。在 Java 类库内，你可以采用 Swing 类来进行用户界面编程，采用 java.io 类来进行文件的访问，采用 java.awt.event 来接受用户输入，你还可以使用其他类。通过使用这些类，你的编码负担大大降低，只需编写必要的代码即可。

需要学习的一种重要的 Java 编程技能是，找到可在项目中使用的 Java 类和包。重用开发好的类库要比从头编写自己的类简单得多。

Oracle 并非唯一一家开发大量 Java 类的开发商，读者在本章后面使用 XML 对象模型 (XOM) 时将看到，计算机程序员兼图书作者 Elliott Rusty Harold 也开发了一个类库。Harold 是 Java 语言和 XML 方面的专家，对现有 XML 类库的工作原理感到失望（读者可能明白了这里的含义，Java 本身是因 James Gosling 对另一种语言感到失望而开发的）。

Harold 创建了自己的类库，用树结构表示 XML 数据，并将每个元素作为节点。

你可以从 www.xom.nu 上下载这个类库。

在你的计算机的某个文件夹中解压这个文件包，我使用的是 Windows XP 系统，使用的文件夹是 C:\java\XOM，这将顶层文件夹 C:\java 指定为我所使用的 Java 库。在下载并解压该类库之后，必须将它添加到 NetBeans 中的当前项目中。

1. 选择 File->Project Properties，打开 Project Properties 对话框。
2. 在 Categories 面板中单击 Libraries，然后单击 Add Library 按钮，打开 Add Library 对话框。
3. 单击 Create 按钮，打开 Create New Library 对话框。
4. 在 Library Name 文本框中输入 XOM，然后单击 OK 按钮，打开 Customize Library 对话框。
5. 单击 Add JAR/Folder，打开 Browser JAR/Folder 对话框。
6. 找到存储 XOM 文件夹，然后选择 xom-1.2.1 和 xom-samples 文件（XOM 的版本号可能不同），然后单击 Add JAR/Folder。
7. 在 Customize Library 对话框中，单击 OK 按钮。
8. 在 Add Library 对话框中选择 XOM，然后单击 Add Library。
9. 在 Project Properties 对话框中，单击 OK。

现在，XOM 库可在你的项目中使用。

XOM 包含可用于读写 XML 数据并将其存储到文件或其他目的地的类。

Watch Out!

警告：

XOM 以开源许可 GNU LGPL (Lesser General Public License, 更严格的通用公共许可) 方式免费提供。你可以在不修改依附于 XOM 库的 Java 应用程序的前提下，分发 XOM 库。

你当然也可以对这个库中的类进行修改，但必须以 LGPL 方式提供修改后的库。有关这种许可的更详细信息，请访问 www.xom.nu/license.xhtml。

接下来创建应用程序 WeatherStation，它能够读取 Weather Underground 网站（www.wunderground.com）以 XML 方言形式提供的天气信息。

注意：

网站地址 <http://tinyurl.com/rd4r72> 是一个可以重定向到 Weather Underground 站点实际地址的简短 URL。Weather Underground 的实际地址很长，因此难以正确输入。它的一个完整地址如下所示：

<http://wunderground.com/auto/wui/geo/ForecastXML/index.xml?query=Wasilla,AK>

上述地址可以显示 Wasilla, Alaska 的天气预报信息。

By the Way

XOM 类库中的核心类位于 `nu.xom` 包中，要在程序中使用这些类，可使用下面的 `import` 语句：

```
import nu.xom.*;
```

`Builder` 类能够加载并分析使用任何方言的 XML 数据，只要它是格式良好的。

下面的代码创建一个 `Builder` 对象并使用它加载天气文件：

```
File file = new File("forecast.xml");
Builder builder = new Builder();
Document doc = builder.build(propFile);
```

XOM 也可以加载 Web 上的 XML 数据。不过它调用的 `build()` 方法是将数据所在的网络地址作为参数，如下面的代码所示：

```
Builder builder = new Builder();
Document doc = builder.build("http://tinyurl.com/rd4r72");
```

当 `builder` 对象加载 XML 数据时，以 `Document` 对象的方式提供它，该对象存储了整个 XML 文档。

要检索文档的根元素，可使用 `Document` 对象的 `getRootElement()` 方法：

```
Element root = doc.getRootElement();
```

`Element` 类表示单个元素，可使用 `Element` 类的多个方法来查看元素的内容：

- 方法 `getFirstChildElement()` 提取与指定名称相同的第一个子元素；
- 方法 `get(int)` 读取与指定索引（从 0 开始编号）匹配的元素；
- 方法 `getChildElements()` 提取所有的子元素；
- 方法 `getValue()` 读取元素的文本；
- 方法 `getAttribute()` 检索元素的一个属性。

下面的语句检索 `comment` 元素及其值：

```
Element highF = high.getFirstChildElement("fahrenheit");
String highTemp = highF.getValue();
```

在多个元素的名称相同时（如 `forecastday` 元素），这种方法将不管用。对于这样的元素，

可检索所有的元素并使用 for 循环遍历它们：

```
Elements days = root.getChildElements("simplifieday");
for (int current = 0; current < days.size(); current++) {
    Element day = days.get(current);
}
```

该程序没有使用属性，但是可使用方法 `getAttribute()` 来访问元素的属性，该方法将属性名作为参数：

```
Attribute key = day.getAttribute("key");
```

有了属性后，就可以使用其 `getValue()` 方法来获取其对应的值：

```
String keyValue = key.getValue();
```

创建一个名为 `WeatherStation` 的 Java 空文件，然后输入程序清单 21.3 中的所有文本。

程序清单 21.3 WeatherStation.java 的源代码

```
1: import java.io.*;
2: import nu.xom.*;
3:
4: public class WeatherStation {
5:     int[] highTemp = new int[6];
6:     int[] lowTemp = new int[6];
7:     String[] conditions = new String[6];
8:
9:     public WeatherStation() throws ParsingException, IOException {
10:         // get the XML document
11:         Builder builder = new Builder();
12:         Document doc = builder.build("http://tinyurl.com/rd4r72");
13:         // get the root element, <forecast>
14:         Element root = doc.getRootElement();
15:         // get the <simpleforecast> element
16:         Element simple = root.getFirstChildElement("simpleforecast");
17:         // get the <forecastday> elements
18:         Elements days = simple.getChildElements("forecastday");
19:         for (int current = 0; current < days.size(); current++) {
20:             // get current <forecastday>
21:             Element day = days.get(current);
22:             // get current <high>
23:             Element high = day.getFirstChildElement("high");
24:             Element highF = high.getFirstChildElement("fahrenheit");
25:             // get current <low>
26:             Element low = day.getFirstChildElement("low");
27:             Element lowF = low.getFirstChildElement("fahrenheit");
28:             // get current <icon>
29:             Element icon = day.getFirstChildElement("icon");
30:             // store values in object variables
31:             lowTemp[current] = -1;
32:             highTemp[current] = -1;
33:             try {
34:                 lowTemp[current] = Integer.parseInt(lowF.getValue());
35:                 highTemp[current] =
36:                     Integer.parseInt(highF.getValue());
37:             } catch (NumberFormatException nfe) {
38:                 // do nothing
39:             }
40:             conditions[current] = icon.getValue();
41:         }
42:     }
43: }
```

```

40:     }
41: }
42:
43: public void display() {
44:     for (int i = 0; i < conditions.length; i++) {
45:         System.out.println("Period " + i);
46:         System.out.println("\tConditions: " + conditions[i]);
47:         System.out.println("\tHigh: " + highTemp[i]);
48:         System.out.println("\tLow: " + lowTemp[i]);
49:     }
50: }
51:
52: public static void main(String[] arguments) {
53:     try {
54:         WeatherStation station = new WeatherStation();
55:         station.display();
56:     } catch (Exception exception) {
57:         System.out.println("Error: " + exception.getMessage());
58:     }
59: }
60: }

```

应用程序 `WeatherStation` 不需要命令行参数，它将生成 Wasilla, Alaska 的 6 个天气预报，如下面的输出所示：

Output ▼

```

Period 0
  Conditions: rain
  High: 56
  Low: 40
Period 1
  Conditions: partlycloudy
  High: 59
  Low: 40
Period 2
  Conditions: partlycloudy
  High: 61
  Low: 41
Period 3
  Conditions: chancerrain
  High: 67
  Low: 43
Period 4
  Conditions: chancerrain
  High: 67
  Low: 47
Period 5
  Conditions: chancerrain
  High: 65
  Low: 49

```

注意：

Java 类库包含用于 XML 处理的 Java API (JAXP)，该类与 XOM 的目的相同。JAXP 可将 XML 数据表示为对象或事件流，这可以让程序员更好地控制数据的分析方式。XOM 更容易学习，但是它在任何时候都需要格式正确而且有效的 XML。有关 JAXP 的详情，将访问 <http://jaxp.java.net>。

**By the
Way**

PDG

21.3 读取 RSS 聚合内容 (Syndication Feeds)

有数百种 XML 方言，以独立于平台和软件的方式表示数据。其中最流行的一种是 RSS，它是一种用于共享新闻摘要和链接的格式，这些信息摘要和链接来自在线新闻网站、博客和其他信息源。

RSS 使得能够以 XML 格式提供 Web 内容，非常适合以通过网络能够访问的文件（被称为 feed）方式提供。RSS 阅读器，又称为新闻聚合器，已被数以百万计的信息搜集者用于跟踪最喜欢的网站。还有搜集和共享 RSS 内容的 Web 应用程序。

nu.xom 包中的 Builder 类可通过 Internet 从任何 URL 加载 XML：

```
String rssUrl = "http://feeds.drudge.com/retort";
Builder builder = new Builder();
Document doc = builder.build(rssUrl);
```

这里将使用这种技术来读取一个 RSS 文件，并显示最新的 15 个条目。

在 NetBeans 中创建一个名为 Aggregator 的 Java 空文件，然后输入程序清单 21.4 中的所有文本，并保存。

程序清单 21.4 Aggregator.java 的源代码

```
1: import java.io.*;
2: import nu.xom.*;
3:
4: public class Aggregator {
5:     public String[] title = new String[15];
6:     public String[] link = new String[15];
7:     public int count = 0;
8:
9:     public Aggregator(String rssUrl) {
10:        try {
11:            // retrieve the XML document
12:            Builder builder = new Builder();
13:            Document doc = builder.build(rssUrl);
14:            // retrieve the document's root element
15:            Element root = doc.getRootElement();
16:            // retrieve the root's channel element
17:            Element channel = root.getFirstChildElement("channel");
18:            // retrieve the item elements in the channel
19:            if (channel != null) {
20:                Elements items = channel.getChildElements("item");
21:                for (int current = 0; current < items.size(); current++) {
22:                    if (count > 15) {
23:                        break;
24:                    }
25:                    // retrieve the current item
26:                    Element item = items.get(current);
27:                    Element titleElement = item.getFirstChildElement("title");
28:                    Element linkElement = item.getFirstChildElement("link");
29:                    title[current] = titleElement.getValue();
30:                    link[current] = linkElement.getValue();
31:                    count++;
32:                }
33:            }
```

```

34:         } catch (ParsingException exception) {
35:             System.out.println("XML error: " + exception.getMessage());
36:             exception.printStackTrace();
37:         } catch (IOException ioException) {
38:             System.out.println("IO error: " + ioException.getMessage());
39:             ioException.printStackTrace();
40:         }
41:     }
42:
43:     public void listItems() {
44:         for (int i = 0; i < 15; i++) {
45:             if (title[i] != null) {
46:                 System.out.println("\n" + title[i]);
47:                 System.out.println(link[i]);
48:                 i++;
49:             }
50:         }
51:     }
52:
53:     public static void main(String[] arguments) {
54:         if (arguments.length > 0) {
55:             Aggregator aggie = new Aggregator(arguments[0]);
56:             aggie.listItems();
57:         } else {
58:             System.out.println("Usage: java Aggregator rssUrl");
59:         }
60:     }
61: }

```

在运行该应用程序之前，将命令行参数设置为你喜欢阅读的 feed，它可以是任何 RSS feed。如果你对此不了解，可以使用 <http://feeds.drudge.com/retort>，它包含来自 Drudge Retort（一个在线新闻网站）的新闻标题。

运行该应用程序后，读取自该 feed 的输出如图 21.1 所示。

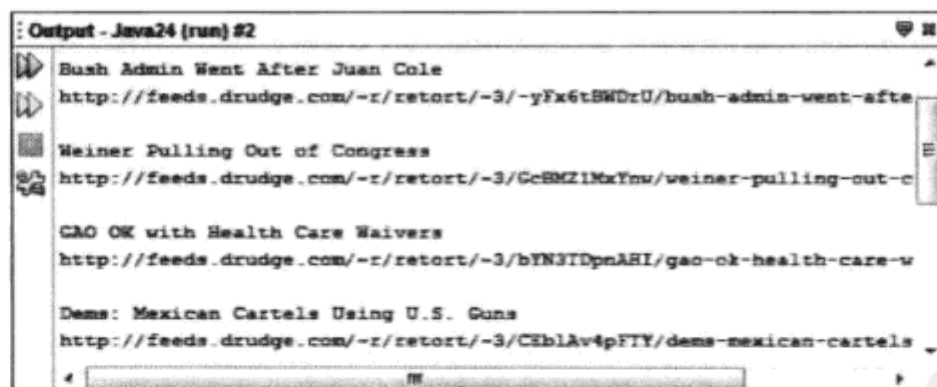


图 21.1

运行 Aggregator
应用程序

21.4 总结

Java 语言让软件能够独立于操作系统。在 Windows 中，使用该语言编写的程序经编译后生成的类文件，可在 Linux 服务器或 Mac OS X 计算机上运行。

XML 给软件生成的数据带来了类似的解放。如果 XML 数据遵循了使其成为格式良好的规则，将可以使用任何分析 XML 的软件读取它，而不必使用创建它的程序。

XML 对象模型（XOM）库使得读写 XML 数据更加容易。

当你使用 Java 和 XML 编程之后，就可以宣称自己不依赖于计算机程序员几十年来一直面临的两个障碍：陈旧的数据和操作系统。

21.5 问与答

问：在应用程序 PropertyFileCreator 生成的 XML 文件中，DOCTYPE 语句有什么作用？

答：这是对一种文档类型定义（DTD）的引用，DTD 是一个文件，定义了要使 XML 数据有效而必须遵循的规则。

如果打开该语句引用的网页 <http://java.sun.com/dtd/properties.dtd>，将看到有关 Java 库中的 Properties 类生成的 XML 文件中每个元素和属性的说明。

虽然 Sun 公司提供了该 DTD，但 Java 官方文档指出，不应依赖它来分析属性配置数据。分析器将忽略它。

21.6 测验

请回答以下问题，以测试对 XML 的理解程度。

21.6.1 问题

- 下面哪个术语不应用来描述被正确格式化的 XML 数据？
 - 格式良好。
 - 有效。
 - dy-no-mite。
- 哪个方法读取父元素的所有子元素？
 - get()。
 - getChildElements()。
 - getFirstChildElement()。
- Elements 类的哪个方法用于确定它包含的子元素数？
 - count()。
 - length()。
 - size()。

21.6.2 答案

- c. 格式良好的数据正确组织了起始标记和结束标记，使用单个根元素包含所有的子元素，开头有？XML 声明。有效的数据遵循了特定 XML 方言的规则。Dy-no-mite 是喜剧演员 Jimmie J.J Walker 的口头禅。

2. b. 方法 `getChildElements()` 返回一个 `Elements` 对象，其中包含所有的子元素。
3. c. 和 `Vector` 一样，`Elements` 也使用 `size()` 方法来确定其包含的子元素数。

21.7 练习

为拓展读者有关可扩展标记语言的知识，请完成下面的练习：

- 修改 `WeatherStaton` 应用程序，使其显示一个来自 `Weather Underground` 预报数据中的附加元素。
- 编写一个程序，它显示 `shortChanges.xml` 包含的数据，这是一个包含博客信息的 XML 文当，网址为 `www.weblogs.com/shortChanges.xml`。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 `www.java24hours.com`。



第 22 章

利用 JAX-WS 开发 Web 服务

本章介绍如下内容：

- 定义 Web 服务的 Java 接口；
- 将接口应用到 Java 类；
- 在 Internet 上部署 Web 服务；
- 查看 Web 服务合约；
- 开发 Web 服务客户端。

由于 Internet 无处不在，将数以百万计的桌面计算机、Web 服务器、视频游戏机，以及其他设备互连起来的诉求，催生了 Web 服务。Web 服务是一种经由 HTTP（网络协议）与其他软件进行通信的软件。

Java 最激动人心的一个特性是用于 XML Web 服务的 Java API (JAX-WS)。JAX-WS 是一组 Java 类和 Java 包，它可以创建对 Web 服务发出请求的客户端，以及接受这些请求的服务。

JAX-WS 支持使用简单对象访问协议（Simple Object Access Protocol, SOAP）和表述性状态转移（Representational State Transfer, REST）实现的 Web 服务。JAX-WS 大大简化了支持这些协议的任务。作为一名程序员，你只需要创建 Java 对象，并调用方法来使用 Web 服务即可，其他事宜将在幕后进行处理。

22.1 定义服务端点接口

创建 JAX-WS Web 服务的第一步是创建服务端点接口（Service Endpoint Interface），这是一个 Java 接口，它定义了客户端在使用 Web 服务时能够调用的方法。

本章将要开发的 SquareRootServer Web 服务可以处理两类简单的任务：

- 计算一个数的平方根；
- 显示当前的日期和时间。

接口是一组方法，它提供了名称、参数和返回类型，但是没有包含实现方法的代码。接口充当对象之间的合约：如果一个对象实现了该接口，其他对象就会知道它们可以调用该对象中接口的所有方法。

在第 15 章中，我们已经在任何这样的 Java 类中实现了 `ActionListener` 接口，即当有按钮被单击时，需要接收操作事件的 Java 类。

在这个项目中，我们要处理的是合约的其他方面。`SquareRootServer` 接口定义了 `squareRoot(double)` 和 `getTime()` 这两个方法，而且这两个方法必须出现在实现了 Web 服务的类中。

下面的语句定义了接口：

```
public interface SquareRootServer {
    double getSquareRoot(double input);
    String getTime();
}
```

在接口中定义方法时，跟在方法后面的是分号而不是将语句块括起来的“{”和“}”字符。接口中没有定义方法的行为，而是由实施了该接口的类来处理。

由于这些方法被称为 JAX-WS Web 服务，因此必须在每个方法前面添加一个额外的修饰词：`@WebMethod` 注解。

```
public interface SquareRootServer {
    @WebMethod double getSquareRoot(double input);
    @WebMethod String getTime();
}
```

使用注解来简化 Java 代码

注解（annotation）是一种巧妙的注释（comment）形式，可以被 Java 解释器、编译器和编程工具所理解。注解可以定义不属于程序一部分的程序信息，使得当程序在编译或运行时，这些程序信息能够触发行为。

注解使用@符号打头，后面是注解的名字。

最常见的一个注解是`@Override`，它表示一个方法覆盖了超类的方法。下面是一个例子：

```
@Override public void paintComponent(Graphics comp) {
    // definition of method here
}
```

如果期间有错误，则它不会覆盖方法，但是当使用了错误的类型或者参数个数不一致时，将发生错误，编辑器可以捕获该错误。

`@WebMethod` 注解表示一个方法可以作为 Web 服务调用。`SquareRootServer` 接口也使用了一个`@WebService` 注解，来指示接口定义了一个服务端点接口。

注解也可以接受供将来自定义使用的参数。`SquareRootServer` 包含一个最终（final）注解：

```
@SOAPBinding(style = Style.RPC)
```

该注解在 Web 服务和调用该服务的客户端程序之间定义了一个合约，本章后面将会详细介绍。

现在，开始编写 Web 服务的代码。在 NetBeans 中创建一个 Java 空文件，其类名为 `SquareRootServer`，其包名为 `com.java24hours.ws`。然后在该文件中输入程序清单 22.1 中的内容。

程序清单 22.1 `SquareRootSever.java` 的完整源代码

```
1: package com.java24hours.ws;
2:
3: import javax.jws.*;
4: import javax.jws.soap.*;
5: import javax.jws.soap.SOAPBinding.*;
6:
7: @WebService
8:
9: @SOAPBinding(style = Style.RPC)
10:
11: public interface SquareRootServer {
12:     // get the square root of a number
13:     @WebMethod double getSquareRoot(double input);
14:
15:     // get the current time and date as a string
16:     @WebMethod String getTime();
17:
18: }
```

该类位于 `com.java24hours.ws` 包中，这样在部署 Web 服务之后，其他软件能够通过 Internet 轻松访问该服务。

完成接口的定义之后，开始编写实现接口两个方法（`getSquareRoot()`和 `getTime()`）的代码。

22.2 创建服务实现 Bean

实现了服务端点接口的 Java 类被称为服务实现 Bean（Service Implementation Bean）。在 JAX-WS 的学习过程中，遇到一些新奇的术语是不可避免的。

`SquareRootServerImpl` 类实现了 `SquareRootServer` 接口，如下所示：

```
public class SquareRootServerImpl implements SquareRootServer {
```

这表示，你要创建的类必须包含接口中的所有方法，而且每一个都有适当的参数。

方法 `getSquareRoot(double)` 和 `getTime()` 是使用前面学习的技术来实现的。

类中唯一比较新鲜的地方是后面的注解，它出现在类语句之前：

```
@WebService(endpointInterface = "com.java24hours.ws.SquareRootServer")
```

该注解表示，类是名为 `com.java24hours.ws.SquareRootSever` 的服务端点接口的服务实现

Bean。你必须使用完整的类名，其中包括其包的名字。

注意，注解后面跟的不是分号，这与语句不同。

开始编写该类：在 `com.java24hours.ws` 包中创建一个名为 `SquareRootServerImpl` 的 Java 空文件，然后输入程序清单 22.2 中的所有内容。

程序清单 22.2 `SquareRootServerImpl.java` 的完整源代码

```

1: package com.java24hours.ws;
2:
3: import java.util.*;
4: import javax.xml.ws.*;
5:
6: @WebService(endpointInterface = "com.java24hours.ws.SquareRootServer")
7:
8: public class SquareRootServerImpl implements SquareRootServer {
9:
10:     public double getSquareRoot(double input) {
11:         return Math.sqrt(input);
12:     }
13:
14:     public String getTime() {
15:         Date now = new Date();
16:         return now.toString();
17:     }
18: }
```

在创建了这两个类之后，即可准备启用该服务，以便其他软件来调用。

警告：

服务实现 Bean 的名字源于 JavaBeans，后者是一个特殊的 Java 类，在 Java 企业版中用作可重用的软件组件。然而，就 JAX-WS 而言，对 Bean 的命名有点不是很合适。任何 Java 对象，只要遵循 Web 服务方法和规则，而且创建时带有合适的注解，就可以作为服务实现 Bean。

