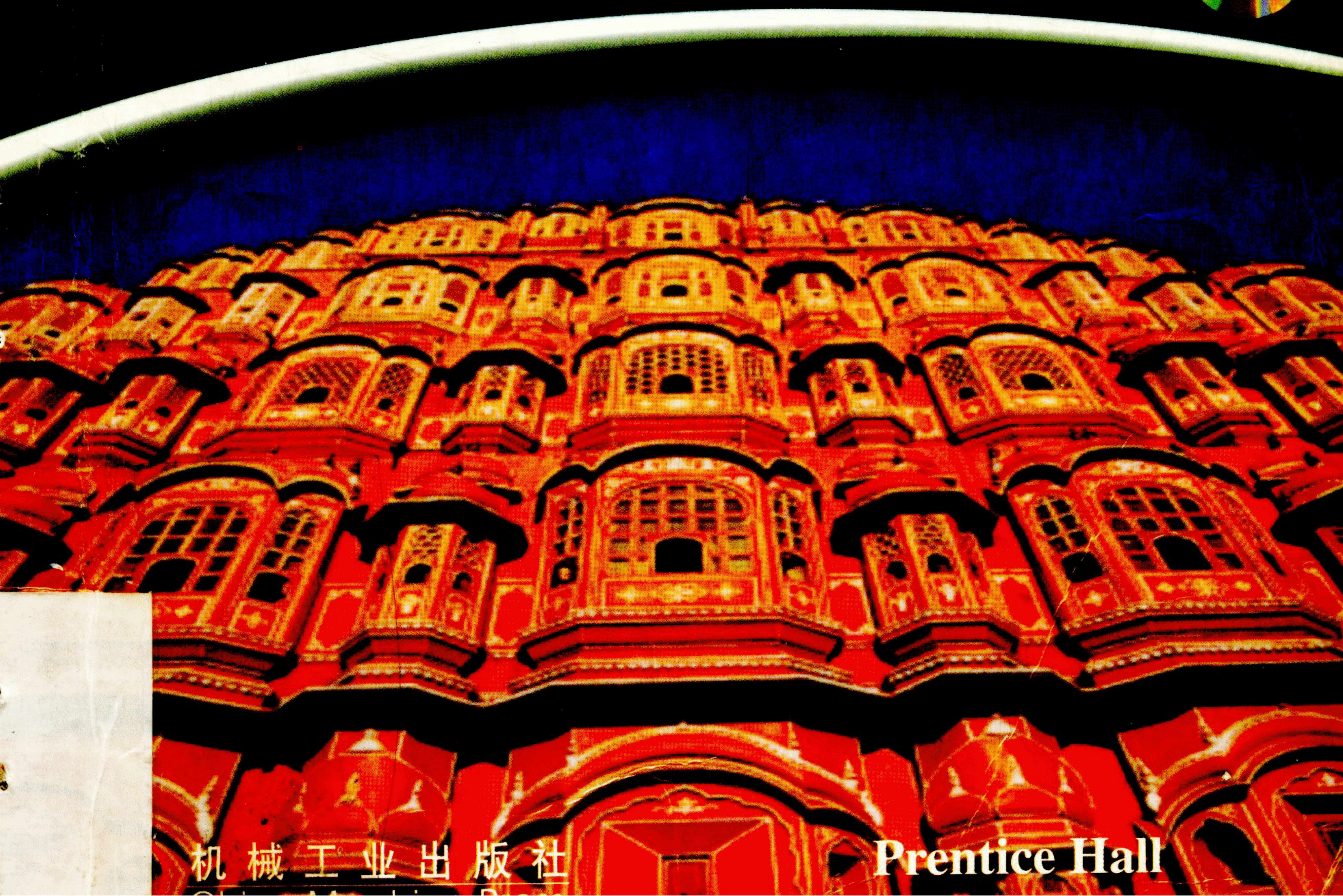
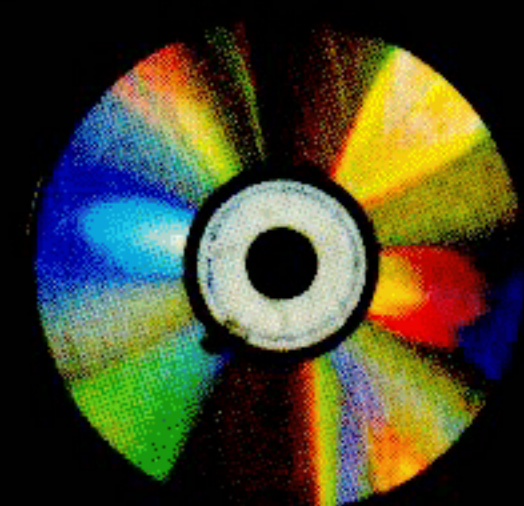


Java 2 图形设计

卷 I: AWT

Graphic Java 1.2 Mastering the JFC Volume I: AWT
(3rd edition)

(美) David M. Geary 著 马欣民 等译



机械工业出版社

Prentice Hall

软件开发技术丛书

Java 2 图形设计

卷 I : AWT

(美) David M. Geary 著

马欣民 等译

王 英 校



机械工业出版社
China Machine Press

本书详细介绍 Java 中的 AWT (抽象窗口工具集)。主要内容包括同位体、轻量构件、剪贴板与数据传输、颜色、无鼠标操作、打印、布局管理器、国际化、串行化、滚动性、字体与字号以及 AWT 中的各种构件。

David M. Geary: Graphic Java 1.2 Mastering the JFC Volume I: AWT (3rd edition)

Authorized translation from the English language edition published by Prentice Hall.

