

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
China Machine Press

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

njs3p6/03

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 · 2000年8月第2次印刷

787mm × 1092mm 1/16 · 41印张

印数: 5 001-8 000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，为了准确掌握 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。

注：Java 2 以前称为 Java1.2，实际上这两个版本是完全一样的。本书中，我们统一采用“Java 2”这个提法。

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 图像生产者和图像消费者	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	
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 SimpleI18NTest 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

第一部分 入门

第 1 章 绪 论

1.1 Java 基本类

Java 基本类 (JFC)，由一些软件包组成。这些软件包主要包括下面一些应用程序接口 (API)：

- 抽象窗口工具集 (AWT) (1.1 及以上版本)。
- Swing 构件。
- Java 2D 应用程序接口 (2D API)。
- 兼容程序接口。

上面列出的这些应用程序接口可能会出现在多个软件包中。例如：2D API 在 `Java.awt` 和 `Java.awt.image` 软件包中都存在，虽然像 `Java.awt.geom` 等一些特殊的软件包也支持 2D API，但是大量的 2D API 类都存在于 `Java.awt` 软件包中。

AWT (1.1 及以上版本) 是 JFC 的核心，同时也是本卷书的主题。AWT 为 JFC 的构成提供了以下的基本结构：

- 代理事件模型。
- 轻量构件。
- 剪贴板和数据传输。
- 打印和无鼠标操作。

1.2 抽象窗口工具集

在开发 applet 和图形应用程序时，一般需要用到 AWT，AWT 是免费 Java 开发工具包 (JDK) 的一部分。

AWT 的作用是给用户提供基本的界面构件，例如按钮、列表、菜单、文本域等等。AWT 构件主要是用来建立图形用户界面的独立平台。此外，AWT 还提供事件处理结构、支持剪贴板、数据传输和图像操作。

随着 2D API 的出现，AWT 还包括提供高级字体操作、打印、地理数据获取和输入方法等功能的软件包。在表 1-1 中，我们列出了 AWT 中的 Java 软件包。

表 1-1 AWT 中的软件包

AWT 软件包	描 述
<code>Java.awt</code>	基本构件实用工具
<code>Java.awt.accessibility</code>	辅助技术
<code>Java.awt.color</code>	颜色和颜色空间

(续)

AWT 软件包	描 述
Java.awt.datatransfer	支持剪贴板和数据传输
Java.awt.dnd	拖放
Java.awt.event	事件类和监听者
Java.awt.font	2D API 字体软件包
Java.awt.geom	2D API 几何软件包
Java.awt.im	引入方法
Java.awt.image	图像处理工具包
Java.awt.peer	同位体构件、界面包
Java.awt.print	支持打印 2D API
Java.awt.swing	Swing 构件
Java.awt.test	测试 AWT 函数有限子集的独立 applet

AWT 的初始版本是基于在简单用户界面中开发小 applet 程序而设计的，与之相比，当前的 AWT 做了很大的改进，它提供事件模型重新设计、剪贴板和数据传输支持以及打印和无鼠标操作等功能。从而与 Parc Place 的 VisualWork 或 Borland 公司的 Object Windows Library (OWL) 等企业级用户界面具有更多的可比性。

注意 本卷没有讲述所有的 Java.awt 软件包，例如 Java.awt.font 和 Java.awt.geom 软件包我们就没有进行讨论，关于这两个软件包，我们将在本套书的第 3 卷——2D API 中进行详细讲述。

1.3 同位体和平台独立

随着 Applet 程序和图形应用程序接口的发展，AWT 提供了一系列的通用类，这些通用类在引用时不需要考虑特定的窗口平台，同位体 (peer) 就属于这种 AWT 类集。同位体是一种本地图形用户接口 (GUI) 构件，由 AWT 类管理。同位体的工作方法和它们对程序开发的影响常常让人混淆，在后面的有关章节中，我们将深入介绍它们。

AWT 构件中，包含有对其同位体的大量实用操作。例如，如果你使用 AWT 创建一个 menu 类的实例，那么当 Java 运行时系统将创建一个菜单同位体的实例，而由创建的同位体实际执行菜单的显示和管理。在创建菜单实例中，Solaris JDK 将产生一个 Motif 菜单同位体；Windows 95 将产生一个 Windows 95 菜单同位体；Macintosh JDK 将产生一个 Macintosh 菜单同位体等等。图 1-1 说明了同位体配合本地窗口系统显示构件的过程。