**Watch
Out!**

22.3 发布 Web 服务

JAX-WS Web 服务可以使用诸如 BEA WebLogic、GlassFish、JBoss 和 Jetty 这样的 Java 应用服务器来部署。如果在支持这些服务器的开发环境中创建了 `SquareRootServer` Web 服务，此时就可以准备启用 Web 服务。

你也可以自己编写 Java 应用程序，使其载入 Web 服务，并通过 Internet 向其他客户端提供服务。

`SquareRootServerPublisher` 应用程序只需要两个步骤就可以处理该任务：

- 载入实现了 Web 服务的类；
- 将该对象发布到 Internet。

`javax.xml.ws` 包中的 `EndPoint` 类有一个类方法 `publish(String, Object)`，它可以部署 Web 服务。

该方法的第一个参数是可以访问 Web 服务的网络地址, 对该项目而言, 该网络地址是 `http://127.0.0.1:5335/service`。该网络地址以主机名 127.0.0.1 打头, 这被称为本地主机, 因为该主机是你创建并运行 Java 程序的本地计算机。

网络地址的第 2 部分是本地主机的端口号, Web 服务在该端口号等待连接。这里使用的端口号是 5335, 因为该端口号不太可能被计算机上其他 Internet 感知 (Internet-aware) 程序使用。

地址的最后一部分 “/service” 是路径。每一个 Web 服务必须有一个唯一的路径。如果在你的计算机上运行其他 Web 服务, 则它们的路径不能与 `SquareRootServer` 相同。

为了部署 Web 服务, 在 `com.java24hours.ws` 包中创建一个名为 `SquareRootServerPublisher` 的 Java 空文件, 然后输入程序清单 22.3 中的所有文本。

程序清单 22.3 `SquareRootServerPublisher.java` 的完整源代码

```
1: package com.java24hours.ws;
2:
3: import javax.xml.ws.*;
4:
5: public class SquareRootServerPublisher {
6:     public static void main(String[] arguments) {
7:         SquareRootServerImpl srsi = new SquareRootServerImpl();
8:         Endpoint.publish(
9:             "http://127.0.0.1:5335/service",
10:            srsi
11:        );
12:    }
13: }
```

运行该程序时, 它将在计算机上的 5335 端口等待连接。只要程序支持 SOAP 或基于 REST 的 Web 服务, 则无论它是使用 Java 语言还是其他语言编写的, 都可以调用其中的 Web 服务的方法。只要你的 Web 服务位于 Internet 中, 任何连接到 Internet 的软件都可以调用其方法。

22.4 使用 Web 服务描述语言文件

在启用该 Web 服务之前, 你可以使用任意的 Web 浏览器来测试 `SquareRootServerPublisher` 应用程序的可用性。

打开浏览器, 然后输入地址 `http://127.0.0.1:5335/service?wsdl`。该浏览器将显示程序清单 22.4 中的 XML 文件。该文件由刚才创建的应用程序提供。

该文件是一个使用 Web 服务描述语言 (Web Service Description Language, WSDL) 编写的服务合约, WSDL 是一个 XML 方言, 它可以清楚地说明 Web 服务的运行方式, 以便服务器和客户端能够充分使用它。

在创建 JAX-WX 服务和客户端来访问 Web 服务时, 没有必要必须理解 WSDL。当然, 最好还是看一下该文件的内容, 以对基于 SOAP 和 REST 的 Web 服务的运行方式有一个理解。

程序清单 22.4 WSDL 合约

```
1: <?xml version="1.0" encoding="UTF-8"?>
2: <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version
```

```

3: is JAX-WS RI 2.2.2 in JDK 7. ->
4: <!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version
5: is JAX-WS RI 2.2.2 in JDK 7. ->
6: <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
7: xmlns:tns="http://ws.java24hours.com/"
8: xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9: xmlns="http://schemas.xmlsoap.org/wsdl/"
10: targetNamespace="http://ws.java24hours.com/"
11: name="SquareRootServerImplService">
12: <types></types>
13: <message name="getSquareRoot">
14: <part name="arg0" type="xsd:double"></part>
15: </message>
16: <message name="getSquareRootResponse">
17: <part name="return" type="xsd:double"></part>
18: </message>
19: <message name="getTime"></message>
20: <message name="getTimeResponse">
21: <part name="return" type="xsd:string"></part>
22: </message>
23: <portType name="SquareRootServer">
24: <operation name="getSquareRoot" parameterOrder="arg0">
25: <input message="tns:getSquareRoot"></input>
26: <output message="tns:getSquareRootResponse"></output>
27: </operation>
28: <operation name="getTime" parameterOrder="">
29: <input message="tns:getTime"></input>
30: <output message="tns:getTimeResponse"></output>
31: </operation>
32: </portType>
33: <binding name="SquareRootServerImplPortBinding"
34: type="tns:SquareRootServer">
35: <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
36: style="rpc"></soap:binding>
37: <operation name="getSquareRoot">
38: <soap:operation soapAction=""></soap:operation>
39: <input>
40: <soap:body use="literal"
41: namespace="http://ws.java24hours.com/"></soap:body>
42: </input>
43: <output>
44: <soap:body use="literal"
45: namespace="http://ws.java24hours.com/"></soap:body>
46: </output>
47: </operation>
48: <operation name="getTime">
49: <soap:operation soapAction=""></soap:operation>
50: <input>
51: <soap:body use="literal"
52: namespace="http://ws.java24hours.com/"></soap:body>
53: </input>
54: <output>
55: <soap:body use="literal"
56: namespace="http://ws.java24hours.com/"></soap:body>
57: </output>
58: </operation>
59: </binding>
60: <service name="SquareRootServerImplService">
61: <port name="SquareRootServerImplPort"
62: binding="tns:SquareRootServerImplPortBinding">
63: <soap:address location="http://127.0.0.1:5335/service"></soap:address>

```

```
64: </port>
65: </service>
66: </definitions>
```

WSDL 之所以称为服务合约，是因为它规定了访问 Web 服务的方式、可以与服务交换的信息，以及所传输信息的数据类型。

WSDL 合约的第 13~22 行定义了 Web 服务的方法、这些方法的参数，以及作为响应返回的数据。查看这些代码行，看是否能够找到 `getSquareRoot()` 方法所在的位置。该方法接受一个 `double` 型参数，并返回一个 `double` 型值。

合约中提到的数据类型不是 Java 的数据类型，它们可以用于任何支持 SOAP 的编程语言（没有专门针对 Java 的 Web 服务）。

By the Way

注意：

由于 WSDL 合约定义了 Web 服务的细节，因此可以使用它来自动完成 Web 服务编程的大部分过程。Java 开发工具包（JDK）包含命令行工具 `wsimport`，它可以将 WSDL 文件作为输入，然后用于访问 Web 服务的 Java 类。

22.5 创建 Web 服务客户端

这一节将创建 `SquareRootClient` 应用程序，它可以调用你前面创建的 Web 服务的方法。当然，该服务必须处于运行状态，以便客户端与其连接。

由于像 JAX-WS 库这样的 Web 服务技术支持 SOAP、REST、HTTP 和 XML 等标准，因此不一定非要使用 Java 程序来连接该 Web 服务。Perl、Python、Ruby 和其他语言都有支持 Web 服务的库。

JAX-WS 库在 `javax.xml.ws` 包中提供了 `Service` 类，这是一个可以创建调用 Web 服务的对象的工厂（factory）。

类方法 `Service.create(URL, QName)` 创建了工厂，其参数 `URL` 和 `QName` 分别来自 `java.net` 和 `java.xml.namespace`。

`URL` 必须是 Web 服务的 WSDL 合约的地址：

```
URL url = new URL("http://127.0.0.1:5335/service?wsdl");
```

Watch Out!

警告：

前面提到，URI 不一定非要是一个可用的网络地址，尽管 `http://ws.java24hours.com` 看起来很像一个可用的网络地址，但是它是作为唯一的标识符出现的。我拥有域名 `java24hours.com`，并可以控制其子域名的使用方式，因此我将 `http://ws.java24hours.com` 作为 URI 使用。我可以保证其他 Web 服务提供商不会使用该标识符。

QName 是一个限定名称 (qualified name), 这是一个与 Web 服务提供者关联起来的 XML 标识符。限定名称包含命名空间 URI 和本地的标识符。

命名空间 URI 和 URL 相似, 但是不一定非要作为网络地址来使用。由于平方根 Web 服务的包名称是 `com.java24hours.ws`, 在 Java 中, 按照惯例, 该名称与 Internet 主机名 `ws.java24hours.com` 关联起来, 用于该 Web 服务的命名空间 URI 是 `http://ws.java24hours.com`。

该 Web 服务的本地标识符是服务实现 Bean 的名字, 而且后面添加有 “Service” 单词。下面是一个创建限定名称的语句:

```
QName qname = new QName(
    "http://ws.java24hours.com/",
    "SquareRootServerImplService"
);
```

使用 URL 和限定名称则可以创建 Web 服务客户端工厂:

```
Service service = Service.create(url, qname);
```

该工厂有 `getPort(Class)` 方法, 这个方法创建了指定类的独享。为了对作为方法参数使用的 Java 类进行识别, 可以使用名为 `class` 的类变量。感到困惑了? 当在 Java 语句中看到它时, 就会明白了:

```
SquareRootServer srs = service.getPort(SquareRootServer.class);
```

使用 `SquareRootServer.class` 作为参数来调用 `getPort()` 方法时, 将导致工厂创建一个 `SquareRootServer` 对象。该对象存储在 `srs` 变量中。

可以在 Java 的其他对象中调用 `SquareRootServer` 对象的方法:

```
System.out.println(srs.getTime());
System.out.println(srs.getSquareRoot(625D));
```

JAX-WS 库将这些方法调用作为 SOAP 消息打包, 然后通过 Internet 发送到 Web 服务, 然后进行方法调用。

当服务响应这些调用时, 它将响应打包为 SOAP 消息, 然后通过 Internet 发送回去, 之后再被转换为 Java 数据类型。

创建一个名为 `SquareRootClient` 的 Java 文件, 然后输入程序清单 22.5 中的所有内容。

程序清单 22.5 SquareRootClient.java 的完整源代码

```
1: package com.java24hours.ws;
2:
3: import java.net.*;
4: import javax.xml.namespace.*;
5: import javax.xml.ws.*;
6:
7: class SquareRootClient {
8:     public static void main(String[] arguments) throws Exception {
9:         URL url = new URL("http://127.0.0.1:5335/service?wsdl");
10:        QName qname = new QName(
11:            "http://ws.java24hours.com/",
12:            "SquareRootServerImplService"
```



```

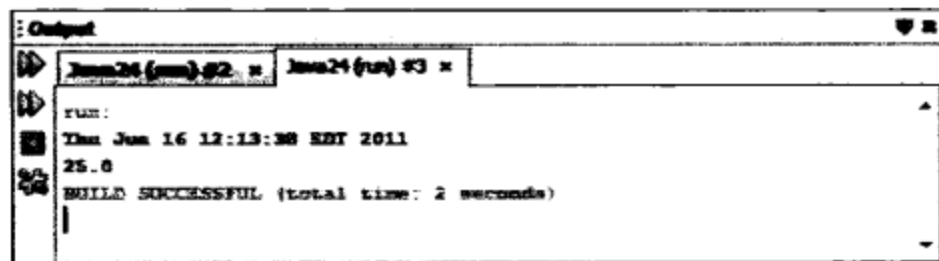
13:         );
14:         Service service = Service.create(url, qname);
15:         SquareRootServer srs = service.getPort(SquareRootServer.class);
16:
17:         System.out.println(srs.getTime());
18:         System.out.println(srs.getSquareRoot(625D));
19:     }
20: }

```

运行该客户端应用程序时，如果 SquareRootPublisher 应用程序也已经处于运行状态，则会看到图 22.2 中的输出。

图 22.2

调用 Web 服务并
显示响应



22.6 总结

在 Java 中，有一类 API 用作基于 XML 的 RPC (JAX-RPC)，JAX-PRC 是一种允许 Java 对象经由 Internet 向另外一个对象进行远程过程调用 (RPC) 的技术。而 JAX-WS 包和类的集合则继承了该类 API。

无论软件放在哪里，使用什么语言编写而成，都可以进行对其调用的能力在 Web 2.0 软件开发浪潮中具有重要的作用。

在 20 世纪 90 年代中期，网络开始流行之时，人们就已经享用到了 Internet 独一无二的连接优势，而 Web 2.0 则允许软件来采用这种连接优势。

本节讲解了使用 JAX-WS 来创建和使用 Web 服务的 4 个步骤：为服务创建一个接口（服务端点接口）、实现该服务（服务实现 Bean）、在 Internet 上发布服务，以及创建客户端访问该服务。

许多编程工具，包括 NetBeans 和 JDK，都可以自动创建代码，从而简化创建和访问 Web 服务的工作。

22.7 问与答

问：XML-RPC 是如何应用于 SOAP Web 服务中的？

答：XML-PRC 是一个协议，它可以通过 HTTP 来调用方法，并接收 XML 格式的数据。SOAP 也是处理这些事情。事实上，这两个 Web 协议具有同一个起源。

XML-PRC 诞生于一份协议的草案中，而该协议最终成为 SOAP。由于 XML-RPC 先于 SOAP 而生，其实现也早于 SOAP，因此它沿着自己的路进行发展，并且直到如今仍然广受欢迎。Apache XML-RPC Java 库（可以从 <http://ws.apache.org/xmlrpc> 下载），可以创建使用 XML-RPC 的 Web 服务和客户端。

而 SOAP 是一种更为复杂的 Web 服务协议，其支持的客户端/服务交互的范围更为广泛。

问：为什么 `com.java24hours.ws` 包与 Internet 主机 `ws.java24hours.com` 关联到一起？它们是如何关联的？

答：Java 包的名字是由开发该包的程序员创建的。Oracle 在给 Java 类库中的 Java 包命名时，使用 `java` 或 `javax` 打头，比如 `java.util` 和 `javax.swing`。当其他程序员创建包时，他们遵循了一个惯例，以防止两个实体选择相同的包名称，以及防止两个实体相互混淆。

该惯例是，基于实体所拥有的域名来选择包的名字。由于我是域名 `cadenhead.org` 的拥有者，因此在创建 Java 类时，其包名可以以 `org.cadenhead` 开头，比如 `org.cadenhead.web`。Apache 软件基金会（Apache Software Foundation）拥有 `apache.org` 域名，因此可以将它的 XML-RPC 包命名为 `org.apache.xmlrpc`。

22.8 测验

22.8.1 问题

1. 服务实现 Bean 是什么？
 - a. 一个接口，可以识别通过 Web 服务进行访问的方法。
 - b. 实现了 Web 服务的类。
 - c. 位于 Web 服务和调用该服务的客户端之间的服务合约。
2. 当像 `@WebMethod` 或 `@Override` 这样的文本出现在方法声明中时，将调用什么？
 - a. 注解。
 - b. 断言。
 - c. 恼怒。
3. WSDL 代表什么？
 - a. Web 服务部署语言（Web Services Deployment Language）。
 - b. Web 服务描述语言（Web Services Description Language）。
 - c. Lucy in the Sky with Diamonds。

22.8.2 答案

1. b. 答案 a 指的是服务端点接口。
2. a. 尽管我觉得答案 c 也可能是真的，但是这取决于在这一节遇到了多大的麻烦。
3. b. WSDL 通常被错误地成为 Web 服务定义语言（Web Services Definition Language）。

22.9 练习

为了进一步巩固本节所学的知识，请做如下练习：

- 在平方根 Web 服务程序中添加一个方法，该方法将一个数乘以 10，然后修改 `SquareRootClient` 应用程序来调用该方法。
- 创建一个新的 Web 服务，它使用第 21 章中的 `WeatherStation` 类，而且通过该 Web 服务可以访问当前的温度（高温、低温）、天气情况。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 www.java24hours.com。



第 23 章

创建 Java2D 图形

本章介绍如下内容：

- 设置文本的字体和颜色；
- 设置容器的背景色；
- 绘制直线、矩形和其他图形；
- 绘制 GIF 和 JPEG 图形；
- 绘制填充形状和非填充形状。

在本章，读者将学到如何将 Swing 容器（存放 GUI 组件的纯黑色面板和框架）变成一块可以在上面绘制字体、颜色、形状和图形的艺术画布。

23.1 使用 Font 类

在 Java 中，颜色和字体使用 `java.awt` 包中的 `Color` 和 `Font` 类来表示。通过使用这些类，你可以用不同的字体和字号来显示文本，也可以改变字体和图形的颜色。字体使用构造函数 `Font(String, int, int)` 来创建，该构造函数接受 3 个参数：

- 字体的名字，它可以是描述性名字（比如 `Dialog`、`DialogInput`、“`Monospaced`”、“`San Serif`”或“`Serif`”），也可以是实际的字体名（比如“`Arial Black`”、“`Helvetica`”或“`Courier New`”）；
- `Font.BOLD`、`Font.ITALIC` 和 `Font.PLAIN` 这 3 个类变量中的其中一个；
- 字体的大小（字号），单位为磅。

下面的语句创建了一个 `Font` 对象，其字号为 12 磅，字体为 `Serif`，样式为斜体：

```
Font current = new Font("Serif", Font.ITALIC, 12);
```

如果你使用了一个指定的字体，而不是通用的字体，则该字体必须已经安装在运行程序

的计算机中。你可以将字体样式合并起来，如下面的例子所示：

```
Font headline = new Font("Courier New", Font.BOLD + Font.ITALIC, 72);
```

当有了字体之后，可以调用 Graphics2D 组件的 setFont(Font)方法，将其指定为当前字体。在没有指定其他新的字体之前，后续所有的绘制操作都将使用该字体。下面例子中的语句创建了“Comic Sans”字体对象，并在绘制文本之前将其指定为当前地体：

```
public void paintComponent(Graphics comp) {
    Graphics2D comp2D = (Graphics2D) comp;
    Font font = new Font("Comic Sans", Font.BOLD, 15);
    comp2D.setFont(font);
    comp2D.drawString("Potrzebie!", 5, 50);
}
```

Java 支持消除锯齿功能，因此可以更为平滑地绘制字体和图形，而且它们的外观具有较少的锯齿。为了启用这个功能，必须在 Swing 中设置渲染提示（rendering hint）。Graphics2D 对象具有一个 setRenderingHint(int, int)方法，它可以接受两个参数：

- 渲染提示键（key）；
- 与该键相关联的值。

这些值是位于 java.awt 包中的 RenderingHints 类的类变量。要启用消除锯齿功能，可使用两个参数来调用 setRenderingHint()方法：

```
comp2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);
```

在该例子中，comp2D 对象是 Graphics2D 对象，表示一个容器的绘制环境。

23.2 使用 Color 类

在 Java 中，颜色可使用 Color 类表示，其中包括如下常量：black、blue、cyan、darkGray、gray、green、lightGray、magenta、orange、pink、red、white 和 yellow。

在容器中，可以调用 setBackground(Color)方法来设置组件的背景色，其中上面提到的颜色常量作为该方法的参数，如下例所示：

```
setBackground(Color.orange);
```

与当前的字体相同，在使用 setColor(Color)方法执行绘制任务之前，必先设置当前的颜色。

在下面的代码中，一条语句将当前颜色设置为 orange，并使用该颜色来绘制文本：

```
public void paintComponent(Graphics comp) {
    Graphics2D comp2D = (Graphics2D) comp;
    comp2D.setColor(Color.orange);
    comp2D.drawString("Go, Buccaneers!", 5, 50);
}
```

在使用 setBackground()方法时，我们可以直接在容器上调用，但是 setColor()则不行，

我们必须在 Graphics2D 对象上调用该方法。

23.3 创建自定义颜色

在 Java 中，通过指定 sRGB 通用色彩标准的值，可以创建自定义颜色。sRGB 使用颜色中的红、绿、蓝分量来定义颜色。每一种颜色的取值在 0~255 之间（0 表示没有这种颜色，255 表示该颜色分量具有最大值）。

构造函数 Color(int, int, int) 接口 3 个参数，这 3 个参数分别表示红、绿、蓝的值。下面的代码将绘制一个面板，该面板在暗红色（红：235；绿：50；蓝：50）背景中显示亮橙色（红：230；绿：220；蓝：0）文本：

```
import java.awt.*;
import javax.swing.*;

public class GoBucs extends JPanel {
    Color lightOrange = new Color(230, 220, 0);
    Color darkRed = new Color(235, 50, 50);

    public void paintComponent(Graphics comp) {
        Graphics2D comp2D = (Graphics2D) comp;
        comp2D.setColor(darkRed);
        comp2D.fillRect(0, 0, 200, 100);
        comp2D.setColor(lightOrange);
        comp2D.drawString("Go, Buccaneers!", 5, 50);
    }
}
```

该示例调用 Graphics2D 的 fillRect() 方法使用当前颜色绘制一个填充的矩形。

注意：

通过使用 sRGB 值，可以创建 1650 万种颜色，不过大多数计算机显示器只能近似地显示其中的大部分颜色。

By the Way

23.4 绘制直线和形状

在 Java 程序中，绘制直线和矩形这样的形状就像显示文本那样简单。你只需要一个 Graphics2D 对象来定义绘制平面和表示要绘制内容的对象即可。

Graphics2D 对象有用来绘制文本的方法，如下所示：

```
comp2D.drawString("Draw, pardner!", 15, 40);
```

这将在坐标点(15,40)位置绘制文本“Draw, Pardner!”。绘制直线的方法所使用的坐标系与绘制文本的方法使用的相同。(0, 0)坐标位于容器的左上角，当向右移动时，x 值增加，当向下移动时，y 值增大。通过使用下面的语句，可以确定 applet 能够使用的最大(x, y)值：

```
int maxXValue = getSize().width;
int maxYValue = getSize().height;
```

除了可以绘制直线以外，你还可以绘制填充的形状或者是未填充的形状。所谓填充形状就是绘制该形状时，使用当前颜色将该形状空间完全填充起来。而非填充的形状则是只使用当前的颜色绘制的形状的边界。

23.4.1 绘制直线

在创建一个对象的 2D 图形时，它表示的是正在绘制的形状。

定义形状的类型位于 `java.awt.geom` 包中。

`Line2D.Float` 类能够创建一条连接起点(x, y)和终点(x,y)的直线。下面的语句将创建一条起点为(40,200)，终点为(70,130)的直线：

```
Line2D.Float line = new Line2D.Float(40F, 200F, 70F, 130F);
```

By the Way

注意：

`Line2D.Float` 类与之前用到的大多数类不同，它的类名中间有一个句点。这是因为 `Float` 是 `Line2D` 类中的内部类。有关 `Float` 类的内容，请见第 11 章。