Copyright © 1999 by Sun microsystems Press.

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2000 by China Machine Press.

本书中文简体字版由美国 Prentice Hall 公司授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号: 图字: 01-1999-2352

图书在版编目 (CIP) 数据

Java 2 图形设计: 卷I, AWT/ (美) 吉瑞 (Geary, D. M.) 著; 马欣民等译. —北京: 机械工业出版社, 2000.1

(软件开发技术丛书)

ISBN 7-111-07745-8

书名原文: Graphic Java 1.2 Mastering the JFC Volume I: AWT, 3rd edition.

I. J... II. ①吉... ②马... III. JAVA 语言-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (1999) 第 55437 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 陈剑瓯

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2000 年 1 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 41 印张

印数 0 001 - 5000 册

定价: 79.00 元 (附光盘)

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

前 言

Java 语言可以说是到目前为止推广最快的一种计算机语言，自从 1990 年 SUN 公司开始设计，到现在在全球流行，也只不过是短短的几年时间。Java 语言是从 C++ 中衍生而来，它继承了 C++ 的大量语言成分，同时抛弃了 C++ 中冗余的和容易引起问题的成分，Java 集面向对象、与平台无关、稳固性、安全性、多线程性等多种特征于一身，是一种适合于分布式计算的新型面向对象程序设计语言。

JDK 2 与以前版本的 JDK 相比，在开发图形用户界面、图形 applet 以及应用程序等各个方面都提供了更加强大的支持。在 JDK 2 中，用户完全可以利用 Java 基本类完成整个开发过程。Java 基本类由 4 个 API 组成，分别为 AWT、Swing、辅助 API 以及 2D API。

其中，抽象窗口工具集（AWT）是 Java 最基本的用户界面工具集，它不仅提供标签、按钮以及滚动条等基本构件集，而且还提供授权事件模型、布局管理器、数据传输支持以及轻量构件支持等大量图形构件的基础结构。

Swing 是基于 AWT 基本结构创建的二级用户界面工具集。与 AWT 相比，Swing 提供了更加丰富的构件集。Swing 共提供 40 个构件，是 AWT 构件集的 4 倍。另外，Swing 中的标签以及按钮等构件设计可用于取代 AWT 中的相应构件，而树与表等高级构件则可以提供 AWT 不支持的其他功能。

辅助 API 允许用户开发伤残者也可以顺利使用的 Java applet 以及应用程序，而且还是用于移动用户界面开发的强有力的可选工具。例如：如果 Java 导航系统可以按照声音方式（而不是按照地图显示方式）提醒司机当前车辆所在的位置等信息，则可以极大程度上方便司机用户的使用。

2D API 包含许多重要的对 AWT 图形模型的改进，例如：颜色、形状、图像以及文本管理等。

本卷的目的

JDK 2 中提供了 Swing 构件集，其中包含了一个完整的轻量构件集，可用于替代 AWT 中的重量构件。由此，用户普遍会产生 Swing 是 AWT 替代物的误解，而实际的情形并不是这样的。实际上，Swing 是在 AWT 之上创建的，而且每个 Swing 构件都是一个真正的 AWT 构件。因此，为了准确把握 Swing 构件集的工作方式，读者首先必须掌握 AWT 构件的行为及其工作原理。

本卷正是基于这样的背景而产生的，其最重要的目的是帮助用户掌握 AWT，它将详细介绍 AWT 的各种基础概念与高级概念。当然，本书不仅提供了每个 AWT 构件的详尽描述，而且还提供了 AWT 的一些技巧，并确实做到面面俱到。

本卷的内容

通过对本卷的阅读，读者不仅可以全面掌握 AWT 的设计思想，而且还可以掌握如何才能使这种设计思想在应用程序开发过程中发挥最佳效应。

本卷主要介绍 AWT 中的基础概念与高级概念，而没有介绍 AWT 软件包中包含的其他 JFC

API, 即 2D API。如果需要了解 2D API 的详细内容, 读者可参阅本套书中的第 3 卷。以下是本卷中涉及的主要内容:

同位体

读者可了解到 AWT 中的同位体结构以及同位体方法的利弊。

剪贴板与数据传输

介绍了 AWT 的数据传输模型, 以及如何使用系统剪贴板与本地剪贴板等工具实现数据传输操作。虽然 AWT 构件只对字符串的传输提供直接的剪贴板操作支持, 但是利用 AWT 构件也可以实现其他类型数据的传输操作, 因此我们将以图形与自定义构件的传输为例, 介绍利用剪贴板实现其他类型数据传输的方法。