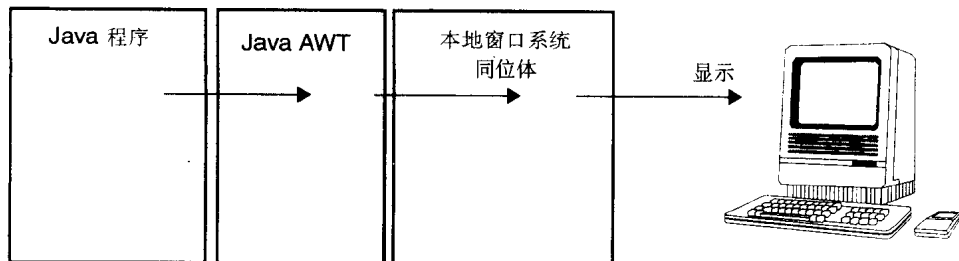


图 1-1 工作中的同位体

一个 Java 程序创建并显示 AWT 构件，AWT 构件创建并显示本地构件（同位体）。

AWT 开发组决定使用同位体方法，这一方法使得交叉平台窗口工具开发变得极为迅速。使用同位体可以避免重新实现本地窗口构件中已包含的实用工具，而且，使用同位体还能使 applet 和应用程序保留在本地系统中，这是因为同位体实质上是由本地构件组成的，而 AWT 类仅仅是同位体外围的包装与操作工具。

虽然在使用 AWT 时，很少需要直接处理同位体，但它们的存在却影响其操作结果。例如，如果没有同位体，则某些 `Java.awt.Component` 方法不会象我们所预期的那样进行工作。

使用同位体方法可以在记录时间内实现 GUI 工具构件。然而，使用同位体也有很多的缺点，同位体设计基础存在缺陷并且不能缩放，在本书后面的相关章节（11.8 节“构件和同位体”）中，我们将进行更详尽的讨论。

1.3.1 轻量构件

AWT 构件全都是重量构件，即它们都具有同位体，并且在本地（不透明）窗口中进行显示。这样使用将花费昂贵的代价，而且在更改其默认行为时，不可以将其派生为子类。此外，它们必须是矩形的，而且不能有透明的背景。

同位体可以快速产生一个 GUI 工具构件。因为本地同位体做了更多的实际工作，而 AWT 类所做的仅仅是表面工作，因此，它很容易开发。开发最初的 AWT，只用了不到 6 个星期的时间。但这种效率带来的利益在很大程度上被一些不利因素抵销了，比如基本的同位体结构、有限的事件模式以及同位体与 AWT 之间不匹配造成的大量缺陷。

1.1 版本的 AWT 引入了轻量构件的概念。轻量构件直接扩展了 `java.awt.Component` 或 `java.awt.Container`。轻量构件没有同位体，在其重量容器窗口中显示，而不是在其本身窗口中显示。轻量构件不会导致与它们自己关连的不透明窗口的性能损失，而且还可以有透明的背景。其中有透明背景的性能意味着即使轻量构件的界限域实际上是矩形的，它也可以显示为非矩形。

1.3.2 AWT 和 Swing 构件

Swing 构件大部分是轻量构件，它们替代 AWT 中的重量构件。除此之外，Swing 构件还包括许多其他有用的构件，如分隔符、图像按钮、文档编辑器以及颜色选择等。关于 Swing 构件，本书的第 2 卷——Swing 构件中将有详细的讨论。

1.3.3 AWT 和 2D API

最初的 AWT 提供了图形软件包，虽然不太完整，但是其质量还是相当不错的。例如，图像软件包提供了图像筛选的基础结构，但它除了为图形笔提供一个像素大小的正方形外，没有提供其他任何东西。

2D API 由一组 2D 图形和图像类构成。2D API 中包含下列软件包：

```
java.awt.color
java.awt.font
java.awt.geom
java.awt.print
```

此外，2D API 还扩展了其他的一些 AWT 软件包，例如 `java.awt.Composite` 和 `java.awt.Stroke`。`Java.awt.image` 软件包也包含许多的 2D API 软件包。

本卷中，我们没有深入探讨 2D API，但是在本卷中，我们将详细地讲述由 AWT 提供的基

本图形和图像处理能力。关于 2D API 的详细内容，读者可以参阅本套书的第 3 卷——2D API。

1.4 构件：AWT 的基础

AWT 是一个构件的世界，其中，大约有一半的类在 `java.awt.Component` 类的扩展 AWT 中。`Component` 类和它所支持的类是 AWT 的基础。

