

# 目 录

## 第一部分 概念参考

第1章 Swing概要 .....	(3)
1.1 JFC技术 .....	(3)
1.2 Swing特点 .....	(4)
1.3 Swing包 .....	(4)
1.4 JComponent类 .....	(5)
1.5 辅助选项支持 .....	(8)
第2章 Swing小程序 .....	(10)
2.1 Hello! Swing World .....	(10)
2.2 Swing小程序集 .....	(12)
第3章 事件处理 .....	(15)
3.1 代表事件模型 .....	(15)
3.2 Swing事件、源和监听对象 .....	(15)
3.3 底层事件 .....	(19)
3.4 匹配类 .....	(24)

## 第二部分 技术参考

第4章 框架、窗格和面板 .....	(31)
4.1 框架 .....	(31)
4.2 Swing的基本容器 .....	(34)
4.3 内部框架 .....	(40)
4.4 桌面窗格 .....	(41)
4.5 面板 .....	(44)
4.6 拆分窗格 .....	(47)
4.7 卡片式窗格 .....	(51)
第5章 布局管理器 .....	(54)
5.1 布局管理器 .....	(54)
5.2 流布局管理器 .....	(55)
5.3 网格布局管理器 .....	(58)
5.4 边界布局管理器 .....	(59)
5.5 卡片布局管理器 .....	(61)
5.6 无序网格布局管理器 .....	(66)

5.7 框式布局.....	(72)
<b>第6章 Swing设计 .....</b>	<b>(81)</b>
6.1 M – UI: 简化的 MVC .....	(81)
6.2 组件模型.....	(82)
6.3 “UI – 代表” .....	(83)
6.4 可插入外观.....	(83)
6.5 Swing设计的模拟代码示例 .....	(85)
<b>第7章 图标、标签和边界 .....</b>	<b>(95)</b>
7.1 图标.....	(95)
7.2 图像图标.....	(97)
7.3 Swing 标签 (JLabel) .....	(100)
7.4 边界 .....	(104)
7.5 边界的类型 .....	(105)
7.6 自定义边界 .....	(111)
<b>第8章 按钮和复选框.....</b>	<b>(115)</b>
8.1 AbstractButton 类.....	(115)
8.2 按钮 .....	(117)
8.3 切换按钮 .....	(122)
8.4 复选框 .....	(127)
8.5 单选按钮 .....	(131)
<b>第9章 列表和组合框.....</b>	<b>(137)</b>
9.1 列表 .....	(137)
9.2 数据模型 .....	(138)
9.3 绘制列表单元 .....	(141)
9.4 列表选择和事件 .....	(145)
9.5 多重鼠标单击 .....	(149)
9.6 组合框 .....	(153)
9.7 JComboBox 模型 .....	(154)
9.8 绘制项目单元 .....	(157)
9.9 可编辑的组合框 .....	(160)
<b>第10章 表格 .....</b>	<b>(165)</b>
10.1 简单表格.....	(165)
10.2 表格模型.....	(169)
10.3 表格选择.....	(175)

10.4 编辑表格单元	(179)
<b>第 11 章 文本小配件</b> ..... (183)	
11.1 JTextComponent 类	(183)
11.2 文档接口	(183)
11.3 文本字段	(183)
11.4 口令字段	(186)
11.5 文本区域	(188)
11.6 编辑器工具箱	(194)
11.7 编辑器窗格	(194)
11.8 文本窗格	(197)
<b>第 12 章 滚动条和滑动条</b> ..... (201)	
12.1 滚动条	(201)
12.2 滚动窗格	(206)
12.3 滑动条	(209)
<b>第 13 章 菜单、工具条和工具提示</b> ..... (216)	
13.1 菜单条和菜单	(216)
13.2 菜单项	(219)
13.3 复选框菜单项	(222)
13.4 单选按钮菜单项	(222)
13.5 弹出式菜单	(225)
13.6 工具条	(228)
13.7 多控件的公共操作	(232)
13.8 工具提示	(236)
<b>第 14 章 对话框和选项窗格</b> ..... (240)	
14.1 对话框	(240)
14.2 选项窗格	(244)
14.3 消息对话框	(246)
14.4 确认对话框	(250)
14.5 输入对话框	(254)
14.6 选项对话框	(258)
<b>第 15 章 文件选择器和颜色选择器</b> ..... (260)	
15.1 文件选择器	(260)
15.2 定制文件视图	(265)
15.3 文件过滤器	(266)