轻量构件

介绍如何创建自定义轻量构件、如何利用双缓冲容器技术拖曳自定义构件, 以及如何在演示区域内实现自定义轻量构件的动画效果等内容。通过相应内容的学习, 读者不仅可以掌握双缓冲的工作机制, 以及为什么应该将轻量构件放置在双缓冲容器内等问题, 而且还可以了解将轻量构件放置在容器中时常容易犯的错误, 以及轻量容器为什么必须手工实现与布局管理器的匹配等内容。

布局管理器

详尽介绍布局管理器的相关内容, 主要包括: GridBagLayout 以及如何创建自定义布局管理器等。同时, 还将介绍如何强制容器安排其包含的构件及其必要性。

国际化与串行化

介绍图形用户界面的国际化方法, 以及 AWT 构件及其事件监听者的串行化方法。

滚动性

介绍利用滚动框构件滚动容器内任意数量构件的方法、滚动框类的限制以及无同位体滚动框架存在的必要性等内容。同时, 我们还将讨论利用图形 Java 工具集创建无同位体滚动框架的方法, 用户在实际应用过程中, 可根据实际环境的需求更改该程序使其满足需要。

其他内容

讨论事件授权模型、颜色、无鼠标操作以及打印等内容, 另外还将详细介绍滚动条、菜单(包括弹出式菜单)、图像管理、图形、字体与字号、对话框以及文本框等构件。

读者对象

本书适用于利用 Java 开发面向对象程序的用户。由于关于 Java 语言细节及其如何与 VB、C、C++ 等语言组合使用的书籍相当多, 因此本书不讲解 Java 语言的细节问题。

互联网信息资源

与 Java 相关的在线资源很多。如果需要访问在线帮助信息，可访问 Sun 公司的主页：

<http://java.sun.com/>

如果希望查阅 Java 新闻信息，则可访问 Java 新闻组地址：

<comp.lang.java>

另外，互联网上还有供 Java 爱好者交换意见、提出问题以及解决方案的邮件列表，如果需要查看这些邮件列表，可访问网址：

<http://java.sun.com/mail.html>

从这些新闻组和网站，读者可以浏览其他许多与 Java 相关的资源、帮助信息、FAQ（常见问题及解答）以及在线杂志等信息。

另外，我们还特别设置了本书的更新信息网站，利用该网站同时还可以查阅 SunSoft 出版社的其他 Java 系列丛书的信息。该网站的地址为：

<http://www.sun.com/books/books/Geary/Geary.html>

本书英文原书书号为 ISBN 0-13-079666-2，当你访问以上网站时可能会用到这个书号。

最后，如果需要设计精美的 Java 图片，则可以访问网站：

<http://www.pixelsight.com:80/PS/pixelsite/pixelsite.html>

本书约定

为了进一步规范本书，我们在本书中使用了一系列的代码约定，如下所示：

约 定	示 例
类名称中每个单词的首字母大写	<code>public class LineOfText</code>
方法名称中第一个单词的首字母小写，其余单词的首字母大写	<code>public int getLength ()</code>
变量名中第一个单词的首字母小写，其余单词的首字母大写	<code>private int length</code> <code>private int bufferLength</code>
static 变量由下划线开头	<code>protected static int _ defaultSize = 2;</code>

注意 本书中提到方法和函数时一般都不带参数，但是当需要讨论这些参数的实际含义时，则将包含相应的参数信息。

参与本书翻译的人员有：马欣明、王兴军、孙克伟、朱家义、陈波、石鹏飞、石丽、于峰、张展等。他们都是多年从事 Java 编译器研究的人员，都十分精通和熟悉 Java 语言。

目 录

前言	
第一部分 入门	1
第 1 章 绪论	1
1.1 Java 基本类	1
1.2 抽象窗口工具集	1
1.3 同位体和平台独立	2
1.3.1 轻量构件	3
1.3.2 AWT 和 Swing 构件	3
1.3.3 AWT 和 2D API	3
1.4 构件: AWT 的基础	4
1.5 构件、容器和布局管理器	4
1.5.1 构件	4
1.5.2 基本构件工具	5
1.5.3 容器	5
1.5.4 布局管理器	7
1.6 小结	7
第 2 章 applet 和应用程序	8
2.1 Java applet	8
2.1.1 使用 appletviewer	8
2.1.2 浏览器的基础结构	9
2.1.3 java.awt.Applet 类	9
2.1.4 关键 Applet 方法	9
2.1.5 java.awt.Component 显示方法	11
2.2 Java 应用程序	11
2.2.1 设置应用程序	12
2.2.2 显示应用程序	13
2.3 组合 applet 和应用程序代码	13
2.4 小结	15
第二部分 图形和图像	17
第 3 章 图形	17
3.1 java.awt.Graphics	17
3.2 Graphics 参数	18
3.3 图形坐标系	20
3.3.1 绘制图形形状	20
3.3.2 绘制构件的四周边界	22
3.3.3 填充形状	22
3.4 Graphics 引用	23
3.4.1 引用副本的 Graphics 引用	23
3.4.2 Graphics 引用的寿命	25
3.4.3 处理 Graphics	26
3.5 绘制和填充形状	27
3.5.1 画直线	27
3.5.2 画折线	28
3.5.3 绘制矩形	29
3.5.4 画弧	33
3.5.5 绘制椭圆	35
3.5.6 绘制多边形	35
3.5.7 绘制文本	35
3.6 转换坐标系原点	36
3.7 剪贴	40
3.8 图形模式	42
3.9 创建图形	45
3.10 小结	48
第 4 章 颜色和字体	50
4.1 颜色模式	50
4.1.1 索引颜色模式	50
4.1.2 直接颜色模式	54
4.2 java.awt.Color 类	56
4.3 系统颜色	58
4.4 字体和字体度量	60
4.4.1 Font 类	60
4.4.2 FontMetrics 类	62
4.5 小结	68
第 5 章 加载和显示图像	70
5.1 Image 类和 Image 包	70
5.2 图像生产者 and 图像消费者	71
5.2.1 异步图像生产	71
5.2.2 ImageProducer	71
5.2.3 ImageObserver	72
5.3 加载和显示图像	72
5.4 applet 和应用程序之间的区别	75
5.5 等待图像加载	77
5.6 一次绘制图像的一行扫描线	78
5.7 MediaTracker	80
5.8 动画 GIF	82
5.9 AWT 构件作为图像观察者	85
5.10 创建图像	86
5.11 加载图像作为资源	87