在上面的语句中，参数后面跟有字母 F，用来指示参数是浮点型数值。如果省略该字母，则 Java 会把参数当作整型数值。

除了直线之外，调用 `Graphics2D` 类的方法还可以绘制其他形状：`draw()` 方法可以绘制轮廓线，而 `fill()` 方法可以绘制填充形状。

将下面的语句与上面的代码相结合，可以绘制出直线对象：

```
comp2D.draw(line);
```

23.4.2 绘制矩形

矩形可以是填充或非填充的，还可以是圆角的或直角的。矩形可以使用构造函数 `Rectangle2D.Float(int, int, int, int)` 来创建，创建时指定如下 4 个参数：

- 矩形左上角的 x 坐标。
- 左上角的 y 坐标。
- 矩形的宽度。
- 矩形的高度。

下面的语句可以绘制一个非填充的直角矩形：

```
Rectangle2D.Float box = new  
    Rectangle2D.Float(245F, 65F, 20F, 10F);
```

这条语句创建的矩形的左上角坐标为(245, 65)，宽度为 20，高度为 10，宽度和高度都以像素为单位。要绘制该矩形的轮廓，可使用下面的语句：

```
comp2D.draw(box);
```


要填充该矩形，可以使用 `fill()` 方法：

```
comp.fill(box);
```

可以使用 `RoundRectangle2D.Float` 类来创建圆角矩形。

这个类的构造函数的前 4 个参数与 `Rectangle2D.Float` 类相同，并增加了下面 2 个参数：

- 水平方向上离矩形角的像素数；
- 垂直方向上离矩形角的像素数。

这些距离用于指定矩形圆角的起始位置。

下面的语句创建一个圆角矩形：

```
RoundRectangle2D.Float ro = new RoundRectangle.Float(
    10F, 10F,
    100F, 80F,
    15F, 15F);
```

该矩形的左上角坐标为(10, 10)。第 3 个和第 4 个参数指定矩形的宽度和高度。在本例中，矩形的宽度给 100 像素，高度为 80 像素。

最后两个参数指定在矩形的 4 个角上，在离角点 15 像素处开始倒圆角。

23.4.3 绘制椭圆和圆

椭圆和圆是使用同一个类——`Ellipse2D.Float` 创建的。这个类的构造函数接受 4 个参数：

- 椭圆的 x 坐标；
- 椭圆的 y 坐标；
- 椭圆的宽度；
- 椭圆的高度。

读者可能已经猜到，(x, y) 坐标不是椭圆或圆的圆心。相反，(x, y) 坐标、宽度、高度描述了椭圆的外接矩形，(x, y) 是该矩形的左上角坐标。如果宽度和高度相同，椭圆将变成圆。

下面的语句创建一个圆，其外接矩形的左上角坐标为 (245, 45)，宽度和高度都是 5 像素：

```
Ellipse2D.Float cir = new Ellipse2D.Float(
    245F, 45F, 5F, 5F);
```

23.4.4 绘制弧线

在 Java 中可绘制的另一个圆形形状是弧线，它是椭圆或圆的一部分。弧线用 `Arc2D.Float` 类创建，这个类的构造函数使用几个与 `Ellipse2D.Float` 相同的参数。要创建弧线，需要指定一个椭圆、该椭圆的可见部分（单位为度）以及弧线的起点。

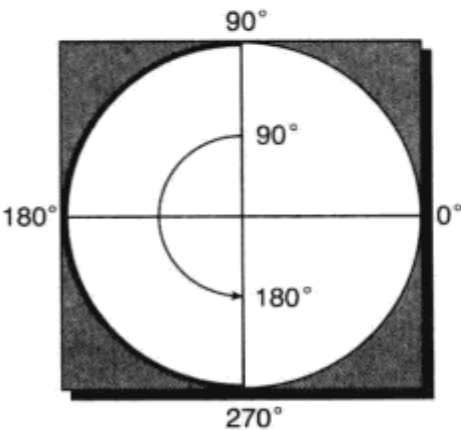
要创建弧线，向构造函数传递下述整型参数：

- 椭圆外接矩形左上角的 x 坐标；
- 该矩形左上角的 y 坐标；

- 该矩形的宽度;
- 该矩形的高度;
- 弧线的起点 (0° ~ 359°);
- 弧线长度, 单位为度;
- 弧线类型。

弧线的起点为 0° ~ 359°, 方向为逆时针方向, 其中 0° 对应于 3 点钟的位置, 如图 23.1 所示。

图 23.1
如何使用度数来定
义弧线



弧线的类型用 Arc2D.Float 类的 3 个类变量之中的一个指定: PIE 将弧线绘制为饼图的一部分; CLOSED 将弧线的起点和终点用直线相连; OPEN 不将终点和起点相连。

下面的语句创建一个非闭合弧线, 其外接矩形的左上角坐标为 (100, 50), 高度和宽度值都是 75, 弧线从 30° 处开始, 长 120°:

```
Arc2D.Float smile = new Arc2D.Float(100F, 50F, 65F, 75F,
    30F, 120F, Arc2D.Float.OPEN);
```

23.5 绘制饼图

在结束本章前, 将创建图形用户界面组件 PiePanel, 它显示一个饼图。该组件是 JPanel 的子类, JPanel 是一个简单的 Swing 容器, 非常适合用于在其中绘制内容。

开始创建类之前, 应定义创建其对象的方式。使用 PiePanel 类的应用程序必须执行下面的步骤:

- 使用构造函数 PiePanel (int) 创建一个 PiePanel 对象, 其中的整型参数指定饼图包含的切片数;
- 调用对象的 addslice (color, float) 方法给切片指定颜色 and 值。

PiePanel 中的每个切片的值为其表示的数量。

例如, 表 23.1 列出了在头 38 年中, 美国学生还贷数据, 这些数据是高等教育办公室提供的。

表 23.1 美国学生还贷统计表

已归还的贷款总额	1010 亿美元
在校生贷款总额	680 亿美元
正在归还的贷款总额	910 亿美元
拖欠贷款总额	250 亿美元

可以使用 `PiePanel` 在饼图中表示这些数据，为此可使用下述代码：

```
PiePanel loans = new PiePanel(4);
loans.addSlice(Color.green, 101F);
loans.addSlice(Color.yellow, 68F);
loans.addSlice(Color.blue, 91F);
loans.addSlice(Color.red, 25F);
```

图 23.2 所示为一个应用程序运行时显示出的结果，该应用程序包含了一个使用学生贷款数据创建的 `PiePanel` 组件。

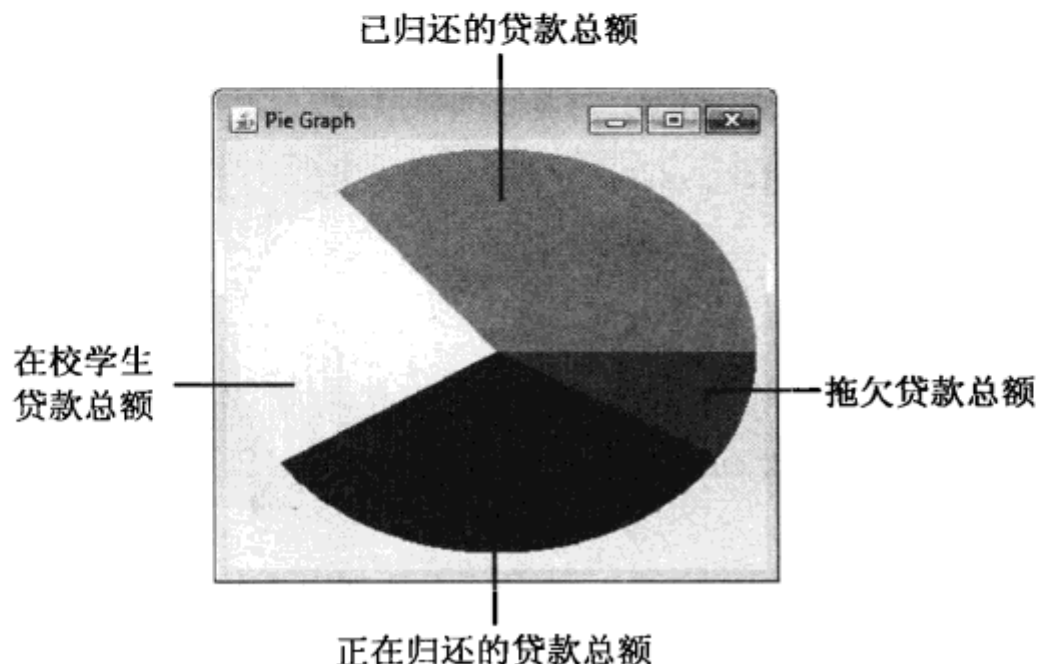


图 23.2

在饼图中显示学生贷款数据

在创建 `PiePanel` 对象时，构造函数中指定了切片数。要绘制每个切片，还需要知道其他 3 项内容：

- 切片颜色，用 `Color` 对象表示；
- 每个切片代表的值；
- 所有切片代表的总值。

使用助手类 `PieSlice` 来表示饼图中的每个切片：

```
import java.awt.*;

class PieSlice {
    Color color = Color.lightGray;
    float size = 0;

    PieSlice(Color pColor, float pSize) {
        color = pColor;
        size = pSize;
    }
}
```

每个切片都是通过调用 `PieSlice(Color, float)` 创建的。所有切片的总值将存储在 `PiePanel` 类的私有实例变量 `totalSize` 中，这个类中还有实例变量 `background` 和 `current`，它们分别用于存储面板背景色和对切片进行计数：

```
private int current = 0;
private float totalSize = 0;
private Color background;
```

有了 `PieSlice` 类后，就可以使用另一个实例变量创建一个 `PieSlice` 对象数组：

```
private PieSlice[] slice;
```

创建 **PiePanel** 对象时，没有指定切片的颜色和大小。在构造函数中只需指定 **slice** 数组的长度并保存面板的背景色：

```
public PiePanel(int sliceCount) {
    slice = new PieSlice[sliceCount];
    background = getBackground();
}
```

addSlice(Color, float)方法用于在面板中添加饼图切片：

```
public void addSlice(Color sColor, float sSize) {
    if (current <= slice.length) {
        slice[current] = new PieSlice(sColor, sSize);
        totalSize += sSize;
        current++;
    }
}
```

current 实例变量用于将每个切片存储到 **slice** 数组的相应元素中。数组的 **length** 变量指出了数组被定义为存储多少个元素，因此只要 **current** 不大于 **slice.length**，就可以继续往面板中添加切片。

正如读者预期的，**PiePanel** 类在其 **paintComponent()**方法中处理所有的图形操作。该任务中最棘手的是绘制代表每个饼图切片的弧线。

这是由下面的语句处理的：

```
float start = 0;
for (int i = 0; i < slice.length; i++) {
    float extent = slice[i].size * 360F / totalSize;
    comp2D.setColor(slice[i].color);
    Arc2D.Float drawSlice = new Arc2D.Float(
        xInset, yInset, width, height, start, extent,
        Arc2D.Float.PIE);
    start += extent;
    comp2D.fill(drawSlice);
}
```

变量 **start** 用于记录弧线的起点，变量 **extent** 用于记录弧线的长度。如果知道所有饼图切片的总值和每个切片的值，便可以将切片的值乘以 360 再除以所有切片的总值，从而计算得到 **extent** 的值。

所有弧线都是在一个 **for** 循环中绘制的：计算出每个弧线的 **extent**，创建该弧线并将 **start** 增加 **extent**。这确保下一个切片紧邻最后一个切片。最后，调用 **Graphics2D** 的 **fill()**方法绘制弧线。

要将上述内容组合在一起，创建一个名为 **PiePanel** 的 Java 空文件，然后输入程序清单 23.1 所示的内容。

程序清单 23.1 PiePanel.java 的完整源代码

```
1: import java.awt.*;
2: import javax.swing.*;
3: import java.awt.geom.*;
```

```

4:
5: public class PiePanel extends JPanel {
6:     private PieSlice[] slice;
7:     private int current = 0;
8:     private float totalSize = 0;
9:     private Color background;
10:
11:     public PiePanel(int sliceCount) {
12:         slice = new PieSlice[sliceCount];
13:         background = getBackground();
14:     }
15:
16:     public void addSlice(Color sColor, float sSize) {
17:         if (current <= slice.length) {
18:             slice[current] = new PieSlice(sColor, sSize);
19:             totalSize += sSize;
20:             current++;
21:         }
22:     }
23:
24:     public void paintComponent(Graphics comp) {
25:         super.paintComponent(comp);
26:         Graphics2D comp2D = (Graphics2D) comp;
27:         int width = getSize().width - 10;
28:         int height = getSize().height - 15;
29:         int xInset = 5;
30:         int yInset = 5;
31:         if (width < 5) {
32:             xInset = width;
33:         }
34:         if (height < 5) {
35:             yInset = height;
36:         }
37:         comp2D.setColor(background);
38:         comp2D.fillRect(0, 0, getSize().width, getSize().height);
39:         comp2D.setColor(Color.lightGray);
40:         Ellipse2D.Float pie = new Ellipse2D.Float(
41:             xInset, yInset, width, height);
42:         comp2D.fill(pie);
43:         float start = 0;
44:         for (int i = 0; i < slice.length; i++) {
45:             float extent = slice[i].size * 360F / totalSize;
46:             comp2D.setColor(slice[i].color);
47:             Arc2D.Float drawSlice = new Arc2D.Float(
48:                 xInset, yInset, width, height, start, extent,
49:                 Arc2D.Float.PIE);
50:             start += extent;
51:             comp2D.fill(drawSlice);
52:         }
53:     }
54: }
55:
56: class PieSlice {
57:     Color color = Color.lightGray;
58:     float size = 0;
59:
60:     PieSlice(Color pColor, float pSize) {
61:         color = pColor;
62:         size = pSize;
63:     }
64: }

```

在程序清单 23.1 中，第 1~54 行定义了 PiePanel 类，在第 56~64 定义了 PieSlice 助手类。PiePanel 类可以在任何 Java 程序的 GUI 中用作组件。要测试 PiePanel，需要创建一个类，并在其中使用该 PiePanel。

程序清单 23.2 是一个使用 PiePanel 类的应用程序——PieFrame。创建一个 Java 空文件，然后输入程序清单 23.2 中的所有文本。

程序清单 23.2 PieFrame.java 的完整源代码

```
1: import javax.swing.*;
2: import javax.swing.event.*;
3: import java.awt.*;
4:
5: public class PieFrame extends JFrame {
6:     Color uneasyBeingGreen = new Color(0xCC, 0xCC, 0x99);
7:     Color zuzusPetals = new Color(0xCC, 0x66, 0xFF);
8:     Color zootSuit = new Color(0x66, 0x66, 0x99);
9:     Color sweetHomeAvocado = new Color(0x66, 0x99, 0x66);
10:    Color shrinkingViolet = new Color(0x66, 0x66, 0x99);
11:    Color miamiNice = new Color(0x33, 0xFF, 0xFF);
12:    Color inBetweenGreen = new Color(0x00, 0x99, 0x66);
13:    Color norwegianBlue = new Color(0x33, 0xCC, 0xCC);
14:    Color purpleRain = new Color(0x66, 0x33, 0x99);
15:    Color freckle = new Color(0x99, 0x66, 0x33);
16:
17:    public PieFrame() {
18:        super("Pie Graph");
19:        setLookAndFeel();
20:        setSize(320, 290);
21:        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22:        setVisible(true);
23:
24:        PiePanel pie = new PiePanel(10);
25:        pie.addSlice(uneasyBeingGreen, 1337);
26:        pie.addSlice(zuzusPetals, 1189);
27:        pie.addSlice(zootSuit, 311);
28:        pie.addSlice(sweetHomeAvocado, 246);
29:        pie.addSlice(shrinkingViolet, 203);
30:        pie.addSlice(miamiNice, 187);
31:        pie.addSlice(inBetweenGreen, 166);
32:        pie.addSlice(norwegianBlue, 159);
33:        pie.addSlice(purpleRain, 139);
34:        pie.addSlice(freckle, 127);
35:        add(pie);
36:    }
37:
38:    private void setLookAndFeel() {
39:        try {
40:            UIManager.setLookAndFeel(
41:                "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel"
42:            );
43:        } catch (Exception exc) {
44:            // ignore error
45:        }
46:    }
47:
48:    public static void main(String[] arguments) {
49:        PieFrame pf = new PieFrame();
50:    }
51: }
```

`PieFrame` 类是一个简单的图形用户界面，包含一个组件，该组件是在第 24 行创建的 `PiePanel` 对象。在第 25~35 行，该对象的 `addSlice()` 方法被调用了 10 次，以将切片添加到饼图中。

运行该应用程序时，`PieFrame` 将显示一个饼图，其中包含 10 个人口最多的国家的人口数量（单位为百万），这些数据来自美国人口普查局于 2011 年 7 月发表的国际数据报告。按从多到少的顺序依次为：中国（13.37 亿）、印度（11.89 亿）、美国（3.11 亿）、印尼（2.46 亿）、巴西（2.03 亿）、巴基斯坦（1.87 亿）、尼日利亚（1.66 亿）、孟加拉国（1.59 亿）、俄罗斯（1.39 亿）和日本（1.27 亿）。

由于 Java 在 `Color` 类中只定义了几种颜色，因此这里创建了 10 种新颜色，并赋予它们描述性名称。这些颜色是用十六进制值表示的（在 Java 中，十六进制值以 `0x` 打头），但也可以在 `Color()` 构造函数使用十进制值来指定它们。

图 23.3 所示为运行该应用程序时的结果。

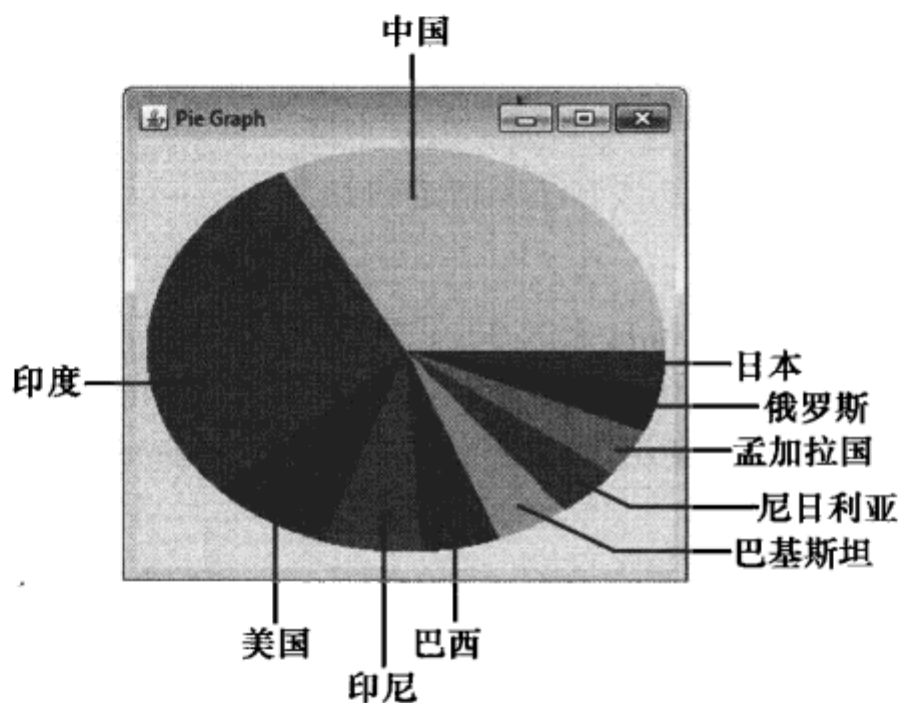


图 23.3

在饼图中显示人口数据

注意：

通过访问 www.cadenhead.org/census，可以找到美国人口普查局发布的当前国际人口数据。

By the Way

23.6 总结

通过使用字体、颜色和图形，可以让程序中的元素更具备吸引力，从而引起用户的注意。

尽管使用 Java 中的形状在进行绘制时，看起来相当麻烦，而且不是那么值得，但是，与从图像文件中载入的图形相比，使用多边形描绘的图形具有两个优势：

- 速度：即使是加载和显示小图形（如图标），所需的时间也比加载和显示一系列多边形长。

- 缩放：对于由多边形组成的图像，只需修改用于创建多边形的值就能改变整个图像的大小。例如，可在 `Sign` 类中添加一个函数，将每个形状的(x, y)点乘以 2 再创建它们，这样图像将比原来大一倍。缩放多边形图像的速度比图像文件快，且得到的结果更好。

23.7 问与答

问：如何沿顺时针方向（而不是逆时针方向）绘制弧线？

答：可以将弧线的长度设置为负数。弧线起点不变，但方向相反。例如，下面的语句绘制一条这样的非闭合弧线，其外接矩形的左上角坐标为 (35, 20)，高度为 20，宽度为 15，弧线的长度为 90°，起点为 0°，方向顺时针方向：

```
Arc2D.Float smile = new Arc2D.Float(35F, 20F, 15F, 20F,
    0F, -90F, Arc2D.Float.OPEN);
```

问：椭圆和圆没有角。在构造函数 `Ellipses.Float` 中指定的 (x, y) 坐标是什么？

答：(x, y) 坐标是椭圆和圆中最小 x 坐标和最小值 y 坐标，如果绘制一个外接椭圆或圆的矩形，该矩形左上角的 (x, y) 坐标就是该方法中指定的参数。

问：如何在 Java 中使用 XRender？

答：在基于 X11 的环境中（通常是 Linux 上），Java 7 支持使用 XRender 渲染引擎来绘制 Java2D 图形。该功能在默认情况下是关闭的，必须使用命令行选项 `Dsun.java2d.xrender=true` 来启用该功能。XRender 允许 Java 程序使用现代图形处理单元（Graphics Processing Unit, GPU）的性能来绘制图形。

在 NetBeans 中，通过选择 `Run->Set Project Configuration->Customize` 可以设置该选项。使用 `VM Options` 字段来设置该选项，然后单击 `OK` 即可。

23.8 测验

回答下面的问题，看是否掌握了字体和颜色的使用技巧。

23.8.1 问题

1. 下面哪一个不是用来选择颜色的常量？
 - a. `color.cyan`。
 - b. `Color.teal`。
 - c. `Color.magenta`。
2. 当对颜色进行更改并在容器中进行重绘时，必须怎么做才能使其可见？
 - a. 使用 `drawColor()` 方法。

- b. 使用 `repaint()` 语句。
 - c. 什么都不做。
3. RGB 表示什么意思？
- a. Roy G Biv。
 - b. Red Green Blue。
 - c. Lucy in the Sky with Diamonds。