15.4 颜色选择器.....	(272)
<b>第 16 章 树 .....</b>	<b>(277)</b>
16.1 创建树.....	(277)
16.2 树模型.....	(281)
16.3 用树完成交互.....	(287)
16.4 绘制树节点单元.....	(292)
16.5 可编辑的树节点.....	(295)
<b>第 17 章 定时器和进度指示器 .....</b>	<b>(297)</b>
17.1 定时器.....	(297)
17.2 进度条.....	(302)
17.3 进度监视器.....	(306)
17.4 输入流的进度监视器.....	(310)
<b>第三部分 语 法 参 考</b>	
附录 A JFC Swing 快速参考 .....	(317)
附录 B AWT 事件类快速参考 .....	(582)
附录 C Swing 中重要的 AWT 类 .....	(603)
附录 D 辅助选项快速参考 .....	(618)

# 第一部分 概念参考

第1章 Swing概要

第2章 Swing小程序

第3章 事件处理

# 第1章 Swing 概要

JDK 1.2(Java 开发工具包,后正式命名为 Java 2)的正式版本是 1998 年 12 月发布的。JDK 1.2 包括了一个复杂的图形 API 和用户接口 API 的集合,该集合的核心软件包称为 Java 基础类(JFC)。JFC 中一个重要的技术就是 Swing,它组合了大量的可用于构建复杂用户界面的轻量级组件。图 1-1 表示出了 JFC 及 Swing 与 JDK 1.2 之间的关系。

Swing 组件是现有的抽象窗口工具包(AWT)组件的增强,但不必用它们完全取代 AWT 组件。这两种组件可以用在同一个界面上。Swing 组件包括从按钮、复选框、滚动条和滑动条这样的简单控件到诸如文本窗格和编辑器窗格那样复杂的工具。

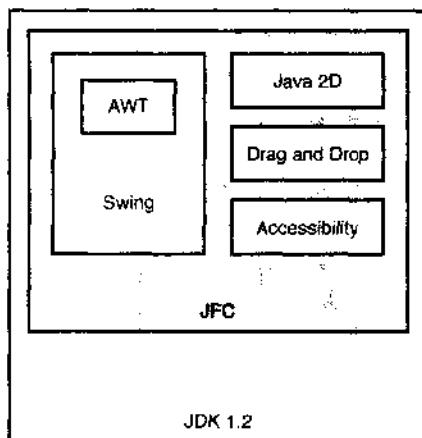


图 1-1 JDK 1.2, JFC 和 Swing

## 说明:

Swing 最初是作为 JDK 1.1 的扩展 API 出现的,它必须单独下载后才能使用。而在 JDK 1.2 中,JFC 和 Swing 已经被直接集成在了 JDK 中,因此就不需要另外得到 Swing API 了。

## 1.1 JFC 技术

JFC 包含五种主要技术,其中之一就是 Swing。JFC 也包括了抽象窗口工具包(AWT),辅助选项、拖 – 放以及 2D 图形处理。本书着重介绍 Swing 和辅助选项 API。以下是 JFC 的简要说明:

- 辅助选项 API——辅助技术(或系统)对于那些需要额外的帮助来使用用户界面的残疾人来说是很有用的。辅助系统包括屏幕阅读器、屏幕放大器以及语音识别系统。辅助选项 API 提供了一个接口,它使得辅助技术能够与 JFC 和 AWT 组件进行交互。

和通信。本章将要进一步讨论辅助技术，并提供附录 D“辅助选项快速参考”的介绍。在 Swing 中还要考察相关的 API。

- 抽象窗口工具包(AWT)——抽象窗口工具包，亦即 AWT，对 Java 程序员来说并不陌生。对于 JFC 本身来说，AWT 是它的基石，也是随 JDK 1.0 发行的核心库之一。事实上，AWT 奠定了 JFC 组件的基础。尽管可以用 Swing 组件创建用户界面，但仍需要用到 AWT 中所支持的布局管理和事件处理模块。正因为如此，本书包含了一些 AWT 中的布局管理和事件处理的内容，以使读者不必另外去查找有关的信息。本书最后也提供了相关 API 的快速参考。如果需要更多的信息，请参看 Sams 公司出版的《JFC unleashed》一书。
- 2D 图形/Java 2D API——在 Java 程序设计中，AWT 中的 API 在一定程度上支持图像处理。但随着 Java 技术逐步成熟，人们认识到，有必要将更为复杂的图像处理 API 引入到 Java 之中。因此，JDK 1.2 就包含了 2D 图像处理，以增强现有图像处理能力。2D 图像 API 能够支持先进的 2D 图形和图像。
- 拖 - 放——借助于可在本地平台进行拖 - 放操作的应用程序，一个与 Java 应用程序相邻的本地用户应用程序就能在 Java 应用程序和本地应用程序之间进行拖 - 放操作。用户动作(拖动动作)的精确程度与其在本地平台上的动作是一致的。