5.12 小结	88	8.4 关于鼠标和按钮	160
第 6 章 图像过滤	89	8.5 监控鼠标事件	161
6.1 ImageProducer 接口	89	8.6 检测双击	163
6.2 ImageConsumer 接口	90	8.7 动作事件	164
6.3 图像生产者和图像消费者之间的关系	91	8.8 标识构件	166
6.4 AWT 图像过滤器	91	8.9 基于继承的事件模型的缺点	168
6.4.1 CropImageFilter	92	8.9.1 事件的传播	168
6.4.2 使用常规图像过滤器	94	8.9.2 继承的要求	170
6.4.3 ReplicateScaleFilter 和 AreaAveraging ScaleFilter	94	8.9.3 将事件处理插入到构件类中	170
6.4.4 组合图像过滤器	96	8.9.4 传送事件的责任	170
6.5 ImageConsumer 属性	98	8.9.5 handleEvent () 转换语句	170
6.6 实现自定义图像过滤器	99	8.10 小结	171
6.7 扩展 RGBImageFilter	100	第 9 章 授权事件模型	172
6.7.1 DissolveFilter	100	9.1 授权事件模型	172
6.7.2 DissolveEdgeFilter	103	9.1.1 构件、事件和监听者	172
6.8 扩展 ImageFilter	105	9.1.2 过滤事件	174
6.8.1 ImageFilter 类	106	9.2 结构概述	174
6.8.2 向图像过滤器传输图像位	107	9.2.1 事件	176
6.8.3 用于传输像素的 ColorModel	108	9.2.2 构件作为事件源	178
6.8.4 扩展 ImageFilter 的溶解过滤器	109	9.2.3 多点传送事件源	179
6.8.5 波形过滤器	113	9.2.4 单点传送事件源	179
6.9 实现 ImageConsumer 接口	119	9.2.5 事件源接口	179
6.9.1 图像溶解	119	9.2.6 监听者	179
6.9.2 Dissolver 类的实现	123	9.2.7 事件处理方法的 JavaBeans 设计 模式	181
6.10 双缓冲入门	128	9.3 AWT 适配器	181
6.11 小结	133	9.4 构件事件	183
第 7 章 无过滤图像处理	134	9.4.1 构件和容器事件	183
7.1 缩放和闪烁图像	134	9.4.2 焦点事件	185
7.2 抓取像素	137	9.4.3 键盘事件	187
7.3 内存图像源	140	9.4.4 鼠标和鼠标移动事件	190
7.3.1 使用 MemoryImageSource 裁剪图像	141	9.4.5 窗口事件	192
7.3.2 内存图像源和动画	142	9.4.6 画布和面板的焦点和键盘事件	193
7.4 小结	151	9.4.7 消耗输入事件	194
第三部分 事件和布局管理器	153	9.4.8 绘制事件	195
第 8 章 基于继承的事件处理	153	9.5 语义事件	195
8.1 旧 AWT 事件模型	153	9.5.1 动作事件	196
8.1.1 覆盖事件处理方法	153	9.5.2 可调事件	197
8.1.2 被传播的事件	154	9.5.3 项目事件	200
8.1.3 事件类型常量	154	9.5.4 文本事件	201
8.1.4 事件的向外传播	157	9.6 事件适配器	202
8.1.5 覆盖被传播的事件的处理	158	9.6.1 处理没有适配器的多点 事件源事件	203
8.2 事件修饰常数	158	9.6.2 类型安全的多路处理适配器	206
8.3 鼠标按钮事件	159	9.6.3 一般的多路处理适配器	208