23.8.2 答案

- 1. b. Jacksonville Jaguars 的原色 teal 已经从 Color 类中移除。
- 2. b. 调用 `repaint()` 方法时，必须手动调用 `paintComponent()` 方法。
- 3. b. 如果 c 选项是正确答案，则只有在若干年后回忆时，才会“看到”（钻石）的颜色。

23.9 练习

为了进一步巩固你在编程中使用字体和颜色的技巧，请完成下面的练习：

- 创建 `PieFrame` 类的一个新版本，其中，颜色值和饼图切片的价值不再出现在应用程序的源代码中，而是作为命令行参数传入。
- 创建一个应用程序，它使用颜色、形状和字体在面板中绘制一个“停止”标记。

有关为完成这些练习而编写的 Java 程序，请访问本书的配套网站 [//www.java24hours.com](http://www.java24hours.com)。



第 24 章

编写 Android app

本章介绍如下内容：

- 为什么要创建 Android；
- 如何创建 Android app；
- Android app 是如何架构的；
- 如何在模拟器上运行 app；
- 如何在 Android 手机上运行 app。

Java 是一款通用的编程语言，可以在多种平台上运行。其中有一个平台在过去的 4 年间发展势头迅猛，成为使用 Java 语言进行开发的一个全新领域。

Android 最初作为在手机上使用的操作系统，如今已经应用到其他设备中，并专门运行使用 Java 编写的程序。

这些称为 app 的程序是在开源的移动平台上开发的，该平台对开发者来说完全免费。任何人都可以编写、部署和销售 Android app。

在本章，你将学习到 Android 是如何出现的，它为什么这么特殊，以及为什么数以万计的程序员在该平台上进行开发。通过本章的学习，你还可以创建一个 app，并将其运行在 Android 手机上。如果没有 Android 手机，则可以将其运行在模拟器上。

24.1 Android 简介

Google 在 2005 年收购了 Android 技术，并于两年后将其推出，旨在通过业界的努力来建立一种全新而且开放的移动手机平台，该平台没有专利保护，因此与 RIM 黑莓和苹果 iPhone 所使用的技术不同。一些著名的移动手机生产厂商和技术厂商，包括 Google、Intel、Motorola、Nvidia、Samsung 以及其他公司在内，组成了开放手机联盟（Open Handset Alliance），旨在本着互惠互利的目的，来推动这个新平台的发展。

Google 发布了 Android 软件开发工具包 (SDK)，这是一个免费的工具集，用以开发 Android app。运行 Android 的第一款手机是 T-Mobile G1，它于 2008 年 6 月上市。

Android 技术在发展初期进展缓慢，但是从 2010 年初期开始呈爆发态势增长，并成为 iPhone 和其他移动平台的强劲对手。所有的主流手机运行商现在都提供 Android 手机，而且 Android 在平板电脑和电子书阅读器市场也得到了显著增长。

在 Android 出现之前，如果要进行移动应用程序的开发，需要昂贵的编程工具和开发计划。而且手机生产厂商可以决定谁能够为其生产的手机开发 app，还可以决定 app 是否可以卖给其他用户。

Android 的出现打破了这一壁垒。

Android 的开源和非专利属性意味着任何人都可以开发、发布和销售 app。而且其中涉及的唯一成本是将 app 提交给 Google app 市场时收取的象征性费用，其他的都是免费的。

在 Android 开发者站点 (<http://developer.android.com>) 上可以下载 Android SDK，以及其他编程工具。当你开发 app 时，将会经常求助于该站点，因为它囊括了 Android Java 类库中的所有类，因此可以作为详尽的在线资源进行参考。

如果你使用的 IDE 支持 Android SDK，则可以很容易地开发 Android app。最流行的 Android 编程 IDE 是 Eclipse，它也是免费而且开源的。Eclipse 的一个 Android 插件可以使得 Android SDK 功能无缝地嵌入到 Eclipse 中。

你可以使用 Eclipse 来编写 Android app，并在模拟器（其运行行为很像 Android 手机的软件）上进行测试，甚至也可以将 app 部署在真实的 Android 设备上。

在过去，Java 语言主要用来编写运行在 3 个场合的程序，这 3 个场合是桌面计算机、Web 服务器和 Web 浏览器。

Android 可以让 Java 用在各处。你用 Java 编写的程序可以部署在上百万台手机和其他运动设备上。

在 20 世纪 90 年代中期，当 James Gosling 在 Sun 公司工作时，他发明 Java 语言的初衷就是希望该语言能够运行在各种设备上，比如手机、智能卡和家电中。如今，Android 的出现实现了 Java 语言的设计初衷。

当 Java 语言先是作为运行交互式 Web 程序的一种方式，继而成为通用编程语言，而逐渐流行起来时，Java 开发人员已经将 Java 的设计初衷抛到一边。

15 年过后，据业内人士估计，Android 平台承载了全世界十几亿的 Java 程序。

随着时间的发展，Android 将成为最普遍、最具潜力的 Java 编程领域，而且一定会是硕果累累。

这可能最有趣。

警告：

本章内容在该书中占据的篇幅最长，因为当你作为 Android app 开发人员起步，朝着将来的百万富翁努力时，需要知晓很多内容。

**Watch
Out!**

24.2 创建 Android app

Android app 是最常见的 Java 程序，它使用了应用程序的框架，该框架是所有 app 共有的一组核心类和文件。为了让 app 在 Android 设备上正确运行，该框架包含了一组 app 架构规则。

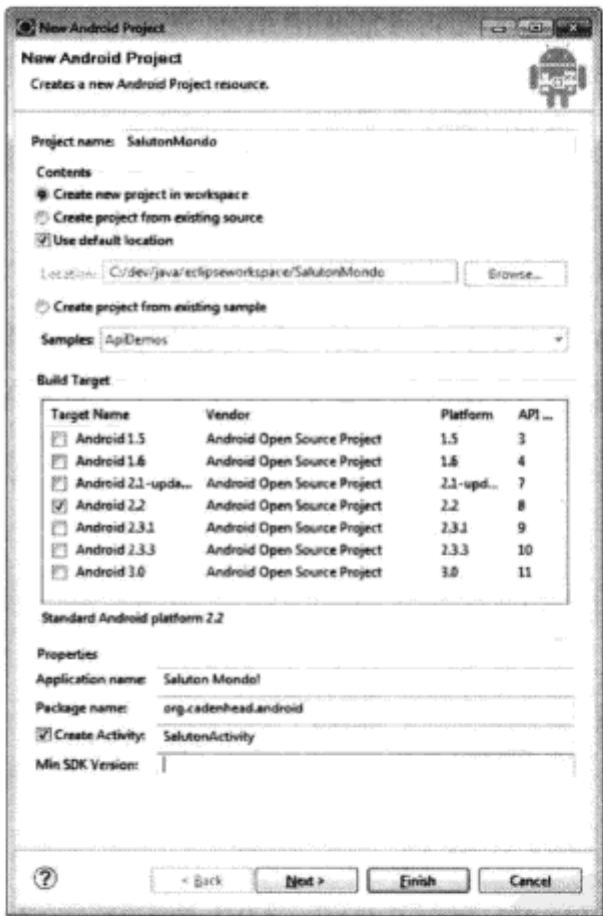
要开始编写 app，你必须安装并配置 Android SDK、Eclipse IDE，以及用于 Eclipse 的 Android 插件。

如果你是头一次进行 Android 编程，你可以在附录 D 找到获取并安装这些工具的方法。

将要创建的第一个项目是编写一个 SalutonMondo app，该程序可以在 Android 设备的屏幕上显示一行文本。

- 1. 运行 Eclipse IDE，它的外观和行为与 NetBeans 很像。
- 2. 选择 File->New->Android Project，打开 New Android Project 向导，如图 24.1 所示。

图 24.1
在 Eclipse 中创建
一个新的 Android
项目



- 3. 在 Project name 文本框中输入 SalutonMondo。
- 4. 在 Contents 区域，选中 Create New Project in Workspace。
- 5. Use Default Location 复选框确定该项目的存储位置。如果可以接受默认值，保持该复选框选中状态。否则取消选中该复选框，单击 Browse 按钮，然后选择项目要存储的文件夹。
- 6. 每一个 Android 项目需要一个构建目标（build target）。该目标表示可以运行 app 的 Android 最老版本。由于以一个新发布的 Android 版本都有增强的特性，你选择的目标决定了可以使用的特性。

对于像本例这样简单的 app，可以使用早期的目标，这里选择 Android 2.2。

7. 在 Application name 文本框，将该 app 命名为 Saluton Mondo!。该名字将显示在 Android 设备上。
8. Package name 文本框应该包含 app 类所属的 Java 包名。输入 org.cadenhead.android。
9. Create Activity 复选框指示这个新的 app 是否是使用 Activity 类创建的。行为(activity)是 app 可以完成的任务。将该复选框保持为选中状态，然后在旁边的文本框中输入 SalutonActivity。
10. 单击 Finish。该 app 创建完毕，SalutonMondo 条目出现在 Package Explorer 面板中。

24.2.1 剖析一个 Android 新项目

在一个 Android app 项目中，大约包含 20 个文件和文件夹，而且它们的组织方式相同。取决于 app 的功能，它可以包含更多的文件，但是这些开始文件和文件夹必须存在。

图 24.2 所示为在创建一个新的 Android 项目后，Eclipse Package Explorer 的界面。

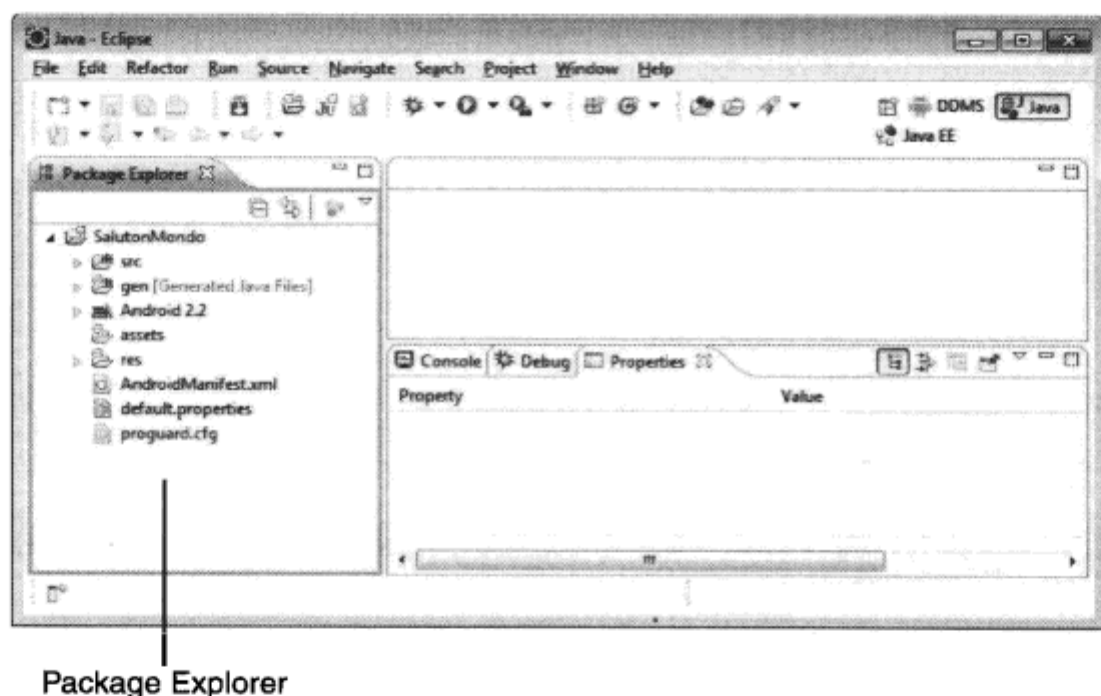


图 24.2

查看 Android 项目的组成部分

你可以使用文件夹来探索该项目的文件和文件夹结构。这个新的 SalutonMondo app 在开始时具有如下组件。

- /src 文件夹：该 app 的 Java 源代码的根文件夹。
- /src/org.cadenhead.android/SalutonActivity.java：在该 app 运行时，将会默认执行的行为类。
- /gen 文件夹：用于存放生成的 Java 源代码，你不需要手动编辑这些代码。
- /gen/org.cadenhead.android/R.java：为该 app 自动生成的资源管理源代码（不要手动编辑！）。
- /assets：不会被编译为 app 的文件夹或文件资源。
- /res：存放应用程序资源（如字符串、数字、布局文件、图形和动画）的文件夹。而且特殊的资源类型（如 layout、values、drawable-hdpi、drawable-ldpi 和 drawable-mdpi）可以使用子文件夹来存放。这些文件夹包含 5 个资源文件：icon.png 的 3 个版本、

main.xml 和 strings.xml。

- AndroidManifest.xml: app 的主配置文件。
- default.properties: 由 Android 插件生成的构建 (build) 文件, 不要对其进行编辑。
- proguard.cfg: ProGuard 的配置文件, ProGuard 是一款优化 app 的工具, 而且可以使源代码更难以被反编译。

这些文件形成了应用程序的框架。作为 Android 程序员, 你要做的第一件事就是学习如何修改该框架, 以便发现每个组件要完成的功能。

该框架中还包含实现特定目的的其他文件。

24.2.2 创建 app

尽管到目前为止什么也没有做, 但是你可以成功地运行这个 Android 项目。该项目的框架可以作为一个 app 运行。

但是这一点也不好玩, 因此可以自定义 SalutonMondo app, 使其显示传统的计算机编程问候语 “Saluton Mondo!”。

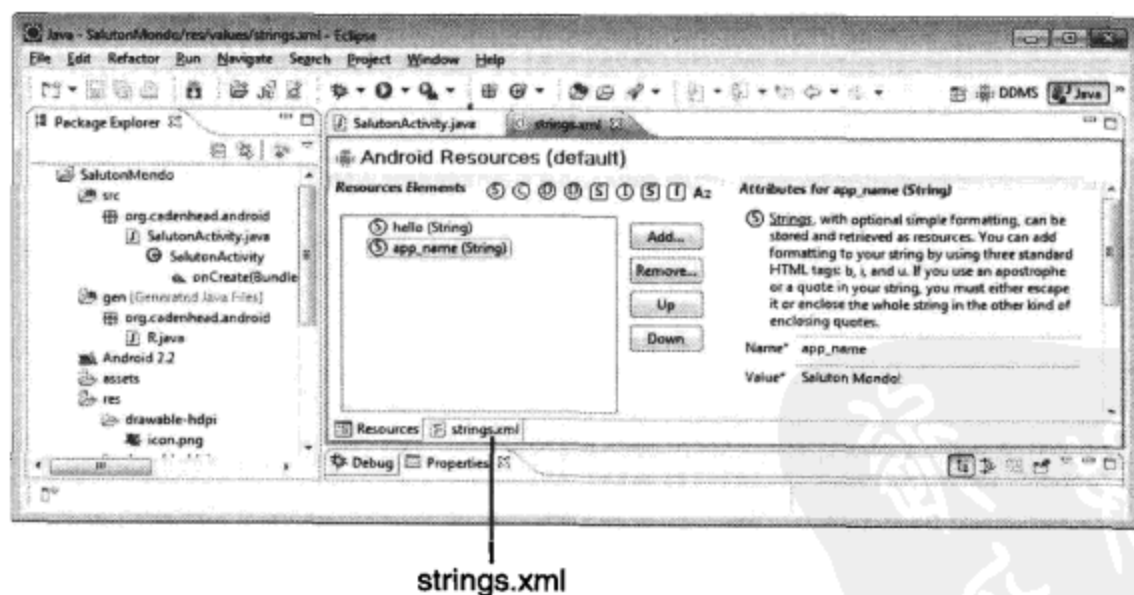
在第 2 章, 是通过调用 `System.out.println()` 方法, 并将文本 “Saluton Mondo!” 作为字符串显示出来的。

Android 显示已经存储在 strings.xml 资源文件中的字符串。可以在 /res/values 文件夹中找到该文件。

在 Package Explorer 中导航到该文件夹。双击 strings.xml, 打开资源编辑器, 如图 24.3 所示。

图 24.3

编辑 Android app
的字符串资源



和 Java 中的变量一样, 字符串和其他资源都有名字和值。在资源元素面板中列出了两个字符串资源: hello 和 app_name。

在给资源命名时, 需要遵循 3 个规则:

- 必须小写;
- 没有空格;

➤ 只能使用下划线字符作为标点。

在资源元素面板中单击一个字符串，将会出现出现 Name 文本框和 Value 文本框，而且还会带有如何编辑字符串的指导（见图 24.3）。

在运行 New Android Project Wizard 时，需要选择 app_name 字符串资源。该名字应该与之前命名的名字匹配，但是可以随时修改该字符串。

hello 字符串包含 app 运行时，显示在 app 主界面（也是唯一的界面）的文本。单击该字符串的名字可以对其修改。

在 Value 文本框，输入“Saluton Mondo!”。

资源存储在 XML 文件中。Resources 编辑器是一个简单的 XML 编辑器。你也可以直接编辑 XML 文件自身。单击编辑器底部的 strings.xml 选项卡，载入该文件，即可直接进行编辑（该选项卡在图 24.3 中做出了标识）。