- `Component`（构件）——菜单、按钮、标签、列表等构件的抽象基本类。
- `Container`（容器）——扩展 `Component` 的抽象基本类。由 `Container` 演变的类，如 `Panel`、`Applet`、`Window`、`Dialog` 和 `Frame` 等。在容器中，可以包含多个构件。
- `LayoutManager`（布局管理器）——定义容器中构件位置和尺寸方法的接口。Java 中定义了几种默认的布局管理器。
- `Graphics`（图形类）——定义构件内图形操作的基本类，每个构件都有一个相关的图形对象。

图 1-2 是一个类框图，显示了构件、容器和布局管理器之间的关系[⊖]。

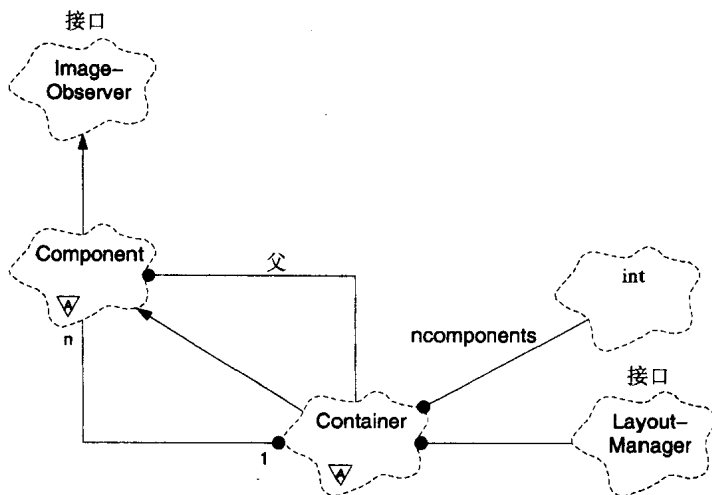


图 1-2 构件、容器和布局管理器

1.5 构件、容器和布局管理器

构件和容器构成了 AWT 中的简单基础关系：容器中可以包含构件，所有的容器都有布局管理器[⊖]，用来指定容器中构件的位置和形状。布局管理器的职能由 `java.awt.LayoutManager` 和 `java.awt.LayoutManager2` 接口指定，AWT 中的许多操作都发生在构件、容器和布局管理器之间。

1.5.1 构件

在 Java 语句中，面板、滚动条、标签、文本框、按钮等等用户界面构件一般都称作构件，因为它们全都扩展了 `java.awt.Component` 类。表 1-2 列出了完全从 `java.awt.Component` 演变产

⊖ 图形 Java 包含了许多类框图，可以显示类之间的关系。附录 A “AWT 类框图” 全面详细地介绍了 AWT 的类框图。

⊖ `java.awt.Container` 类除外。

生的 AWT 构件。

表 1-2 AWT 构件

构 件	超 类	描 述
Button	Component	触发行为的文本按钮
Canvas	Component	绘制图形的画布
Checkbox	Component	可检验的布尔构件
Choice	Component	文本输入的弹出菜单
Dialog	Window	可模式化窗口
FileDialog	Dialog	选择文件的相关平台对话框
Frame	Window	具有标题栏和可选菜单的顶层窗口
Label	Component	显示字符串的构件
List	Component	文本输入的可滚动列表
Panel	Container	一般构件容器
Scrollbar	Component	滚动项目的 adjustable 构件
ScrollPane	Container	可滚动容器
Textarea	TextComponent	多行可滚动的文本框
TextComponent	Component	TextArea 和 TextField 的基本功能
TextField	TextComponent	输入文本的单行构件
Window	Container	没有标题的无边界窗口

图 1-3 给出了除扩展 `java.awt.Window` 类之外的 Java AWT 标准构件。

1.5.2 基本构件工具

`java.awt.Component` 类是一个抽象类，它为扩展的类提供了许多功能。例如，一个构件可以有以下成员：

- 图形对象。
- 位置。
- 尺寸。
- 本地同位体。
- 父容器。
- 字体和字体维数（在 AWT 中指字体尺度）。
- 前景和背景颜色。
- 地点、场所。
- 最小值、最大值和首选尺寸。

1.5.3 容器

`Java.awt.Container` 也是一个扩展 `Component` 的抽象类。一个容器可以包含多个构件，您可以使用容器将相关构件组合在一起，成为一个整体。使用容器的这种技术，可以简化您的 Applet 程序设计，而且对于构件的显示安排也很有用。注意 `Applet` 类是一个扩展 `Container` 的 `Panel`