9.7 内部类	211	10.6.1 GrdBagLayout 和 GridBag	261
9.7.1 三维按钮	212	Constraints	261
9.7.2 在单独的监听者类中封装事件处理		10.6.2 网格单元和显示区	263
代码	213	10.6.3 与构件比较的显示区	263
9.7.3 监听自己	215	10.6.4 GridBagConstraints.anchor	264
9.7.4 命名内部类	215	10.6.5 GridBagConstraints.fill	266
9.7.5 匿名内部类	216	10.6.6 GridBagConstraints.gridx 和 GridBag-	
9.7.6 修改默认的事件处理行为	217	Constraints.gridy	266
9.8 从自定义构件中激发 AWT 事件	218	10.6.7 GridBagConstraints.gridwidth 和 GridBag-	
9.9 从自定义构件中激发自定义事件	220	Constraints.gridheight	268
9.9.1 一个未经专门设计的方案	220	10.6.8 GridBagConstraints.weightx 和 GridBag-	
9.9.2 有关步骤	221	Constraints.weighty	269
9.9.3 开发自定义事件类	221	10.6.9 GridBagConstraints.insets	272
9.9.4 开发监听者接口	223	10.6.10 GridBagConstraints.ipadx 和 GridBag-	
9.9.5 为注册监听者定义接口	223	Constraints.ipady	272
9.9.6 开发可以激发自定义事件的自定义		10.6.11 GridBagLab	274
构件	224	10.6.12 GridBagLayout 和输入表单	274
9.10 调度事件和 AWT 事件队列	228	10.6.13 在嵌套的面板中布置构件	278
9.11 有效事件	230	10.6.14 嵌套面板之间的通信	281
9.12 基于继承的机制	232	10.6.15 GridLabApplet 的实现	282
9.13 事件处理设计	235	10.7 null 布局管理器	285
9.13.1 使用基于继承的事件模型	236	10.8 自定义布局管理器	290
9.13.2 监听自己	236	10.8.1 BulletinLayout	290
9.13.3 在单独的类中封装事件		10.8.2 运行 BulletinLayout 自定义布局管	
处理代码	237	理器	293
9.13.4 使用内部类	238	10.8.3 RowLayout	296
9.13.5 命名内部类与匿名内部类相比	238	10.8.4 运行 RowLayout 自定义布局管	
9.13.6 向容器传播事件	238	理器	302
9.14 小结	239	10.8.5 RowLayoutApplet 的实现	306
第 10 章 构件、容器和布局管理器	240	10.8.6 ColumnLayout	308
10.1 最大的三种 AWT 构件	240	10.8.7 运行 ColumnLayout 自定义布局管	
10.2 布局管理器	241	理器	311
10.2.1 两种类型的布局管理器	242	10.9 小结	314
10.2.2 布局管理器和容器空白区	243	第四部分 AWT 构件	315
10.2.3 同位体和空白区	244	第 11 章 AWT Component 类	315
10.2.4 布局管理器和构件首选尺寸	245	11.1 构件	315
10.3 绘制一个容器的构件	245	11.2 java.awt.Component	316
10.4 强制一个容器布置它的构件	248	11.3 构件属性	316
10.5 标准 AWT 布局管理器	252	11.4 不赞成的方法	317
10.5.1 使用何种布局管理器	253	11.5 构件的位置、边界和坐标	320
10.5.2 BorderLayout 布局管理器	254	11.6 构件的首选、最小和最大尺寸	320
10.5.3 CardLayout 布局管理器	255	11.7 构件的可见性和响应	320
10.5.4 FlowLayout 布局管理器	257	11.8 构件和同位体	320
10.5.5 GridLayout 布局管理器	259	11.9 显示构件	323
10.6 GridBagLayout 布局管理器	260	11.10 构件和 zorder	324

11.11 构件和光标	326	14.3 java.awt.TextArea	387
11.12 构件和串行化	330	14.4 小结	391
11.13 构件和国际化	333	第 15 章 滚动：滚动条与滚动框	392
11.13.1 Locale	333	15.1 java.awt.Scrollbar	392
11.13.2 资源包	334	15.2 java.awt.ScrollPane	403
11.13.3 SimpleH8Ntest applet	335	15.2.1 滚动构件	403
11.13.4 资源包属性	338	15.2.2 滚动图像	407
11.13.5 从国际化代码中分离出 GUI	338	15.2.3 程序控制滚动	411
11.13.6 可用地区和两个字符的编码	339	15.3 小结	415
11.14 构件和 JavaBeans	339	第 16 章 窗口、框架与对话框	416
11.14.1 约束属性	339	16.1 java.awt.Window	417
11.14.2 在自定义构件中实现 约束属性	342	16.1.1 快闪屏	418
11.15 构件和树锁定	344	16.1.2 提示框帮助	421
11.16 小结	348	16.2 java.awt.Frame	424
第 12 章 基本构件：标签、按钮、画布 和面板	349	16.3 java.awt.Dialog	426
12.1 标签和按钮	349	16.3.1 模式对话框与多线程	429
12.1.1 java.awt.Label	349	16.3.2 java.awt.FileDialog	432
12.1.2 java.awt.Button	352	16.4 小结	436
12.2 画布和面板	353	第 17 章 菜单	437
12.2.1 java.awt.Canvas	354	17.1 菜单类	437
12.2.2 java.awt.Panel	355	17.2 文件菜单	438
12.3 小结	357	17.3 处理菜单事件	439
第 13 章 项目选择：复选框、选择框和 列表	358	17.4 拖离菜单	441
13.1 复选框	358	17.5 MenuBar Printer	441
13.1.1 java.awt.ItemSelectable 接口	358	17.6 FrameWithMenuBar 类	442
13.1.2 java.awt.Checkbox	358	17.7 帮助菜单	445
13.1.3 相容的复选框	358	17.8 复选框菜单项	448
13.1.4 相互排斥的复选框	360	17.9 级联菜单	450
13.2 选择框和列表	362	17.10 动态更改菜单	452
13.2.1 是采用列表还是选择框	362	17.11 弹出式菜单	456
13.2.2 java.awt.Choice	363	17.11.1 弹出式菜单与构件	456
13.2.3 java.awt.List	369	17.11.2 处理弹出式菜单事件	458
13.2.4 双列表构件	372	17.11.3 显示与构件相关的弹出式菜单	460
13.3 小结	380	17.12 小结	462
第 14 章 文本构件	381	第 18 章 无鼠标操作与打印	463
14.1 java.awt.TextComponent	381	18.1 无鼠标操作	463
14.1.1 文本选择	382	18.2 键盘遍历	463
14.1.2 TextComponent 监听者	383	18.2.1 标准 AWT 构件与键盘遍历	463
14.2 java.awt.TextField	383	18.2.2 自定义构件及其键盘遍历	466
14.2.1 输入的有效性	384	18.3 菜单快捷键	470
14.2.2 退出有效性	384	18.3.1 菜单类与快捷键	471
14.2.3 过程中确认	386	18.3.2 菜单快捷键示例	471
		18.4 打印	473
		18.4.1 获取 PrintGraphics 的引用	474
		18.4.2 打印自身的 applet	474