此时，strings.xml 看起来如下所示：

Output ▼

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Saluton Mondo!</string>
  <string name="app_name">Saluton Mondo!</string>
</resources>
```

在该编辑器中可以修改 XML 文件中的所有内容，甚至是标记。string 元素包含一个 name 属性，表示该资源的名字。该元素的值作为字符数据封装在标记内部。

返回 Resources 编辑器，单击 Resources 选项卡。然后在 Eclipse 工具栏中单击 Save 按钮来保存对 strings.xml 文件做出的修改。

修改之后，即可准备运行该 app。

警告：

尽管你可以直接修改 XML 文件，但最好别修改。在创建 Android app 的资源时，通常没有必要修改 XML 文件。但是，如果在定义资源时，Eclipse 编辑器对其中的某些内容不支持，则需要修改 XML 文件。在字符串中不存在这样的问题，所以最好别在 Resources 编辑器中修改，否则将会带来错误。

Watch Out!

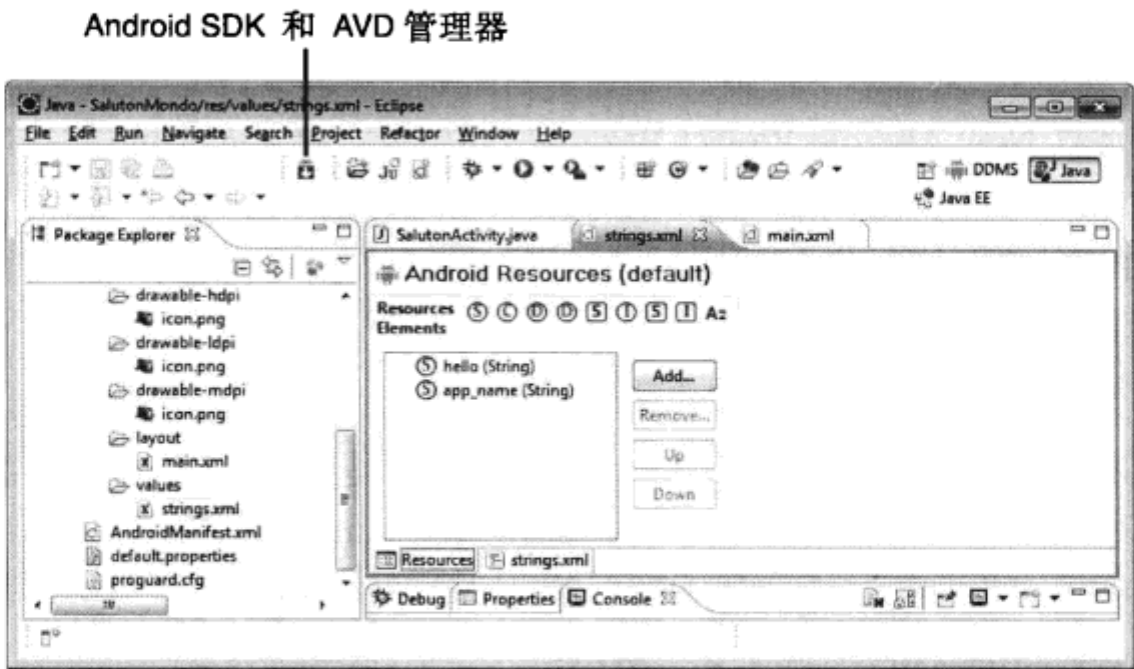
24.2.3 安装 Android 模拟器

在生成（build）Android app 之前，必须设置其调试环境，这可以在 Eclipse 内处理。你必须在桌面计算机上安装可以运行 app 的 Android 虚拟设备（Android Virtual Device, AVD），将其作为模拟器。你还必须创建项目的调试配置。这一切完成之后，你可以生成该 app，并在模拟器中运行它。

为了配置 Android 虚拟设备，首先在 Eclipse 工具栏中单击一个绿色的 Android 图标，它

带有一个向下的箭头，如图 24.4 所示。

图 24.4
配置 Android 虚拟设备



这将启用 Android SDK 和 AVD 管理器，这是 Android SDK 中的一个工具。在左侧面板单击 Vitural Devices 条目，则右侧会列出已经创建的模拟器。管理器如图 24.5 所示。

图 24.5
创建一个新的 Android 模拟器



为了增加一个新的模拟器，单击 New 按钮，然后执行如下步骤。

1. 在 Name 文本框中输入 “SimpleAVD”。
2. 在 Target 下拉列表中，选择 Android 的目标版本。这里选择的是 Android 2.2 – API Level 8。
3. 在 Size 文本框选择模拟 SD 卡的大小。输入 1024 然后从后面的下拉列表中选择 MiB，这表示 SD 卡的大小为 1024MB。你的计算机上必须有足够可用的空间，如果你不希望占据太大空间，则可以调小该值。最小值为 9MB。
4. 单击 Create AVD 按钮，很快就可以创建一个新的模拟器（通常不会长于 1 分钟）。

可以根据需要创建模拟器。可以对它们进行自定义，使其用于不同的 Android 版本的显示类型。

关闭 Android SDK and AVD 管理器，然后到 Eclipse 主界面。

24.2.4 创建调试配置

在启用 SalutonMondo app 之前，需要做的最后一件事是在 Eclipse 中创建调试配置，其步骤如下。

1. 选择 Run->Debug Configuratons，打开 Debug Configurations 窗口。
2. 在左侧面板中，双击 Android Application 条目（见图 24.6），可以看到新创建了一个 New_Configuration 条目，该条目作为 Android Application 的子项。右侧面板将显示这个新条目的一些配置选项。

Android Application 条目

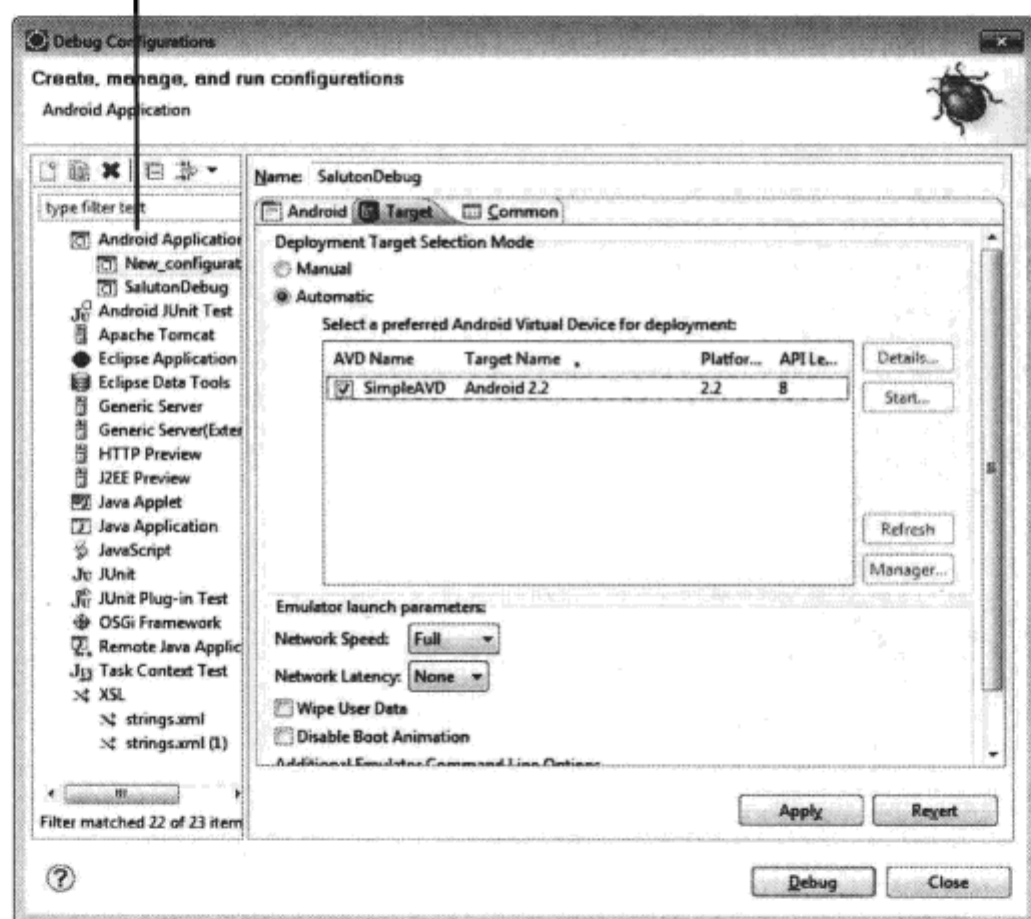


图 24.6

创建 Android 调试配置

3. 在右侧面板中的 Name 文本框中，将其值修改为 SalutonDebug。
4. 单击 Browse 按钮，打开 Project Selection 对话框。
5. 选择 SalutonMondo 项目，然后单击 OK。
6. 单击 Target 选项卡。
7. 在 Deployment Target Selection Mode 下，选择 Automatic（如果之前没有被选择）。然后在表格中选择目标 AVD。
8. 在表格中，选择 SimpleAVD 模拟器复选框。
9. 单击 Apply 按钮保存所有变更，然后单击 Close。

24.3 运行 app

现在你已经有了 Android 模拟器，并创建了调试配置，接下来可以运行这个 app 了。单击位于 Package Explorer 顶部 SalutonMondo，然后单击 Eclipse 工具栏中的调试图标。

Android 模拟器将该 app 载入到它自己的窗口中，这可能需要一分钟甚至更长的时间，所以在它启动的过程中，请耐心等待。

该模拟器将“Saluton Mondo!”作为文本和 app 的标题显示出来，如图 24.7 所示。模拟器可以像手机那样工作，但是此时需要用鼠标单击按钮（而不是用手）来辅助它的运行。单击“Back”按钮，关闭 app，来看一下它是如何模拟 Android 设备的。

图 24.7

在 Android 模拟器
中运行 app



和真实设备一样，模拟器可以做很多事情，比如在计算机已经连接到 Internet 的前提下，模拟器也可以连接到 Internet，而且它还可以接受虚假的电话呼叫和短消息。

模拟器毕竟不是功能齐备的设备，因此你开发的 app 必须要在真实的 Android 手机和平板电脑上进行测试。

如果可以使用 USB 线缆将 Android 手机（或其他设备）连接到计算机，在该手机被设置为调试模式的前提下，你可以在上面运行 app。使用 Android SDK 开发的 app 只能部署在处于调试模式的手机中。

在手机中，通过选择 Home->Settings->Applications->Development，进入调试模式。此时将显示 Development settings，从中选择 USB debugging 选项。

接下来，在 Eclipse 中执行如下步骤：

1. 选择 Run->Debug Configurations，打开 Debug Configuration 窗口。
2. 在右侧面板中单击 Target 选项卡。

3. 将 Deployment Target Selection Mode 从 Automatic 修改为 Manual。

4. 单击 Apply 和 Close。

使用 USB 线缆将 Android 手机连接到计算机上。此时将会在屏幕顶部的工具栏中出现一个 Android bug 图标。如果将该工具栏拖动到窗口中，将会看到消息“USB Debugging Connected”。

返回 Eclipse，单击工具栏中的 bug 图标，打开 Android Device Chooser 对话框（见图 24.8）。

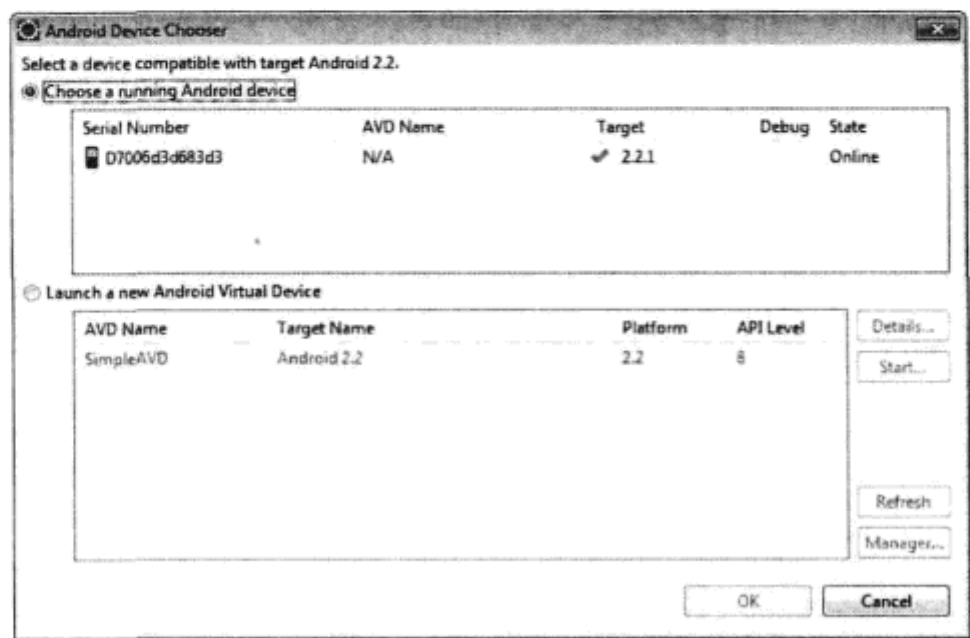


图 24.8

在 Android 手机上部署 app

如果 Android 手机被检测到，它将显示在图 24.5 中 Choose a Running Device 选项下面的表格中。

选中 Choose a Running Device 选项，单击手机设备的名字，单击 OK。该 app 将运行在手机中。

如果你在第 2 章创建的第一个 Java 程序那样，你创建的第一个 Android app 非常稀松平常。你接下来将要创建的这个项目将不会如此。

24.4 设计真实的 app

Android app 可以使用设备的所有功能，比如短消息服务、基于位置的服务、以触摸进行的输入。在本章最后第一个编程项目中，你将创建一个真实的服务，名字为 Take Me To Your Leader。

该 app 使用了 Android 手机的功能来进行呼叫、浏览网站，以及使用 Google 地图进行定位。该 app 可以让你通过手机、Web 和地图与白宫取得联系。

为了开始该项目，先执行如下步骤，以在 Eclipse 中创建一个新的项目。

1. 单击 File->New->Android Project，打开 New Android Project 向导。

2. 在 Project Name 文本框输入 Leader。

3. 确保 Create New Project 被选中。

4. 选择 Build Target Android 2.2。

- 5. 在 Application Name 文本框输入 Take Me To Your Leader。
- 6. 在 Package Name 文本框输入 org.cadenhead.android。
- 7. 确保 Create Activity 被选中，然后在后面的文本框中输入 LeaderActivity。此时 New Android Project 向导应该如图 24.9 所示。

图 24.9
创建一个新的
Android 项目



- 8. 单击 Finish。

和 SalutonMondo 项目一样，该项目也会出现在 Eclipse Package Explorer 中。为了避免混淆，在进行下一步操作之前关闭 SalutonMondo 项目。在 Package Explorer 中右键单击 SalutonMondo，然后从弹出菜单中选择 Close Project。

Did you know?

提示：

该项目包含了大量的知识。随着工作的进行，你会发现，将浏览器打开并定位到 Android Developer 站点的 Reference 内容 (<http://developer.android.com/reference>)，将会为你提供很多帮助。你可以在 Android 类库中搜索 Java 类，以及项目中出现的文件的名字，以获悉更多内容。

24.4.1 组织资源

在创建 Android app 时，需要使用 Java 来编程，但是大部分工作是在 Eclipse 界面中完成的。在你完全精通 Android SDK 的功能时，不用编写一行 Java 代码就可以完整大部分工作。

为了无需编程，首先要创建供 app 使用的资源。每一个新的 Android 项目在开始时都有几个放置了资源的文件夹。要查看这些文件夹，在 Package Explorer 中展开 Leader 文件夹，

然后展开/res 文件夹以及它所有的子文件夹（见图 24.10）。

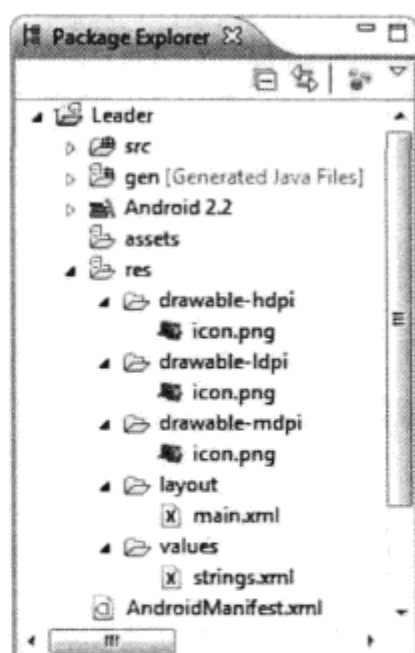


图 24.10

查看 app 的资源文件夹

资源包含 PNG/JPG/GIF 格式的图形、存储在 strings.xml 文件中的字符串、XML 格式的用户界面布局文件，以及你可以创建的其他文件。通常需要添加到项目中的两个文件是 color.xml 和 dimens.xml，其中前者是 app 中用到的颜色，后者用于设置文本大小和其他要显示内容的尺寸。

新项目中的/res 文件夹包含如下几个文件夹：drawable-hdpi、drawable-mdpi、drawable-ldpi，其中每个文件夹中存放着 icon.png 的一个版本。icon.png 是 app 的图标，这是一个比较小的图形，用来启用 app。

icon.png 的这 3 个版本是相同的图形，但是分辨率不同。你在这里不会用到这些图标，因此可以将它们删除：在 Package Explorer 中单击一个 icon.png 文件，然后按下键盘上的 Delete 键。在删除每一个文件时，系统都会要求用户进行确认。

删除这些文件时，Package Explorer 中将出现两个红色的 X：一个位于 AndroidManifest.xml 上面，另一个位于最顶层的 Leader 上面（图 24.11 中对其进行了标识）。这些 X 符号标识该 app 发生了错误，该错误将阻止 app 的编译和运行。

由于 app 现在缺乏图标，因此引发了错误。在项目添加一个新的图形文件 appicon.png，然后在 AndroidManifest.xml 文件（app 的主配置文件）中将其指定为该 app 的图标。

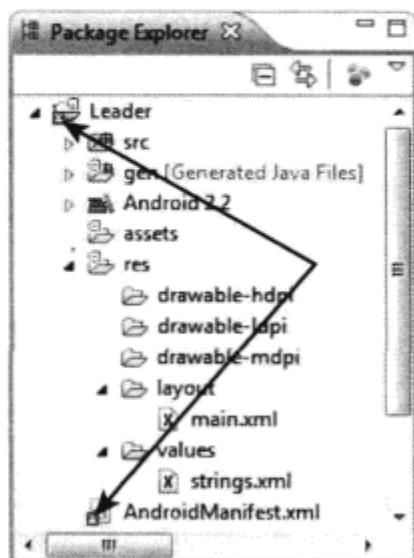


图 24.11

检测和修复 app 中的错误

本书配套站点包含该 app 需要的 appicon.png 和另外 4 个图形文件: browser.png、maps.png、phone.png 和 whitehouse.png。访问 www.java24hours.com, 然后定位到第 24 章的页面, 下载这 5 个文件, 并将它们存放到计算机的临时文件夹中。

Android 支持多个分辨率, 但是这里不会用到。这里不是使用已有的 drawable 文件夹, 而是创建一个新的文件夹, 步骤如下。

1. 在 Package Explorer 中单击/res 文件夹。
2. 选择 File->New->Folder, 打开 New Folder 对话框。
3. 在 Folder Name 文本框中输入 drawable。
4. 单击 Finish。

/res 文件夹中将创建一个名为 drawable 的新文件夹。app 需要的所有图形可以存到这里, 而且不用考虑它们的分辨率。

使用拖放技术可以将文件添加到资源中。打开包含 5 个文件的临时文件夹, 选择它们, 然后拖放到 Package Explorer 中的 drawable 文件夹中

现在该项目有了新的图标, 可以将其设置为 app 的图标 (通过编辑 AndroidManifest.xml 来实现), 这样 Package Explorer 中出现的错误图标将消失。

Watch Out!

警告:

app 中的资源使用 ID 来表示, 将资源名字的扩展名去掉后, 就是其 ID。比如, appicon.png 的 ID 是 appicon, browser.png 的 ID 是 browser。不同资源的 ID 不可能相同 (以不同分辨率存储在 3 个 drawable-*dpi 文件夹中的同一个图形是一个例外, 因为这 3 个图形被当做一个资源)。

如果两个资源的名称相同, 但是扩展名不同, 比如 appicon.png 和 appicon.gif, 则 Eclipse 会标识错误, 而且该 app 无法编译。

资源的名称只能包含小写字母、数字、下划线和点号。该项目中的文件遵循了这些规则。

24.4.2 配置 app 的 Manifest 文件

在 Android app 中, 主要的配置工具是名为 AndroidManifest.xml 的文件, 它位于 app 主文件夹中。app 使用的所有 XML 文件都可以手动编辑, 或使用 Eclipse 中内置的编辑器来编辑。其中后者使用起来比较方便, 而且不容易发生错误。当你编写 Android 程序的经验逐渐丰富时, 可以考虑修改 XML 文件, 否则不建议修改。

为了为 app 选择适合的图标, 请执行如下步骤。

1. 在 Package Explorer 中双击 AndroidManifest.xml, 该文件将在 Eclipse 内置的编辑器中打开。
2. 编辑器的底部位置将出现几个选项卡。单击 Application 选项卡来查看与该 app 相关的设置 (见图 24.12)。

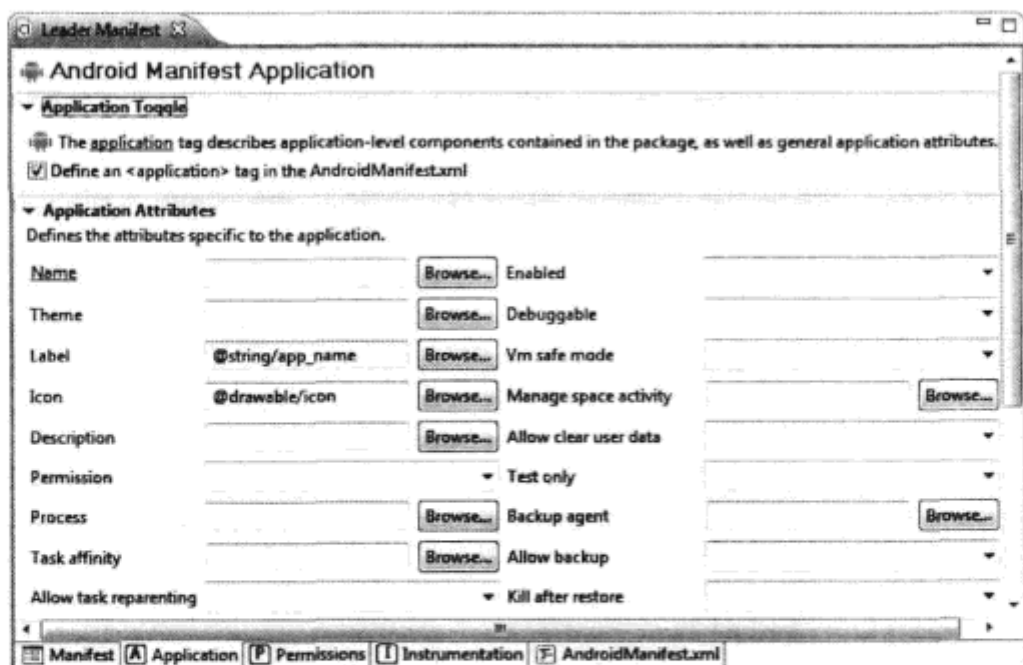


图 24.12

编辑 app 的
AndroidManifest.
xml 文件

3. Icon 文本框表示 app 的图标，当前在文本框内的值@drawable/icon 不正确。单击该文本框后面的 Browse 按钮，打开 Resource Chooser 对话框，其中列出了 app 中包含的 5 个“drawable”资源。

4. 选择 appicon 并单击 OK。Icon 文本框现在有了正确的值。

5. 保存文件：单击 Eclipse 工具栏中的 Save 按钮，或者是选择 File->Save。

红色的 X 将从 Package Explorer 中消失，这表示已经为 app 指定了合适的图标。

24.4.3 设计用户界面

app 的图形用户界面由布局组成，布局是用来放置文本框、按钮、图形和自定义控件的容器。向用户显示的每一个画面都有一个或多个布局。有的布局可以水平或垂直地叠放组件，有的布局可以将组件组织到表中，有的布局还有其他布置方式。

app 可以有一个或多个画面。一个游戏可以包含如下画面：

- 游戏在载入时显示的运行画面；
- 带有按钮（用以查看其他画面）的主菜单画面；
- 显示游戏规则的帮助画面；
- 列出最高游戏得分的分数画面；
- 包含游戏开发人员的致谢画面；
- 游戏进行中的画面。

Leader app 包含了一个画面，上面放置的按钮用来练习美国总统，或者其他领导人。

app 的所有画面都存放在/res/layout 文件夹中。对于一个新项目而言，该文件夹中存放了一个 main.xml 文件，而且该文件已经被指定为 app 在载入时所要显示的画面。

为了编辑该画面的布局，在 Package Explorer 中双击 main.xml 文件，将会在 Eclipse 主窗口中打开该画面，如图 24.13 所示。

图 24.13

编辑 app 的
main.xml 文件



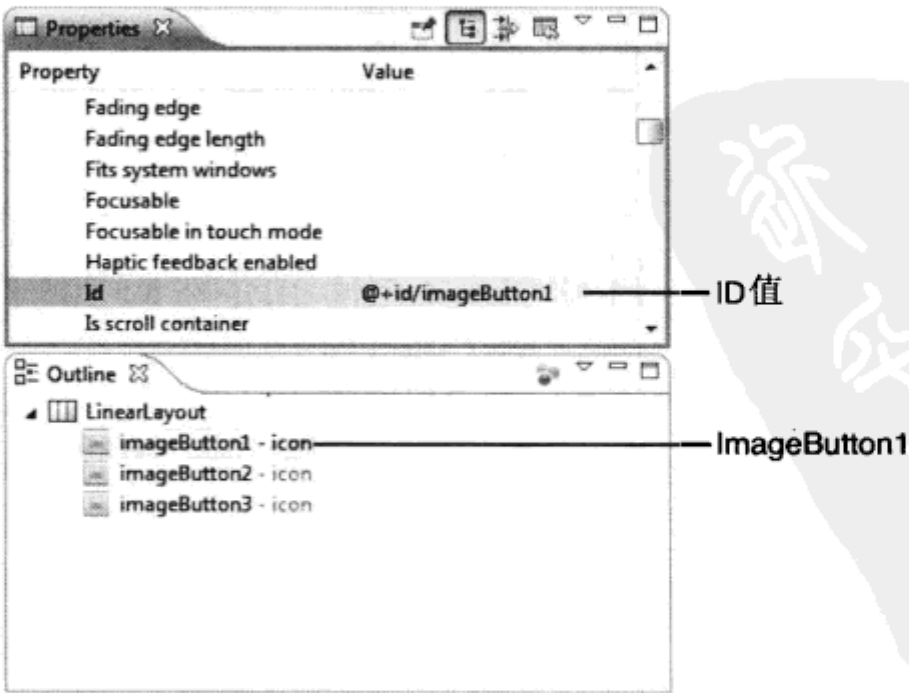
编辑窗口中包含一个 **Palette** 面板以及几个可以展开的文件夹。**Form Widgets** 子面板（有可能已经展开）显示一些简单的控件（**widget**），这些控件可以立刻被拖放到画面中。

按照如下步骤将 3 个图形按钮添加到画面中。

1. 删除显示“Hello World”文本的 **textview** 控件。在画面中单击该控件，然后按下 **Delete** 键。
2. 双击 **Palette** 面板中的 **Image & Media** 文件夹，将其展开。
3. 从 **Palette** 中拖动 **ImageButton** 控件到画面中。画面中出现一个狭窄的蓝色盒子，而且画面下面会出现一条错误消息。只所以出现该错误，是因为按钮缺乏图片——不用担心。
4. 拖动另外两个 **ImageButton** 控件到画面中。它们将会在画面中垂直叠放。
5. 一个 **Outline** 面板列出画面中的控件。从中选择 **imageButton1**，该按钮的属性在 **Properties** 面板中打开（见图 24.14）。

图 24.14

自定义控件的属性



6. 滚动 Properties 面板，直到发现 ID 属性为止。ID 属性的值当前被设置为 @+id/imageButton1。将其修改为 @+id/phonebutton。
7. 滚动到 Src 属性，其当前值是 drawable/icon。单击该值，出现一个椭圆形按钮 (...).
8. 单击该按钮，打开 Reference Chooser 对话框。
9. 展开 Drawable 标题，可以看到 app 的图形列表，这些图形是之前添加进去的资源。选择 phone 然后单击 OK。现在按钮有了一个电话图形。
10. 在 On Click 属性，输入值 processClicks（这将在下一节解释）。
11. 针对 imageButton2，重复步骤 5~10，将其 ID 属性设置为 @+id/webbutton，Src 属性设置为 drawable/browser。
12. 针对 imageButton3，重复步骤 5~10，将其 ID 属性设置为 @+id/mapbutton，Src 属性设置为 drawable/maps。
13. 单击画面上方的 Set Horizontal Orientation 按钮（见图 24.13），按钮现在一字排开。
14. 单击 Outline 面板中的 LinearLayout，画面的属性将出现在 Properties 面板中。
15. 单击 Background 的值，然后单击椭圆形按钮 (...)，打开 Reference Chooser。
16. 展开 Drawable，选择 whitehouse，然后单击 OK。此时白宫的图片将成为画面的背景。
17. 单击 Save 按钮。

完成后的画面如图 24.15 所示。



图 24.15

预览 app 的图形
用户界面

24.4.4 编写 Java 代码

此时，你已经完成了该 app 的大部分工作，但是你还没有编写一行 Java 代码。当你充分使用 Android SDK 的功能，而不求助于编程时，app 的开发工作无疑会轻松很多。

app 被组织为 Activities, 这表示 app 可以处理的任务。每一个 Activity 都由它所属的 Java 类来定义。当创建 app 时, 需要确定名为 LeaderActivity 的 Activity 已经创建。这样, 当 app 运行时, 匹配该名字类能够自动运行。

LeaderActivity.java 的源代码可以在 Package Explorer 中找到, 它位于/src/org.cadenhead.android 文件夹中。双击该文件, 以进行编辑。

刚开始时, 该类的代码如程序清单 24.1 中所示。

程序清单 24.1 LeaderActivity.java 在刚开始时的文本

```
1: package org.cadenhead.android;
2:
3: import android.app.Activity;
4: import android.os.Bundle;
5:
6: public class LeaderActivity extends Activity {
7:     /** Called when the activity is first created. */
8:     @Override
9:     public void onCreate(Bundle savedInstanceState) {
10:         super.onCreate(savedInstanceState);
11:         setContentView(R.layout.main);
12:     }
13: }
```

与所有 Activities 相同, LeaderActivity 类是 Activity 的子类, 后者位于 android.app 包中, 它包含了显示画面、收集用户输入、保存用户设置等所需要的所有行为。

第 9~12 行定义的 onCreate() 方法将在载入 LeaderActivity 类时被调用。该方法首先使用 super 调用其超类中的同名方法。然后, 再调用 setContentView() 用来选择要显示的画面。传递给该方法的参数是一个实例变量 R.layout.main, 它引用/res/layout 中的 main.xml 文件。你应该还记得, 资源的 ID 是去除扩展名后的文件名。

Did you know?

注意:

你可以打开并编辑/res/gen/org.cadenhead.android 文件夹中的 R.java 文件, 以获悉 main 资源被称为 R.layout.main 的原因。R 类是由 Android SDK 自动生成的, 其目的是确保能够通过资源的 ID 来引用资源。你最好不要修改该类。

在 LeaderActivity 类中, 首先要创建一个类变量。在类定义语句后面添加如下语句:

```
public static final String TAG = "Leader";
```

该变量作为该类的标识符使用, 可以用来记录在类运行时所发生的事件。Android 类可以等级它们的行为, 以让你知道 app 中发生了什么。下面是一条记录 (log) 语句, 你在后面会用到:

```
Log.i(TAG, "Making call");
```

这条语句显示一条记录消息, 而且使用名字 Leader 进行标记。

android.util 包中的 Log 类显示记录中的消息。该类有 5 个方法来记录消息, 每一个方法

都可以只是消息的类型，比如警告消息、调试消息或错误消息。方法 `i()` 表示信息 (information) 消息，该消息可以解释 app 中发生了什么。

传递给 `Log.i()` 的第一个参数表示 app，第二个参数包含要显示的消息。

在前面设计 app 用户界面时，你将每一个按钮的 On Click 属性设置为 `processClicks`。这表示当用户单击画面中的按钮控件时，将会调用 `processClicks()` 方法。现在来实现该方法。在 `LeaderActivity` 中的 `onCreate()` 方法下面添加如下语句：

```
public void processClicks(View display) {
    Intent action;
    int id = display.getId();
}
```

在调用该方法时需要用到一个参数 `View`，这是来自 `android.view` 包中的 `View` 类对象。`View` 是 app 中某些可视化的显示类型。在本例中，它是包含 `Dialer`、`Browser` 和 `Maps` 按钮的画面。

`View` 对象的 `getId()` 方法返回被单击按钮的 ID：`phonebutton`、`webbutton` 或 `mapbutton`。

该 ID 存储在 `id` 变量中，并在 `switch` 语句中使用，这样可以根据单击的按钮采取相应的行为：

```
switch (id) {
    case (R.id.phonebutton):
        // ...
        break;
    case (R.id.webbutton):
        // ...
        break;
    case (R.id.mapbutton):
        // ...
        break;
    default:
        break;
}
```

该代码的 `switch` 语句使用每一个 ID 的整数值作为参数，并执行 3 种行为中的一种。方法 `processClicks()` 中的第一个语句创建了一个变量来存放一个 `Intent` 对象。`Intent` 是 Android 的 `android.content` 包中的一个类：

```
Intent action;
```

在 Android 中，使用 `Intent` 来让一个 `Activity` 告诉另外一个 `Activity` 要做什么。这也是 app 与 Android 设备进行通信的一种方式。

该方法使用了下面 3 个 `Intent`：

```
action = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:202-456-1111"));

action = new Intent(Intent.ACTION_VIEW,
    Uri.parse("http://whitehouse.gov"));

action = new Intent(Intent.ACTION_VIEW, Uri.parse("geo:0,0?q=White House,
    Washington, DC"));
```


Intent()构造函数接受 2 个参数:

- 要采取的行为, 由它的类变量来表示;
- 与行为相关的数据。

这 3 个 Intent 告诉 Android 设备: 设置一个打给白宫的拨出电话, 其电话号码为 (202) 456-1111; 浏览 <http://whitehouse.gov> 网站; 使用部分地址 “White House, Washington, DC” 来载入 Google 地图。

在创建 Intent 之后, 下面的语句将让它执行相应的任务:

```
startActivity(action);
```

程序清单 24.2 为 LeaderActivity 类的完整文本。在程序清单 24.1 的基础上添加 import 语句 (见程序清单 24.2 中的第 3~8 行) 和 processClicks() 方法。确保你的代码与程序清单 24.2 完全相同。

程序清单 24.2 LeaderActivity.java 的完整源代码

```

1: package org.cadenhead.android;
2:
3: import android.app.Activity;
4: import android.content.Intent;
5: import android.net.Uri;
6: import android.os.Bundle;
7: import android.util.Log;
8: import android.view.View;
9:
10: public class LeaderActivity extends Activity {
11:     public static final String TAG = "Leader";
12:
13:     /** Called when the activity is first created. */
14:     @Override
15:     public void onCreate(Bundle savedInstanceState) {
16:         super.onCreate(savedInstanceState);
17:         setContentView(R.layout.main);
18:     }
19:
20:     public void processClicks(View display) {
21:         Intent action;
22:         int id = display.getId();
23:         switch (id) {
24:             case (R.id.phonebutton):
25:                 Log.i(TAG, "Making call");
26:                 action = new Intent(Intent.ACTION_DIAL,
27:                     Uri.parse("tel:202-456-1111"));
28:                 startActivity(action);
29:                 break;
30:             case (R.id.webbutton):
31:                 Log.i(TAG, "Loading browser");
32:                 action = new Intent(Intent.ACTION_VIEW,
33:                     Uri.parse("http://whitehouse.gov"));
34:                 startActivity(action);
35:                 break;
36:             case (R.id.mapbutton):
37:                 Log.i(TAG, "Loading map");
38:                 action = new Intent(Intent.ACTION_VIEW,
39:                     Uri.parse("geo:0,0?q=White House, Washington, DC"));
40:                 startActivity(action);
41:                 break;

```



```

42:         default:
43:             break;
44:     }
45: }
46: }

```

输入完毕之后保存该文件。该文件应该会成功编译（Eclipse 将自定执行该编译）。如果没有，则红色的 X 将显示在 Package Explorer 中，指示项目中出错的文件。

当修复错误后，则可以准备运行该 app。

你必须先为该项目创建一个新的调试配置：

1. 在 Eclipse 的主工具栏中单击靠近 Debug 按钮的箭头，然后选择 Debug Configurations，打开 Debug Configuration 对话框。
 2. 双击左侧面板的 Android Application，创建一个名为 New_configuration (1) 的新配置。
 3. 在 Name 文本框中输入 LeaderDebug。
 4. 单击 Browse 按钮，选择 Leader 项目，然后单击 OK。
 5. 单击 Target 选项卡。
 6. 在 Deployment Target Selection Mode 下选中 Automatic，然后选择 SimpleAVD Android 虚拟设备。
 7. 将 Deployment Target Selection Mode 更改为 Manual，单击 Apply，然后单击 Close。
- 现在，创建了一名为 LeaderDebug 的新调试配置。

要运行该程序，单击靠近 Debug 按钮的箭头，然后选择 LeaderDebug（如果有的话）。如果没有，则选择 Debug Configuration，再选择 LeaderDebug，然后单击 Debug。此时打开 Android Device Chooser。选择 Launch a New Android Virtual Device，然后选择 SimpleAVD，然后单击 OK。

在接下来的几分钟时间里，模拟器将载入并自动运行该 app。

模拟器并不能模拟 Android 设备的一切行为。这个 Leader app 的 Dialer 和 Browser 按钮应该能够正常工作，但是在使用 Maps 按钮时，可能会遇到问题。

该 app 也可以在 Android 手机上运行，前提是该手机可以运行在 Android SDK 中，而且已经被设置为调试模式。单击 Debug 按钮附近的箭头，然后选择 LeaderDebug，此时该选项应该绝对存在。选择 Choose a Running Android Device，在列表中选择你的手机，然后单击 OK。

图 24.16 为运行在我手机上的 app。当手机从肖像模式切换到风景模式时，该 app 也随之切换（该截图还显示我有 141 个新的语音信息，我应该查看一下）。



图 24.16

在手机上运行该 Leader app

这个 Leader app 也将它自己的“Take Me to Your Leader”图标添加到手机应用程序中。即使你拔掉 USB 线缆，该 app 也会在手机上继续运行。

祝贺你！现在世界上有了 10 亿零 1 个 app 了。

Did you Know?

注意：

读者可能也猜到，Android 编程所需要的知识肯定要比本章讲解的内容要多。Sams 出版社还出版了其他 Android 编程相关的图书，如《Sams Teach Yourself Android Application Development, 2nd Edition》（由 Lauren Darcey 和 Shane Conder 编写）。

24.5 总结

本书的目标是帮助读者熟悉编程的概念，提升编写应用程序的信心，无论应用程序是用于桌面计算机、Web 页面、Web 服务器，还是手机。

当你具有了 Java 编程经验之后，其他相关的经验也会随之增长，因为像面向对象编程、虚拟机和安全环境等概念在软件开发中都处于前沿地位。

如果你的 Java 编程经验还有所欠缺，可以查看该书的附录，以寻找有用的信息。

在本章内容结束后，你可以通过多种途径继续学习 Java 语言。你可以访问 <http://weblogs.java.net> 站点来查看有关 Java 语言的讨论，在 <http://www.careerbuilder.com> 这样的网站上会列出大量的与 Java 编程有关的工作机会。此外，在该书的配套站点 <http://www.java24hours.com> 上可以找到作者的信箱、查看该书练习的答案，以及勘误等内容。

作为一名 Java 程序员，你可以通过阅读 Sams Teach Yourself Java in 21 Days 来进一步提到你的编程技能。我也是该书的作者，该书对这里讲到的各个主题进行了扩展，此外还添加了很多新的主题，比如 JDBC、Java servlets 和网络编程。本书还会涵盖 Android 的内容。

24.6 问与答

问：为什么使用 Eclipse 而不是 NetBeans 来创建 Android app？

答：也可以使用 NetBeans 来开发 app，但是使用起来会相当麻烦，而且对 Android 编程的支持也不是很好。Eclipse 被 Google 指定为首选的 Android IDE。Android Developer 站点 (<http://developer.android.com>) 上的官方文档和指南使用的都是 Eclipse。

大多数的 Android 编程图书使用的也是 Eclipse。尽管当你从 NetBeans 切换到 Eclipse，以进行 Android 开发时，会存在一个适应阶段。但是当你掌握了 app 的编写、调试和部署基础之后，就会发现 Eclipse 更容易使用，因为它得到了程序员和技术作者更好的支持。

问：ProGuard 是如何使 app 的源码更难以被反编译的？

答：Java 类文件很容易使用逆向工程来破解。逆向工程是接受可执行代码，分析出相应

源代码的过程。由于 Android app 的设计者不希望自己编写的代码被其他开发人员使用，因此在创建 Android 项目时可以使用 ProGuard。

在编译类文件时，ProGuard 通过删除其中未使用的代码来优化 app。而且 ProGuard 会将类、字段和方法的名字更改为无意义而且模糊的内容，这样即使有人反编译了 Java 代码，也很难轻易地读懂源代码。

只有当 app 在发布模式下创建时，才使用 ProGuard 的模糊特性。如果过早地使用该特性，会让调试变得很困难。

24.7 测验

24.7.1 问题

1. 下面哪一个公司不是开放手机联盟中的成员？
 - a. Google。
 - b. Apple。
 - c. Motorola。
2. 下面哪一个工具可以增大开发人员窃取 Java 程序源代码的难度？
 - a.反编译器（decompiler）。
 - b.重编译器（recompiler）。
 - c.模糊器（obfuscator）。
3. Android 模拟器不能执行下面哪项任务？
 - a. 接收短消息。
 - b. 连接到 Internet。
 - c. 进行电话呼叫。

24.7.2 答案

1. b. Apple。Android 是作为开源的一部分创建的，不涉及专利性，而且与 Apple iPhone 存在竞争关系。
2. c。
3. c。模拟器并不能模拟真实设备的所有行为，因此它只是测试 app 过程的一部分。

24.8 练习

为了进一步巩固你的 Android 知识，请做如下练习。

- 将 SalutonMondo app 的文本改为“Hello World”，然后在模拟器和 Android 设备（如

果有的话)上运行该 app。

- 针对不同的世界领导人创建 Take Me To Your Leader app 的新版本,并自定义电话、Web 地址和地图信息。

有关为完成这些练习而编写的 Java 程序,请访问本书的配套网站 www.java24hours.com。



附录 A

使用 NetBeans IDE

尽管可以使用 Java 开发工具包(JDK)和文本编辑器来开发 Java 程序,但是如果使用 IDE,由此带来的编程体验会更好。

在本书的前 23 章,使用的是 NetBeans,这是 Oracle 为 Java 程序员提供的一款免费 IDE,它可以更容易地组织、编写、编译和测试使用 Java 开发的软件。NetBeans 包含一个项目和文件管理器、图形用户界面设计器,以及许多其他工具。它的一个杀手级特性是用户在输入代码时,其代码编辑器能够自动检测到 Java 语法错误。

NetBeans 的当前版本是 7.0,它已经成为 Java 专业开发人员最喜欢的一款工具,而且 NetBeans 提供的功能以及自身的性能要物超所值。此外,它还是 Java 新手最容易上手的一款 IDE。

在本附录中,你将学到如何安装 NetBeans,以及如何将其用到本书中的项目中。

A.1 安装 NetBeans

NetBeans IDE 已经逐渐成为一款主流的 Java 编程工具。Java 语言的发明人 James Gosling 在为图书《NetBeans Field Guide》作序时,提到“我一直使用 NetBeans 来开发 Java 程序”。当然,我也皈依到 NetBeans 门下。

NetBeans 对 3 个版本的 Java 语言都提供支持,这 3 个版本是:Java Standard Edition(JSE)、Java Enterprise Edition (JEE) 和 Java Mobile Edition (JME)。它还支持 Web 应用开发、Web 服务和 JavaBeans。

你可以从 www.netbeans.org 上下载该软件,它可以用于 Windows、MacOS 和 Linux。下载 NetBeans 时,可以与 Java 开发工具包(JDK)一起打包下载,如果你的计算机上没有安装该 JDK,则可以进行安装。

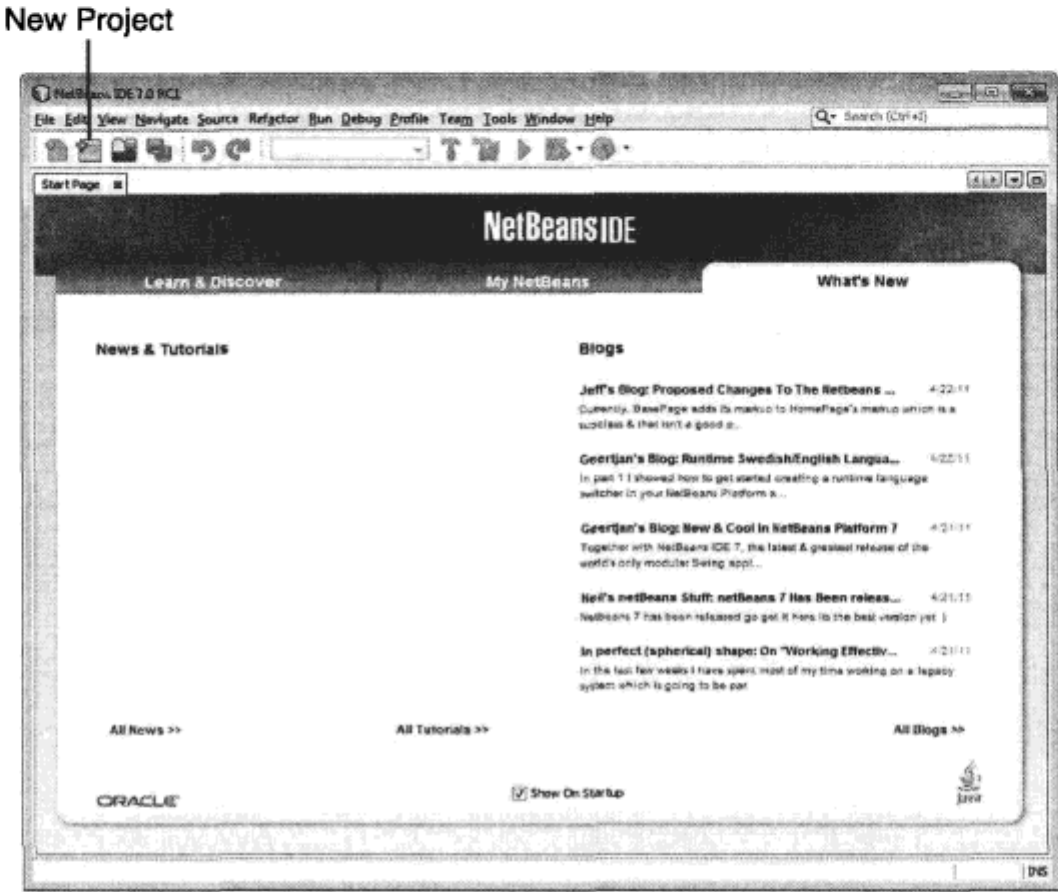
如果你想确保下载的 NetBeans 版本与本书中使用的相同,可以访问本书配套站点 www.java24hours.com。单击该书的封面,在新打开的页面中即可看到下载 JDK 和 NetBeans 7.0 的链接。可以选择从此处下载。

A.2 创建新项目

下载 JDK 和 NetBeans 时，其方式与在计算机上安装软件相同，只不过下载时用到的是下载向导，而安装时用到的是安装向导。你可以根据自己的喜好，将该软件安装到任何文件夹和菜单组中，但是最好不要修改其默认的位置。

安装 NetBeans 后，当第一次运行它时，将会看到一个起始页，它显示相关新闻和编程指南的链接（见图 A.1）。可以使用 NetBeans 内置的 Web 浏览器在该 IDE 中来阅读这些内容。

图 A.1
NetBeans 用户界面



NetBeans 项目包含一组相关的 Java 类、这些类使用的文件，以及 Java 类库。每一个项目都有自己的文件夹，你可以使用文本编辑器和其他编程工具来探索和修改 NetBeans 的外观。

为了开始创建一个新的项目，单击图 A.1 中的 New Project 按钮，或者是选择 File->New Project 菜单命令。打开 New Project Wizard，如图 A.2 所示。

NetBeans 可以创建不同类型的 Java 项目，但是在本书中，你只需要关注 Java Application 即可。

对于你的第一个项目（以及本书中大多数项目）而言，选择项目类型 Java Application，然后单击 Next。该向导会让用户选择项目的名字和位置。

Project Location 文本框指明了使用 NetBeans 创建的编程项目的根文件。在 Windows 中，这将是 My Documents 的子文件夹，名为 NetBeansProjects。你创建的所有项目都存放在该文件夹中，而且每一个项目都有自己的子文件夹。

在 Project Name 文本框，输入 Java24。Create Main Class 文本框也随之改变，并推荐使用 java24.Java24 作为项目中 Java 主类的名字。将其修改为 Spartacus，然后单击 Finish，接受所有其他的默认值。现在 NetBeans 创建了项目和该项目的第一个类。

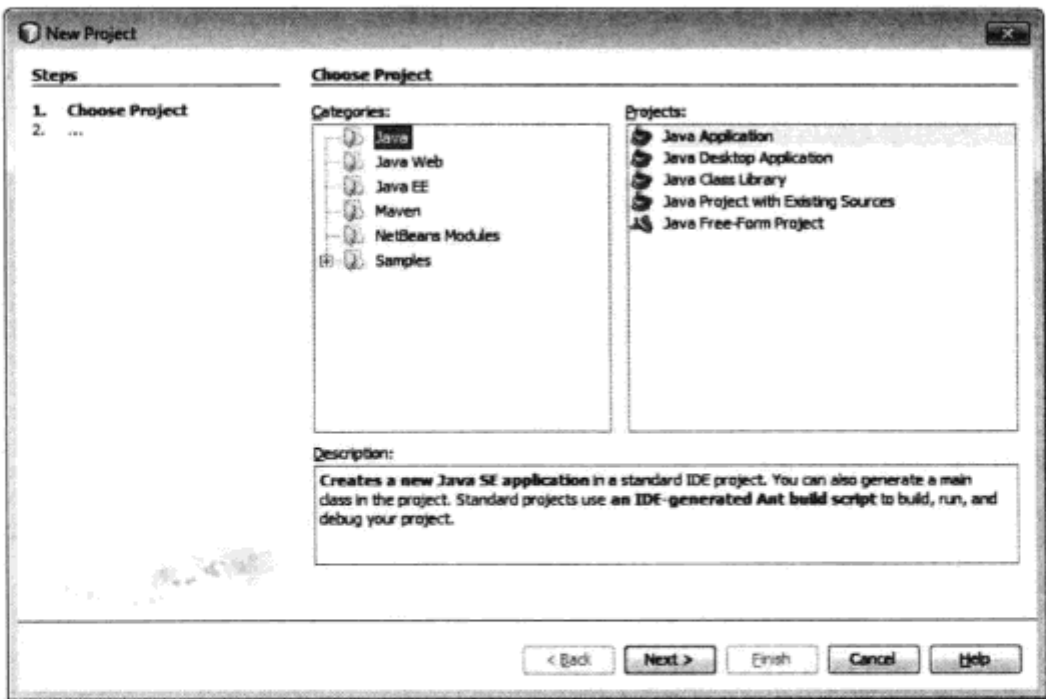


图 A.2

New Project Wizard

A.3 创建新的 Java 类

当 NetBeans 创建了一个新的项目时，它将设置所有需要的文件和文件夹，然后创建主类。图 A.3 为项目 Spartacus.java 的第一个类，它在源代码编辑器中打开。

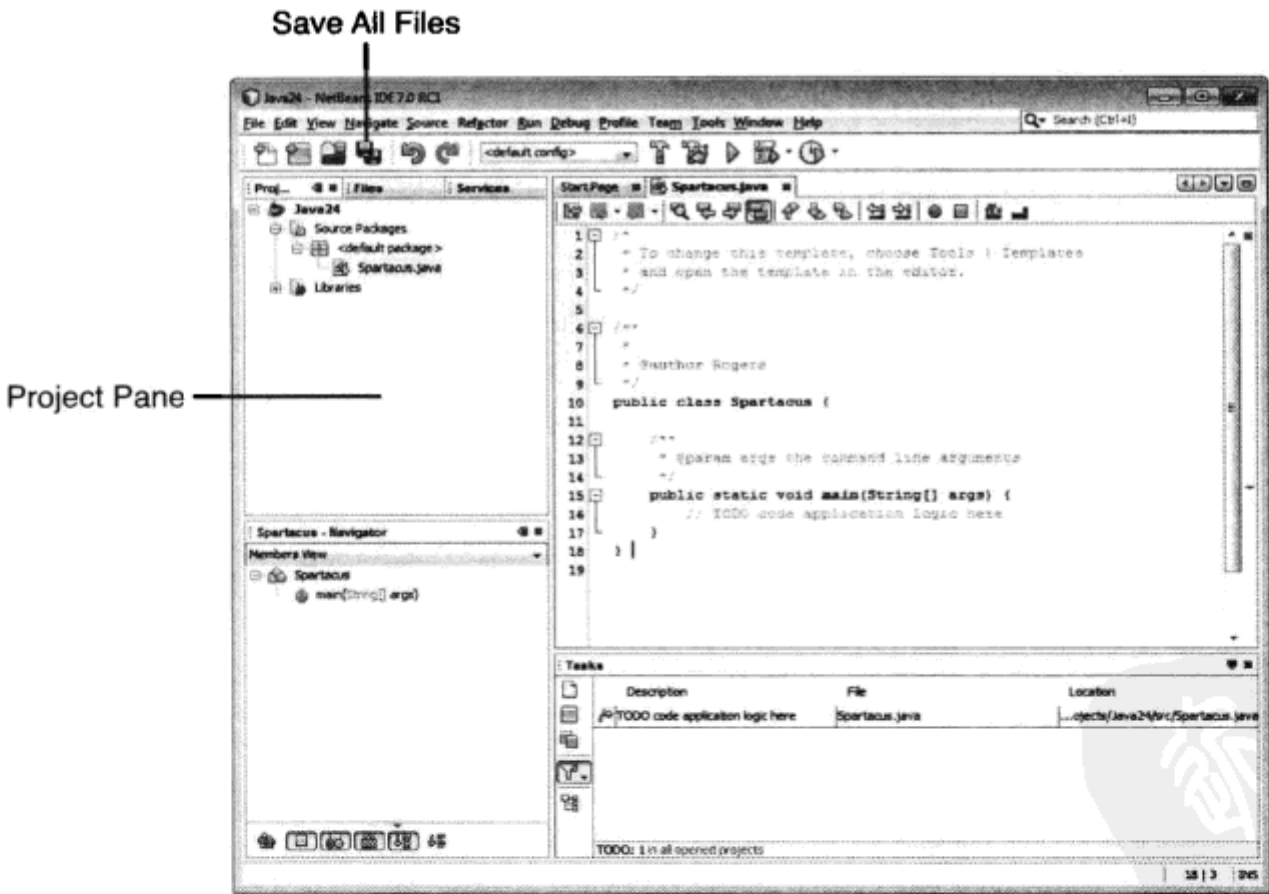


图 A.3

NetBeans 源代码编辑器

Spartacus.java 是 Java 类的一个框架，只包含一个 main()方法。类中以淡灰色显示的所有代码行是注释，其存在的目的是解释类的目的和功能。当类在运行时，会忽略掉注释。

要让类执行一定的任务，在注释行// TODO code application logic here 下面添加一行新的代码：