18.4.3 打印对话框及其属性	476	21.2.5 事件	526
18.4.4 页码设置	478	21.2.6 DragSourceDragEvent 与 DragSource DropEvent	526
18.5 小结	478	21.2.7 DropTargetDragEvent 与 DropTarget DropEvent	527
第 19 章 轻量构件	479	21.2.8 特定的拖动源与放置目标	528
19.1 引入轻量构件	479	21.2.9 继承与授权	528
19.1.1 AWT: 重量构件的世界	479	21.2.10 映射	532
19.1.2 轻量构件与重量构件	479	21.2.11 自动滚动	539
19.2 简单的轻量构件	480	21.3 小结	545
19.2.1 简单的重量构件	480	第 22 章 自定义对话框	546
19.2.2 从重量构件到轻量构件	481	22.1 对话框类	546
19.3 轻量容器	483	22.1.1 GJDialog	546
19.4 轻量构件与 zorder	485	22.1.2 非模式对话框和 DialogClient 接口	547
19.5 轻量构件及其 Graphics	487	22.1.3 再论 GJDialog	548
19.6 轻量构件与首选尺寸	489	22.2 WorkDialog	552
19.7 小结	489	22.3 ButtonPanel	553
第五部分 高级主题	491	22.4 Postcard	555
第 20 章 剪贴板与数据传输	491	22.5 MessageDialog	556
20.1 java.awt.datatransfer 软件包	491	22.6 YesNoDialog	560
20.2 Clipboard 类	492	22.7 QuestionDialog	564
20.2.1 将数据复制到剪贴板及从剪贴板 取出数据	492	22.8 小结	569
20.2.2 ClipboardOwner 类	492	第 23 章 橡皮带技术	570
20.3 系统剪贴板	493	23.1 橡皮带类	570
20.4 局部剪贴板	497	23.2 Rubberband 基本类	571
20.5 数据传送机制	497	23.2.1 Rubberband 方法及类成员	571
20.5.1 数据格式	497	23.2.2 以 XOR 模式绘图	573
20.5.2 Transferable 对象与数据格式	498	23.2.3 绘制橡皮带线条	577
20.5.3 StringSelection	499	23.2.4 绘制橡皮带矩形与椭圆形	578
20.6 将图像复制到剪贴板	500	23.3 橡皮带面板	579
20.6.1 ImageSelection ——封装图像的 Transferable 对象	500	23.4 使用 DrawingPanel 类	583
20.6.2 使用 ImageSelection 类	502	23.5 重分解单元测试	587
20.6.3 增加另外一种数据格式	507	23.6 小结	588
20.7 传递自定义 AWT 构件	511	第 24 章 双缓冲技术	589
20.7.1 封装自定义 AWT 构件的可传递 对象	511	24.1 双缓冲技术与动画	589
20.7.2 图像按钮传送 applet	512	24.2 双缓冲如何工作	589
20.8 小结	515	24.3 可拖动轻量构件与双缓冲容器	591
第 21 章 拖放技术	516	24.3.1 Util 类	592
21.1 java.awt.dnd 软件包	516	24.3.2 BackingStore 类	594
21.2 拖动源与放置目标	517	24.3.3 DoubleBufferedContainer 类	597
21.2.1 简单的拖放程序范例	517	24.3.4 Lightweight 类	605
21.2.2 拖动意图	523	24.4 小结	608
21.2.3 拖动源	524	第 25 章 子图形动画	609
21.2.4 放置目标	525	25.1 构成	609