```
System.out.println("I am Spartacus!");

System.out.println( )方法显示一个文本字符串，在本例中是 “I am Spartacus!”。
```


要确保你输入的内容和上面的一致。在输入时，源代码编辑器能识别你在做什么，并弹出有该 `System` 类相关的有用信息，即 `out` 实例变量和 `println()` 方法。你以后会爱上这一点，但是现在先将其忽略。

在确保输入的代码行没有错误之后，使用分号结尾，然后单击 **Save All Files** 工具栏按钮，来保存该类。

在运行 Java 类之前，必须先将其编译为可执行的字节码。NetBeans 会尝试自动编译类，也可以使用两种方式来手动编译类：

- 选择菜单命令 **Run->Compile File**。
- 在 **Project** 面板中右键单击 `Spartacus.java`，在弹出的菜单中选择 **Compile File**。

如果 NetBeans 不允许你选择这两种方式，则意味着它已经自动编译了类。

如果在编译类时失败，则 **Project** 面板中靠近文件名 `Spartacus.java` 的位置会出现一个红色的惊叹号。为了修复该错误，请将你在源代码编辑器中输入的内容与程序清单 A.1 中列出的源代码进行比较，如果没有问题，再次保存。

程序清单 A.1 Spartacus.java 类

```

1: /*
2:  * To change this template, choose Tools | Templates
3:  * and open the template in the editor.
4:  */
5:
6: /**
7:  *
8:  * @author User
9:  */
10: public class Spartacus {
11:
12:     /**
13:      * @param args the command line arguments
14:      */
15:     public static void main(String[] args) {
16:         // TODO code application logic here
17:         System.out.println("I am Spartacus!");
18:
19:     }
20:
21: }
```

该类在第 10~21 行中定义。第 1~9 行是 NetBeans 在每一个新类中都会添加的注释。

A.4 运行应用程序

在创建并成功编译 `Spartacus.java` 类之后，可以使用两种方式在 NetBeans 内运行。

- 从菜单中选择 **Run->Run File**。
- 在 **Projects** 面板中右键单击 `Spartacus.java`，然后在弹出的菜单中选择 **Run File**。

当运行 Java 类时，其 `main()` 方法将被编译器调用。字符串 “I am Spartacus!” 将显示在

Output 面板中如图 A.4 所示。

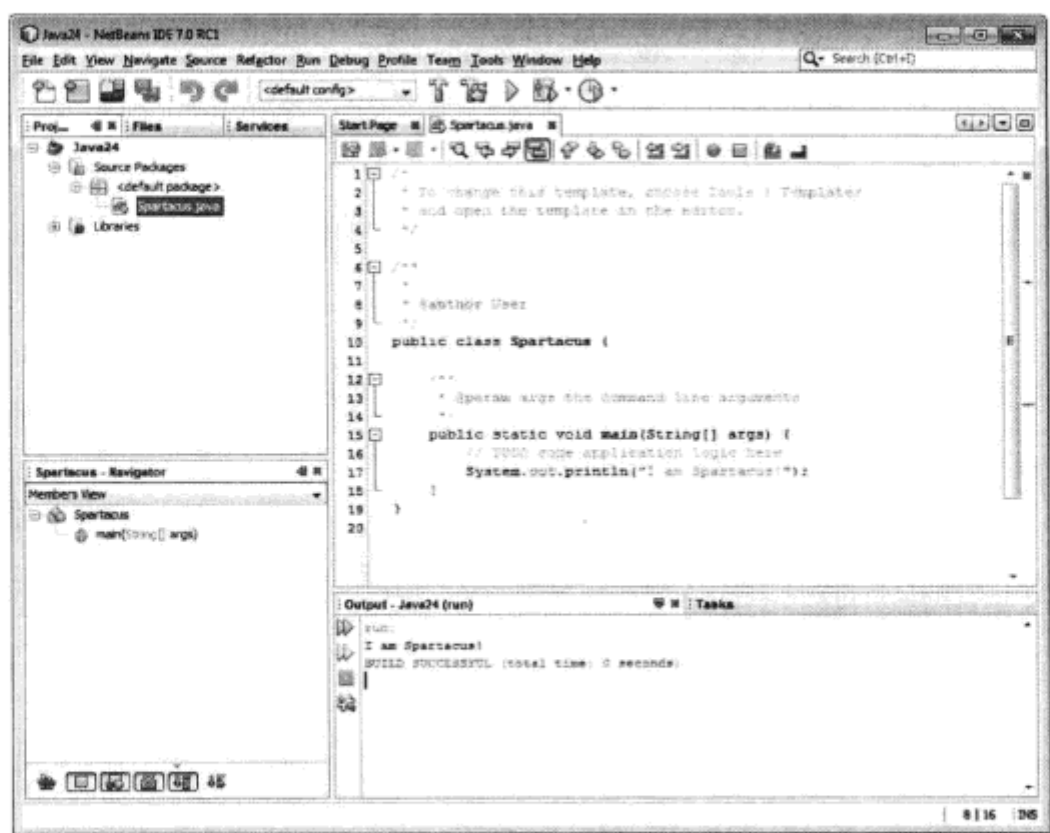


图 A.4
Spartacus 应用程序的输出

Java 类要想运行，则必须有一个 `main()` 方法，如果试图运行一个不包含该方法的类，则 NetBeans 会报错。

A.5 修复错误

Spartacus 应用程序已经编写、编译、运行完毕。现在，我们来看一下，当程序运行错误时，NetBeans 会如何响应。

与其他程序员一样，你以后会通过大量的实践来提升自己的错误修复能力，但此时还是请多加注意。

返回源代码编辑器中的 `Spartacus.java`，然后删除调用 `System.out.println()` 方法的那一行（程序清单 A.1 中的第 17 行）代码后面的分号。甚至在你保存该文件时，NetBeans 都会报错，并在相应行的左边显示一个红色的警告图标（见图 A.5）。

将鼠标放到该警告图标上，会出现一个对话框，该对话框描述了 NetBeans 发现的错误。

NetBeans 源代码编辑器能够识别大多数常见的编程错误和输入错误，这些错误通常出现在编写 Java 程序时。它将组织该文件被编译，直到错误被排除。

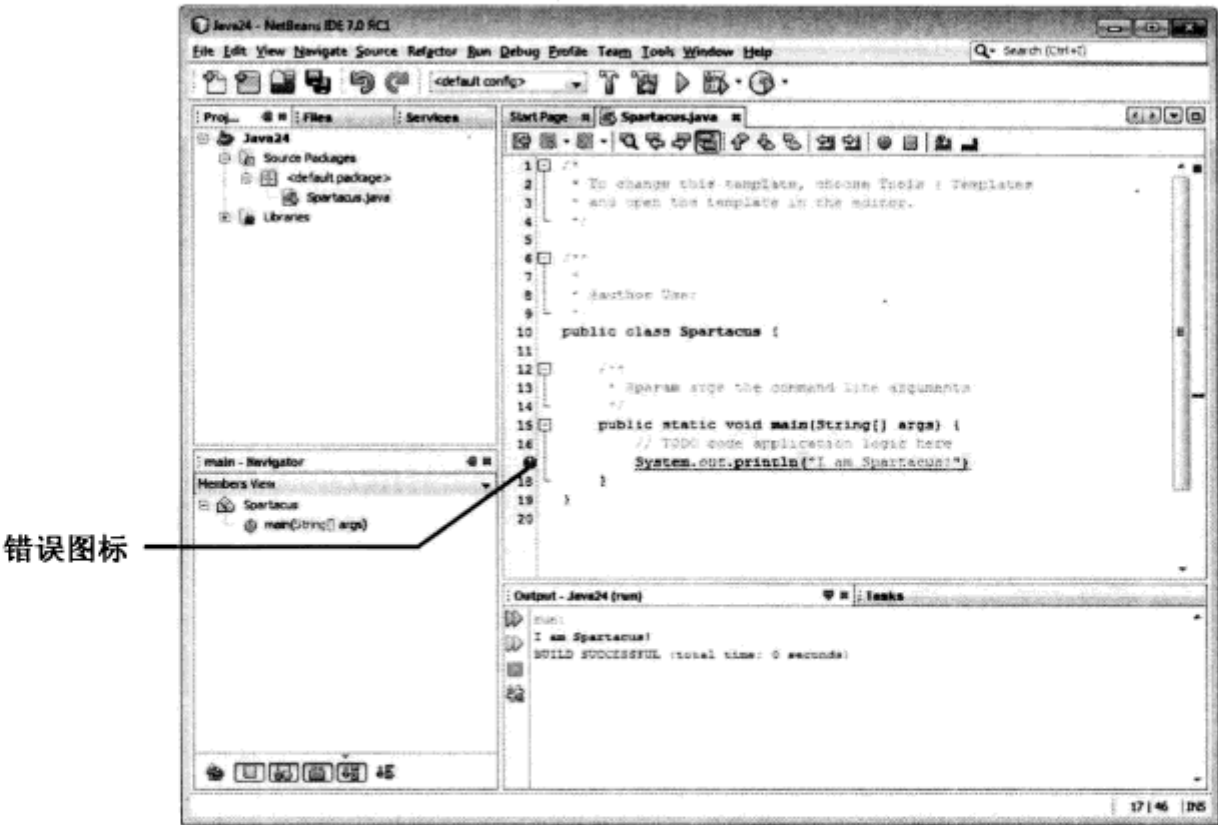
将分号放回代码原处后，则错误图标消失，此时可以保存并运行该类。

在创建和编译本书中的 Java 程序时，将会用到这些基本的特性。

除了这里提到的这些特性之外，NetBeans 还有很多其他特性，但是在深入学习 NetBeans 之前，应该重点关注 Java 的学习。因此在刚开始使用 NetBeans 时，最好将其当成一个简单的项目管理器和文本编辑器。然后利用它编写类、标记错误，以确保能成功编译和运行每一个项目。

当你准备深入学习 NetBeans 时，可以查看 Oracle 提供的培训和文档资源，地址为 www.netbeans.org/kb。

图 A.5
在源代码编辑器中
标记错误



附录 B

Java 资源

结束本书的学习之后，你可能想知道如何进一步提高 Java 编程技能。本附录列出了一些图书、Web 站点、Internet 讨论组和其他资源，你可以通过它们来丰富 Java 知识。

B.1 可以考虑的其他书

Sams 出版社和其他出版社出版了一些与 Java 编程相关的图书，其中有些是对本书内容的进一步深入。这些图书如下所示。

- Sams Teach Yourself Java in 21 Days, by Rogers Cadenhead, ISBN 0-672-33574-3。尽管本书前半部分的某些章节看似多余，但是它对 Java 进行了详细讲解，并添加了很多高级主题。如果你想使用另外 504 个小时来学习 Java，则本书无疑很合适。
- The Java EE 6 Tutorial: Basic Concepts, Fourth Edition, by Eric Jendrock and others, ISBN 0-13708-185-5。该书讲解了 Java Enterprise Edition (JEE) 的相关知识，JEE 是 Java 类库的一个扩展，主要在大型计算环境中的大型企业中使用。
- Java Phrasebook, by Timothy R. Fisher. ISBN 0-67232-907-7。该书囊括了 100 多个 Java 代码案例，这些代码是由专业程序员和 Java Developer's Journal 杂志开发的，你可以将其用在自己的 Java 程序中。
- Agile Java Development with Spring, Hibernate and Eclipse by Anil Hemrajani。该书以 Java Enterprise Edition 为讲解主体，向读者展示了如何使用 Spring 框架、Hibernate 库和 Eclipse IDE 来降低其企业应用编程中的复杂度。

读者可以从 www.informit.com 上免费下载摘自 Java 学习资源，这些资源来自于 Sams 出版社出版的其他 Java 图书。

读者从 Sams 出版社的站点 www.informit.com/sams 可以找到 Sams 将要出版的其他图书。

B.2 Oracle 公司的 Java 官方站点

Oracle 公司的 Java 软件部维护了 3 个网站，Java 用户和 Java 程序员可能会对这些网站感兴趣。

查找有关 Java 的信息时，应首先访问网站 <http://www.oracle.com/technetwork/java>。从这里可下载最新的 JDK 和其他编程资源以及完整的 Java 类库文档；另外还有 bug 数据库、用户组目录和支持论坛。

网站 www.java.net 是一个 Java 程序员大型社区。你可以编写与 Java 语言相关的博客，创建一个新的开源项目，并放到上面与大家共享，你还可以通过该站点与其他程序员进行合作。

网站 www.java.com 旨在让 Java 语言给消费者和非程序员带来更大好处。你可以从这个网站下载 Java 运行环境，以便让用户在自己的计算机中运行使用 Java 语言开发的程序。此外，这里还有一个展览室，它通过 Java 示例来向用户显示 Java 在当今世界中的用途。

B.2.1 Java 类文档

在 Oracle 公司的 Java 网站中，最有用的可能是关于 Java 类库中每个类、变量和方法的文档。数千页的在线免费材料演示了如何在程序中使用这些类。

要查看 Java 7 的类文档，请访问 <http://download.oracle.com/javase.7/docs/api>。

B.3 其他 Java 站点

随着 Java 在 Web 页面上的应用，Java 创造了巨大的奇迹，因此出现了大量专门介绍 Java 和 Java 编程的网站。

B.3.1 本书英文版的配套网站

本书英文版的官方网站为 www.java24hours.com，附录 C 将对该网站做详细介绍。

B.3.2 Café au Lait

这是 Elliotte Rusty Harold（编写了多本关于 Java 编程的优秀图书）管理的一个网站，它包含 Java 新闻、产品发布以及程序员感兴趣的其他站点。对于那些对 Java 感兴趣的人来说，该网站是一项宝贵的资源，其网址为 www.cafeaulait.org。Harold 还提供了一个有关 Java 的常见问题列表。自从他开始检修该网站起，就很少再对其更新，在本书编写之时，该站点可能已经重新运行。

B.3.3 Workbench

我经常通过博客 Workbench 讨论 Java、Internet 技术、计算机图书，以及其他相似的主题，其网址为 <http://workbench.cadenhead.org>。

B.3.4 Java 7 Developer Blog

Java 开发人员 Ben Evans 和 Martijn Vrburg 一直通过他们的 Java 7 Developer Blog 来跟踪 Java 7 的进展，其地址为 www.java7developer.com。该网站中有用来演示 Java 当前版本新特性的代码示例，以及有效使用当前版本的提示，同时还讨论了可能会出现在 Java 8 中的新特性。

B.3.5 其他 Java 博客

还存在几百个与 Java 编程相关的博客，其中有些博客主要以 Java 编程为主，有些则不是。搜索引擎 IceRocket 在 www.icerocket.com/tag/java 站点提供了与 Java 有关的最新博客列表。

B.3.6 InformIT

InformIT 是一个技术参考网站，是 Sams 出版社组建的一个综合性网站，其网址为 www.informit.com。该网站涵盖了十几个与软件开发和 Internet 相关的主题。InformIT 的 Java 社区包括 “How-to” 文章和初学者指南。

B.3.7 Stack Overflow

在线社区 Stack Overflow 是一个程序员可以提供问题，并对其他用户的答案进行评价的地方。该网站为 tagged 类型，因此在搜索时，可以将搜索缩窄为感兴趣的语言或主题。要查看 Java 相关的问题，请访问 <http://stackoverflow.com/question.tagged/java>。

B.3.8 Java Review Service

对网上发布的新程序、组件和工具进行评论；评级为 “Top 1%”、“Top 5%”、“Top 25%” 等。资源按主题分类，包含对资源的评价以及下载源代码（如果有的话）的链接。其网址为 <http://www.jars.com>。

B.3.9 JavaWorld 杂志

自 Java 语言诞生之初，就有了该杂志，它经常发表讲座式文章、Java 进展新闻及其他专题，它还有视频和音频播客。它的网址为 www.javaworld.com。

B.3.10 Developer.com's Java Directory

由于 Java 是一种面向对象语言，因此很容易在自己的程序中使用他人创建的资源。开发重要的 Java 项目前，应在网上查找可在程序中使用的资源。

一个不错的地方是 Developer's Java Directory。该网站对 Java 程序、编程资源和其他信息进行分类和编目，网址为 www.developer.com/java。



附录 C

本书站点

阅读本书后，读者肯定有不太明白的地方，虽然作者不希望如此。

编程是一种专业性很强的技术，包含奇怪的概念和术语，如实例化、三元运算符以及高位优先字节序和低位优先字节序等。

如果读者对本书介绍的任何主题还有不清楚的地方，请参阅本书英文版配套网站，网址为 www.java24hours.com（见图 C.1）。

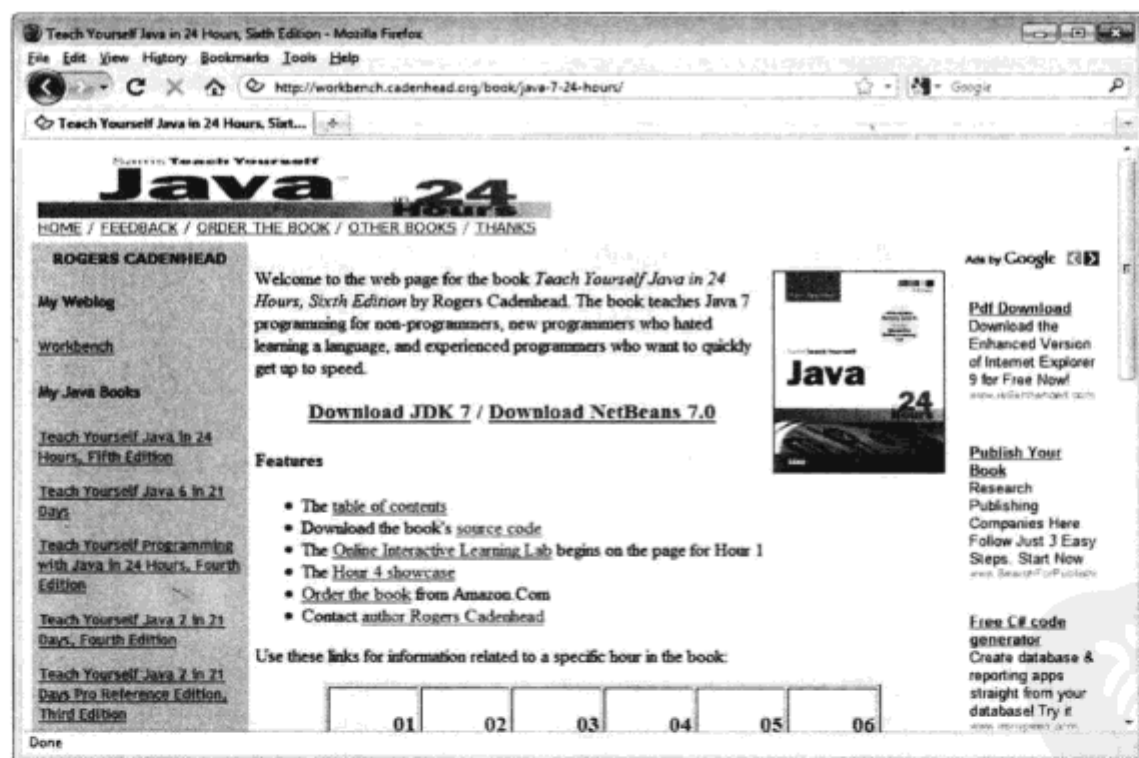


图 C.1

该书的配套网站

该网站提供了以下内容。

- 勘误和说明：作者发现本书中的错误后，将在该网站上进行说明——提供正确的内容和其他有帮助的材料。
- 解答读者提出的问题：如果读者提出的问题没有包含在本书的“问与答”中，我将把它放到这个网站上。
- 本书所有程序所需的源代码、类文件和资源。

- **Java 程序示例：**在该网站上可以找到本书提到的某些程序的工作版本。
- **每章最后的练习答案，**包括源代码。