25.2 序列与子图形	609	25.4.4 EdgeCollision	624
25.2.1 Sequence	609	25.5 使用 animation 软件包	625
25.2.2 子图形	613	25.5.1 简单的动画	625
25.3 Playfield 与 DoubleBufferedContainer	618	25.5.2 碰撞动画	628
25.4 冲突检测	622	25.5.3 两个子图形之间的冲突	631
25.4.1 CollisionArena	622	25.6 小结	634
25.4.2 CollisionDetector	623	附录 A AWT 类框图	635
25.4.3 SpriteCollisionDetector	623	附录 B 关于本书的 CD-ROM	637

第三部分 事件和布局管理器

第 8 章 基于继承的事件处理

在本章中，我们要讲述的是初始的 AWT 事件模型。在新出的 1.1 版本的 JDK 中，AWT 对事件模型有了很大的改进。我们在本书中保留这一章是因为 AWT 仍然支持旧版本的事件模型。但是，在以后的发行的 AWT 版本中，将不再支持旧版本的事件模型，所以，如果你对旧事件模型的有关信息知道得非常清楚，你可以忽略本章而直接进入下一章。在下一章中，我们要讨论新事件模型。在本章的最后，我们增加了一节，用来讲述旧事件模型的缺点和如何使用新事件模型对它们进行修改。在本书以后的版本中，这一章的内容将会被删除掉。

现代图形用户接口一般是事件驱动，这就意味着它们将花费大量的空闲时间，用来等待事件的出现。当事件发生时，事件处理程序将对事件做出反应。在 AWT 中 applet 程序和应用程序是不同的，所以在本章中，我们介绍 AWT 构件中的事件处理，并提供几点说明，包括何时覆盖事件处理方法和以及何时处理事件或将之传递。在本章中，我们还介绍几个 applet 程序实例，对事件处理技术进行说明，如检测鼠标的双击、管理动作事件、产生事件并向自定义构件传递等。

8.1 旧 AWT 事件模型

和几乎所有的面向对象用户接口工具包一样，AWT 属于事件驱动。事实上，因为 AWT 使用了同位体方法，所以基础的事件子系统都是使用本地平台来运行 applet 程序或应用程序。例如，如果在 Motif 窗口中运行 applet 程序，则它使用的是 X Window 事件处理系统。然而，我们讨论的焦点并不是上面所说的本地事件系统，而是我们对事件做出反应的 AWT 层。

要注意本章讨论的是 AWT 1.0 和 AWT 1.02 版本中的原先的、基于继承的事件模型。当前的 AWT 版本[⊖]含有一种新的事件模型，该事件模型以基于授权代替基于继承并改善了旧事件模型。我们建议你使用新的事件模型代替旧事件模型。事件模型是向后兼容的，因此如果你以前接触过原先版本的事件模型，你可以忽略本章。

关于授权事件模型，我们将在下面的一章中进行讨论。如果一定要使用旧的、基于继承的事件模型，那么本章内容是很适合于你的；如果你能使用授权事件模型，那么你就可以忽略本章，直接进入下一章的学习。

8.1.1 覆盖事件处理方法

从本质上来讲，使用旧 AWT 事件模型处理事件是相当简单的：当事件在构件中出现时，构件中的一个方法被调用。如果你想让构件响应事件，你可以简单地覆盖适当的方法。例如，调整构件大小时，你可以调用构件中的 void setSize (int w, int h) 方法；如果你想在调整大小

⊖ 1.1 以后的版本。

事件发生时做一些动作，你应该覆盖 `setSize ()` 并处理其事务。在 AWT 中，有很多 `setSize ()` 这样的构件方法，可以用来响应事件。在下面的表 8-1 中，我们列出了这类方法中的一部分。

表 8-1 构件中使用的驱动事件方法

方 法	事 件
常覆盖的构件方法	
<code>void setSize (int w, int h)</code>	构件已经被调整大小
<code>void setBounds (int x, int y, int w, int h)</code>	构件被调整大小并/或移动
<code>void paint (Graphics g)</code>	构件需要被画出
<code>void update (Graphics g)</code>	默认情况下，清除并重绘
<code>void addNotify ()</code>	创建同位体
不常覆盖的构件方法	
<code>void setLocation (int x, int y)</code>	将构件移动到 x, y 位置
<code>void layout ()</code>	构件需要被布置
<code>void validate ()</code>	如果非法，则布置构件

在表 8-1 中，第一部分中所列举的方法是我们 在开发自定义构件时经常覆盖的方法。而在第二部分所列举的方法则针对一些不常见的事件，例如需要布置构件——请参见后面的第 10 章“构件、容器和布局管理器”中相关的内容。几乎不需要覆盖和这些事件相关的方法。当然，通过覆盖在表 8-1 中所列出的方法，你可以改写构件的默认方法。你将会经常看到在增加改进之前，调用超类版本的方法对事件进行处理：

```
public addNotify () | //Default version creates the peer
    super.addNotify (); //Peer created by this call
|
```

8.1.2 被传播的事件

在上面，我们讨论的事件发生在构件中，它们在发生事件的构件中被处理（或忽略）。这些事件仅仅涉及到它们所发生的构件。

另一类事件被称为被传播的事件，无论被传播的事件什么时候发生在构件中，构件的 `handleEvent (Event)` 方法被调用。构件的 `handleEvent (Event)` 方法可以选择是向其容器传播事件还是完全在其本身中处理事件。