- **本书提到的网站的最新地址：**如果本书提到的某个网站的地址发生变化，且我知道这个新的 URL，我会将其放在网上。

读者也可以通过访问本书英文版的配套网站来给作者发 E-mail。单击 Feedback 链接将打开一个页面，在该页面中可以直接给作者发 E-mail。

请读者随意发表看法，无论是积极的、消极的、冷淡的、模糊的、感兴趣的……

——Rogers Cadenhead



附录 D

设置 Android 开发环境

尽管 Android app 是使用 Java 语言开发的，但是在开发时仅使用标准的 Java 编程工具还不够，还需要 Java 开发工具包（JDK）、Android 开发工具包，以及为 Android 编程量身打造的集成开发环境和 Android 设备的驱动程序。

Eclipse 是用来开发 Android app 的最流行的 IDE，而且对 Android 的支持最好。

在该附录中，你将会设置这些工具，并确保它们能够一起工作，以运行 Android app。这些工具都是免费的，而且可以从 Internet 上下载。

D.1 起步

你可以在如下操作系统上进行 Android 编程：

- Windows XP 或后续操作系统；
- Mac OS X 10.5.8 或后续操作系统（x86）；
- Linux。

为了安装 Android SDK，你需要有 600MB 的磁盘空间，而安装 Eclipse IDE 则需要 1.2GB。

此时，你应该安装了 Java 开发工具包，因为本书中的程序都是使用 Java 开发工具包和 NetBeans 开发的。开发 Android app 需要 JDK 5.0 或更高版本。

如果出于某种原因，你仍然需要 JDK，则可以从 <http://oracle.com/technetwork/java/javase> 上下载。

D.2 安装 Eclipse

尽管其他 IDE（比如 NetBeans）也支持 Android 开发，但是 Eclipse 已经成为编写 Windows app 的最常用的工具。Android 开发人员将 Eclipse 作为他们首选的开发环境，而且他们的官方文档和教程中使用的也是 Eclipse。

Eclipse 与 NetBeans 一样，都提供了编写 Java 程序的图形用户界面。你可以使用它来创建任何类型的 Java 程序，而且它还支持其他编程语言。

Android 需要的 Eclipse 版本为 3.5 或更高版本。

你可以从 <http://eclipse.org/downloads> 下载 Eclipse。

Did you know?

注意:

比较常见的 Android 编程教程中使用的也是 Eclipse, 比如 Sams Teach Yourself Android Application Development in 24 Hours, Second Edition (Lauren Dercey 和 Shane Conder 编写)。由于在本附录设置的工具也可以用在上面提到的这本 Android 开发图书中, 因此你可以在学习完本书后接着学习这本 Android 开发图书。

Eclipse IDE 有多个版本, 请选择用于 Java EE 开发的 IDE。Java EE 是 Java 企业版, 它包含两个可以在 Android 编程中使用的工具: Java 开发工具 (Java Development Tools, JDT) 插件和 Web 工具平台 (Web Tools Platform, WTP)。

Eclipse 被打包为 ZIP 压缩文件。在计算机中安装 Eclipse 时, 没有安装向导。该 ZIP 文件包含一个顶级的 eclipse 文件夹, 其中存放了运行 Eclipse 所需要的所有文件。

将该 ZIP 文件解压缩到其下载文件夹中。在我的 Windows 系统中, 我将它放在了 Program Files (x86) 文件夹中。

解压缩之后, 进入到 eclipse 文件夹, 查找可执行的 Eclipse 应用程序。创建该应用程序的一个快捷方式, 就其放到菜单中, 或者是方便运行该程序的其他位置, 比如桌面或任务栏中。

在启动 Eclipse 之前, 还需要安装 Android SDK。

D.3 安装 Android SDK

Android SDK 是一个免费的工具集, 可以用来创建、调试和运行 Android 应用程序。在编写 Android app 时, Eclipse 将会使用该 SDK。

你可以从 Android 官方网站下载该 SDK, 其地址为 <http://developer.android.com/sdk>。它有对应的 Windows 版本、Mac OS 版本和 Linux 版本。

在安装 Android SDK 时, Windows 版本会有安装向导。其他版本在本书编写之时只是 ZIP 文档 (Mac OS) 或者是 TGZ 文档 (Linux)。

无论使用安装向导还是可以处理这些压缩文档的程序, 在处理完毕之后将其放到存放程序的文件夹中: 应该是放置 Eclipse 文件夹的父文件夹。在我的计算机中, 我将它放在了 Program Files (x86)。

SDK 包含一个 SDK 和 AVD 管理器, 可以用来升级和增强 SDK (前提是 SDK 已经安装了)。

可以通过 Eclipse 的菜单命令来运行该管理器, 这可以轻松地让当前的 SDK 与 Android 先发布的版本保持一致。

在安装了 SDK 之后, 即可准备运行 Eclipse。

D.4 安装在 Eclipse 中使用的 Android 插件

Eclipse IDE 支持多种编程语言和技术，但是只有在安装了相对应的插件之后，才能真正地提供支持。

Eclipse IDE 中除了集成 Android SDK 外，还需要集成一个插件，这个插件在 IED 界面中添加了一个与 Android 相关的菜单命令，这样也就可以创建和管理 Android app 了。

步骤如下。

1. 使用先前创建的快捷方式来启用 Eclipse，也可以通过打开 Eclipse 的安装文件夹，运行其中的可执行程序来启用 Eclipse。该程序将载入几个窗口，以及一个菜单栏和工具栏，它们运行在窗口的顶部。
2. 选择菜单命令 **Help->Install New Software**，打开 Install Wizard，通过它可以找到并安装 Eclipse 使用的插件。插件是从软件库（software repositories）中下载的，因此在 Eclipse 找到插件之前，必须知道软件库的位置。
3. 单击 **Add** 按钮，打开 **Add Repository** 对话框。
4. 让 **Name** 文本框为空，在 **Location** 文本框，输入地址 <http://dl-ssl.google.com/android/eclipse/>，然后单击 **OK**。此时 **Install** 窗口中应该出现一个 **Developer Tools** 条目，如图 D.1 所示。

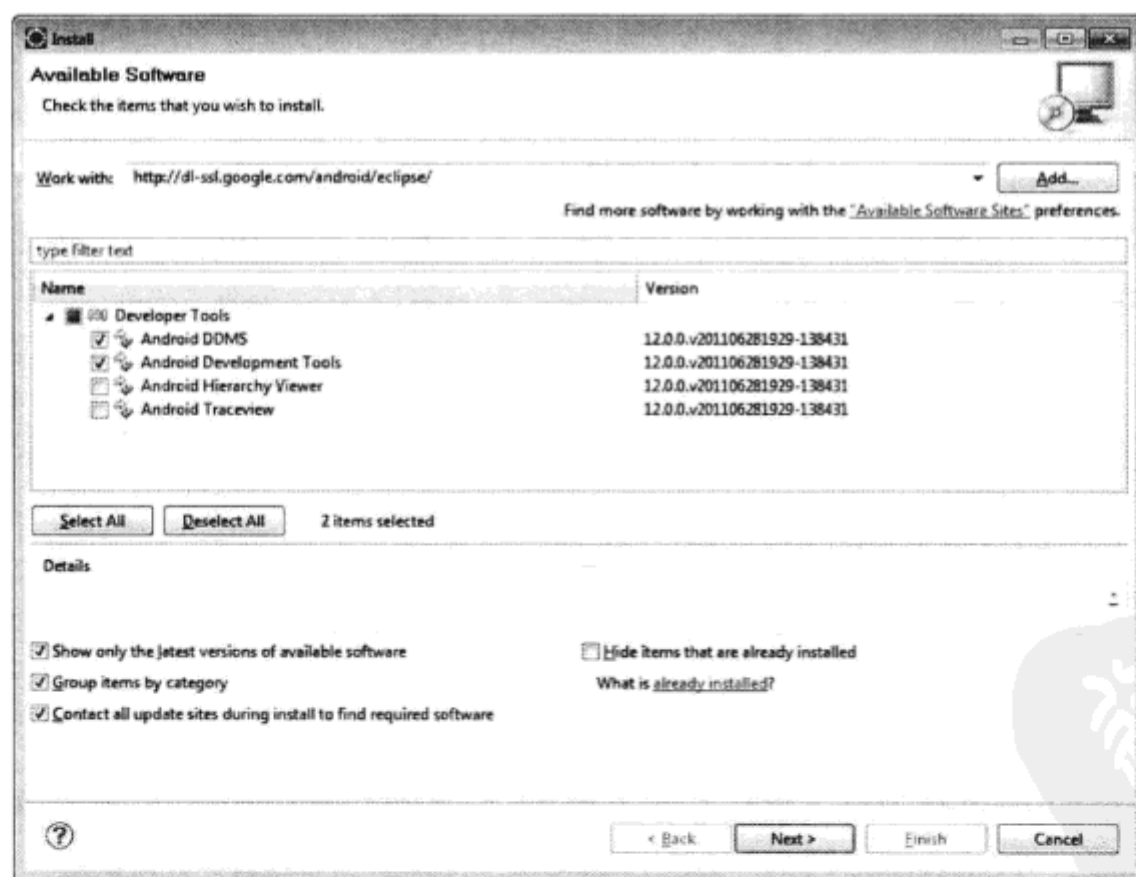


图 D.1

在 Eclipse 中添加新的插件

5. 单击该条目旁边的箭头，将其展开。将看到几个子条目，它们是与 Android 相关的工具，可以将其添加到 Eclipse 中，如图 D.1 所示。
6. 选择 **Android DDMS** 和 **Android Development Tools** 复选框（也可以添加其他工具，比如 **Android Hierarchy Viewer**，但是在开始 Android 编程时，不会看到它们）。
7. 单击 **Next** 来查看许可协议，并检查是否还需要安装其他东西。当运行到该向导的最

后一个窗口时，单击 Finish。

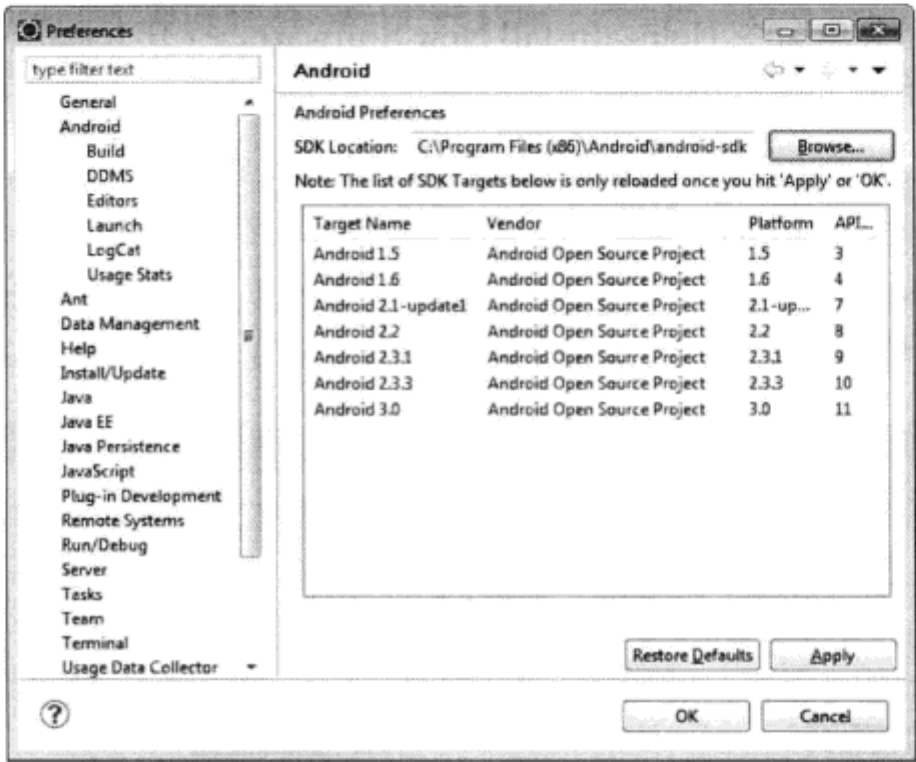
在安装插件之后，关闭 Eclipse，然后再次启动。

必须检查 Eclipse 中的首选项，以确保该 IDE 可以找到 Android SDK。

为此，请采取如下步骤。

1. 选择命令菜单 Window->Preferences，打开 Preferences 对话框，该对话框的左侧是一个分类列表。
2. 单击 Android，查看通用的 Android 首选项。
3. 确保 SDK Location 文本框中的地址是 Android SDK 所在的文件夹。如果不是，单击 Browser 按钮，导航到该文件夹并选中。
4. 在定位了 SDK 之后，将会看到一个 SDK 目标列表，如图 D.2 所示。

图 D.2
在 Eclipse 中设置
Android 首选项



这些目标是你使用 SDK 创建的 app 所运行的 Android 版本。在创建 Android app 时，为了保证它可以运行，必须为其指定最早的 Android 版本。

单击 OK 来关闭该对话框并保存这些首选项。

在安装了 Android 插件并且定位到 SDK 之后，应该能在 Eclipse 中看到一个新的菜单命令。其中一个 Window->Android SDK and AVD Manager。

如果没有找到该命令，则单击 Eclipse 并重新启动。

你可以使用管理器来让该 SDK 保持为最新版。单击 Window->Android SDK and AVD Manger，打开该管理器。

在该管理器的左侧面板中单击 Installed Packages，查看计算机中安装了哪些 SDK 组件。

单击 Available Package，查看哪些组件可以使用，但是还没有安装。

选中其中一个包，Eclipse 可以查看该包中的内容，并列出需要安装的其他内容，如图 D.3 所示。

选择想要安装的包并取消选中不需要安装的包之后，单击 Install Selected。

你应该定期检查，以进行更新。随着新的 Android 手机和其他设备投放到市场中，Android

的发展异常迅猛，因此 SDK 必须对其提供支持。

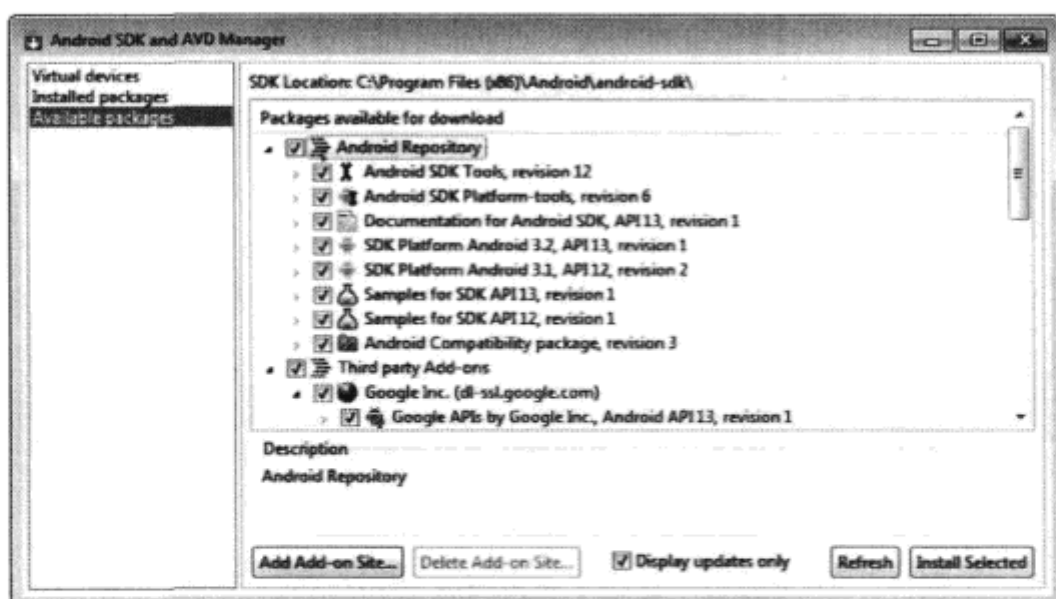


图 D.3

安装 Android SDK 中的新包

D.5 设置你的手机

Android SDK 包含一个模拟器，它的行为与 Android 手机相似，可以运行你创建的 app。当你编写 app 时，这会带来极大的方便，因为尽管你可以让你的 app 在运行在测试环境中，但是某些时候，你还需要查看一下它是如何在真实的 Android 手机（或其他设备）上运行的。

通过计算机的 USB 连接，可以将使用 SDK 编写的 app 部署在 Android 设备中。USB 连接也可以用来向设备传输文件，或者将设备中的文件传输到计算机。

在连接 USB 线缆之前，必须执行如下步骤，以在手机上启用 USB 调试。

1. 在手机的主界面选择 Menu->Settings，打开 Settings app。
2. 选择 Applications->Development，然后单击 USB 调试盒。

在其他设备中，该选项可能位于 Settings 的其他位置，其名字可能为 USB 连接模式、USB 调试或其他名字。Android 站点 <http://developer.android.com> 中有一个文档，它讲解了如何为不同的 Android 设备设置该选项。

将 USB 线缆的一端连接到计算机，另一端连接到你的手机。则在设备的顶部会出现一个像 bug 一样的小 Android 图标。该图标与信号强度图标和电池图标相邻。

向下拖动顶部栏，将看到 USB Debugging Connected 和 USB Connected 消息（见图 D.4）。

此时，手机设置完毕，但是你的计算机也需要一定的配置，才能连接到该设备。如果之前从来没有通过 USB 线缆将手机连接到计算机上，则查看你的手机说明书，以获悉其操作方法。你可能需要从随手机销售的 CD 中安装驱动程序，或者是从手机生产商的网站上下载该驱动程序。

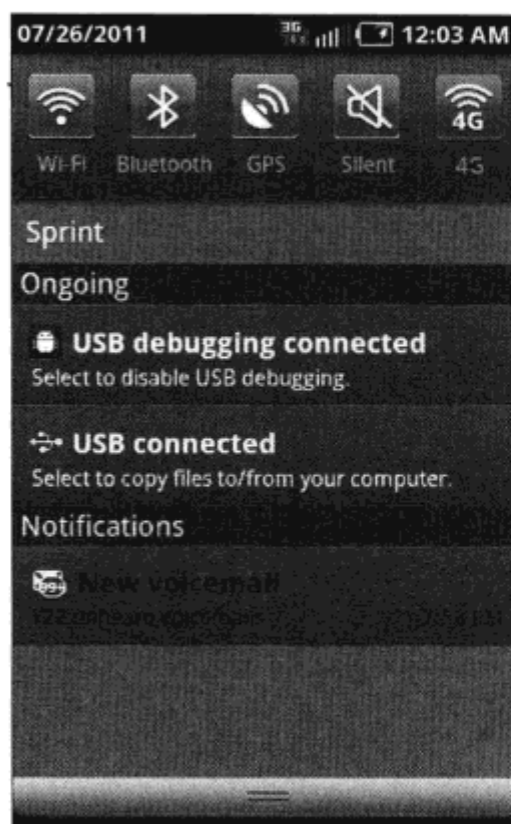
在 Windows 中，Android SDK 和 AVD 管理器在 Eclipse 内运行，你可以使用它来下载 USB 驱动程序包，这是一个驱动程序集，可以用于不同的手机和其他设备，同时还可以下载其他与设备相关的包。选择 Window->Android SDK and AVD Manager，查看哪些包可以使用。

在第 24 章，我们使用 Android 开发工具来创建和运行 Android app。如果所有的设置没

有问题，则它可以成功地运行在模拟器和 Android 手机上。

图 D.4

在 USB 调试模式中
使用 Android 手机



[G e n e r a l I n f o r m a t i o n]

书名= J A V A入门经典 第6版

作者=

s s号= 1 2 9 2 8 2 4 6

I S B N号=

出版社=

出版日期=

起始页= 1

页数= 2 9 2

读秀号=

u r l = h t t p : / / b o o k . d u x i u . c o m / b o o k D e t a i l . j s p ? d x

N u m b e r = 0 0 0 0 0 8 2 4 0 7 4 5 & d = 4 E F 1 6 1 3 C 1 F A 0 4 D 0 1 3 7 E 7 2

0 4 E 1 E F C E 2 6 4 & f e n l e i = 1 8 1 7 0 4 0 3 0 2 & s w = j a v a

封面
书名
版权
前言
目录
正文