注意被传播的事件和在表 8-1 中列出的事件之间的差别。被传播的事件一般是通过构件的 `handleEvent` 方法来处理的，一个 `java.awt.Event` 被传递给这个方法。被传播的事件可以传播给构件的容器；假如这样的话，容器的 `handleEvent` 方法被调用，并且 `java.awt.Event` 被向前传递。

因为被传播的事件涉及一个 `java.awt.Event`，所以它们一般是引用 AWT 事件。在本章的剩余部分，我们仅仅讨论和这种事件相关联的部分。

8.1.3 事件类型常量

每个 `java.awt.Event` 都有一个对应的 id 域，该 id 域的值指示已经发生的事件的类型。通常，构件的 `handleEvent` 方法做的第一件事是检测事件的 id 域来判断事件的类型。例如，一个和鼠标事件相关的 `handleEvent` 方法可能像下面这样：

```

public boolean handleEvent (Event event) {
    if (event.id == Event.MOUSE_DOWN) {
        // react to mouse down
        return true; // event fully handled, do not propagate
    }
    else if (event.id == Event.MOUSE_UP) {
        // react to mouse up
        return true; // event fully handled, do not propagate
    }
    // let superclass handle event and decide to propagate
    return super.handleEvent (event);
}

```

注意在上面所列举的例子中，如果事件是鼠标按下或释放事件，则其返回值是 true，表示我们已经完全完成处理事件，并不希望事件传播给构件的容器。如果事件不是鼠标按下或释放事件，将使用超类对事件进行处理。如果返回的值是 false，则表示事件将直接被传播给构件的容器，并且不给超类机会处理事件。所以，如果覆盖的 handleEvent 方法没有完全完成事件，最好返回 super.handleEvent (event)，而不要直接将事件传播给构件的容器。

在下面的表 8-2 中，完整的列出了 java.awt.Event 类型常数。

表 8-2 java.awt.Event 常数

事件常数
窗口事件常数
WINDOW_DESTROY
WINDOW_EXPOSE
WINDOW_ICONIFY
WINDOW_DEICONIFY
WINDOW_MOVED
键盘事件常数
KEY_PRESS
KEY_RELEASE
KEY_ACTION
KEY_ACTION_RELEASE
鼠标事件常数
MOUSE_DOWN, MOUSE_UP
MOUSE_MOVE
MOUSE_ENTER, MOUSE_EXIT
MOUSE_DRAG
滚动条事件常数
SCROLL_LINE_UP, SCROLL_LINE_DOWN
SCROLL_PAGE_UP, SCROLL_PAGE_DOWN
SCROLL_ABSOLUTE

(续)

事件常数

列表事件常数

LIST_SELECT, LIST_DESELECT

动作事件常数

ACTION_EVENT

LOAD_FILE

SAVE_FILE

GOT_FOCUS, LOST_FOCUS

AWT 技巧：避免从 `handleEvent` 方法中返回 `false` 值

从 `handleEvent` 方法中返回 `false` 会把事件传播给构件的容器，而不让构件的超类有机会处理事件。所以避免从 `handleEvent` 方法中返回 `false` 是一个好的思想。如果类没有完全完成处理某个事件，那么，它将给它的超类一个机会来处理事件并由超类决定是否将事件传播给构件的容器。即使你知道你的超类对某个类型的事件不感兴趣，也应这么做，因为超类可能在未来会被改变以处理这种事件。

被传播的事件处理方法

在覆盖的 `handleEvent` 方法中，检查事件的 `id` 域并判断出事件的类型，这是相当难看的，而且需要保存在表 8-2 中列出的所有事件常数，表 8-2 中列出的常数并不都是面向对象的^①。因此，`java.awt.Component` 实现了一个 `handleEvent` 方法用于分析事件的类型，并调用在表 8-3 中列出的一个辅助方法。表 8-3 中列出了可供 `java.awt.Component.handleEvent()` 调用的方法。

表 8-3 被传播的事件的处理方法

方 法	事 件
<code>boolean action (Event, Object)</code>	动作事件
<code>boolean mouseUp (Event, int, int)</code>	释放鼠标
<code>boolean mouseDown (Event, int, int)</code>	按下鼠标
<code>boolean mouseDrag (Event, int, int)</code>	拖动鼠标
<code>boolean mouseMove (Event, int, int)</code>	移动鼠标
<code>boolean mouseEnter (Event, int, int)</code>	鼠标进入构件
<code>boolean mouseExit (Event, int, int)</code>	鼠标离开构件
<code>boolean keyUp (Event, int)</code>	按键释放
<code>boolean keyDown (Event, int)</code>	按键被按下
<code>boolean gotFocus (Event, Object)</code>	构件得到焦点
<code>boolean lostFocus (Event, Object)</code>	构件失去焦点

当处理被传播的事件时，你可以选择下面给出的两个方法中的一个：

① 请参见 Meyer、Bertrand 所著的“Object Oriented Software Construction”的 10.2.2 节。