## 1.2 Swing 的特点

Swing 的开发源于 Model - View - Controller(MVC)体系结构。基于 MVC 的体系结构允许 Swing 组件被不同的数据模型和视图所替代。可插入的外观效果(pluggable look - and - feel)就是 MVC 体系结构应用的结果。由于 Java 是不依赖于平台的语言，能够在任何客户机上运行，因此可以针对任何平台来调整软件的外观效果。下面是 Swing 主要特性的总结：

- 轻量级组件——从 JDK 1.1 开始，AWT 就支持轻量级组件的开发。要使一个组件成为轻量级组件，那么该组件就不能依赖任何本机系统类，这种类也叫做“对等类”(“peer” classes)。在 Swing 中，大多数组件都有其自己的由 Java 外观类所支持的视图。因此，组件的视图不依赖于任何“对等类”。
- 可插入外观组件——这一特性使得用户不必重新启动应用程序就能够看到 Swing 组件的外观效果。Swing 库支持跨平台的外观——也称为 Java 外观——它使得应用程序不论在哪里运行都具有同样的外观效果。通常，本机外观效果是针对于程序所运行的特定系统的，Windows 和 Motif 就是如此。Swing 库提供了一个 API，它对于确定应用程序的外观效果提供了极大的灵活性。

## 1.3 Swing 包

Swing API 被组织成了若干个包，以支持诸如组件、可插入外观组件、事件、对象边界等不同类别的 API。附录 A“JFC Swing 快速参考”中给出了包的成员(类和接口)及其 API 方法。下面列出了 Swing 中各个包的名称：

`javax.swing`

```
javax.swing.border  
javax.swing.colorchooser  
javax.swing.event  
javax.swing.filechooser  
javax.swing.plaf  
javax.swing.plaf.basic  
javax.swing.plaf.metal  
javax.swing.plaf.multi  
javax.swing.table  
javax.swing.text  
javax.swing.text.html  
javax.swing.tree  
javax.swing.undo
```

另外,还有两个包中存放了 Motif 和 Windows 操作系统的外观类。这两个类包含在 JDK 1.2 中,但由于它们所处理的操作系统属于 Sun Microsystem 公司之外的产品,因此这两个包采用了不同的名称:

```
com.sun.swing.plaf.motif.*  
com.sun.swing.plaf.windows.*
```

## 1.4 JComponent 类

JComponent 类引入时是作为 Swing 组件的父类。这是一个抽象类,它封装了特定于 Swing 组件的基本特性和操作。AWT 的 Container 类是 JComponent 的父类,如图 1-2 所示。因此,AWT 是建立 Swing 组件集的基础。

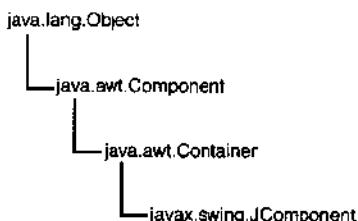


图 1-2 Jcomponent 的类层次结构图

### 说明：

由于 Swing 组件是从 AWT 容器类扩展得到的,因此,可以期望 Swing 组件具有容器的特性。但是也必须根据具体的组件而小心地使用这个特性。

JComponent 类是 Swing 组件各种特性通常的存放位置。可以引用附录 A 中列出的代表这些特性的各个方法。下面所列出的就是这里所述的各种特性:

- 辅助选项支持
- 设定组件边界

- 设定组件界限
- 从一个点控制多个组件的动作
- 击键处理
- 附加更改监听程序
- 激活性质更改事件
- 将组件在其显示区域内对齐
- 自动滚动
- 设置双缓冲区
- 为改变大小的动作设置标注属性
- 赋予工具提示
- 允许操作需要的焦点

#### 1.4.1 Swing 组件

本节给出 Swing 组件的概览。有关 Swing 标签和按钮的内容将通过详细的代码进行讨论,以便演示前面所介绍的诸如事件处理、容器、布局以及边界等概念。表 1-1 列出了 Swing 组件及其定义。

表 1-1 Swing 组件

组件	定义
JApplet	这个类代表 Swing 小程序,它从层次上扩展了 AWT 小程序
JButton	这个类代表 Swing 按钮。这些按钮可以在适当的位置(绝对的和相对的)带有图标和文本。可以用它所支持的任何一个构造函数创建按钮对象(请参看第 8 章,“按钮和复选框”)。做法如下: <code>JButton buttonRef = new JButton(text,icon);</code> 这里 text 和 icon 是用在按钮上的文本标签和图标
JCheckBox	这个类代表 Swing 的复选框(请参看第 8 章)
JColorChooser	这个类代表 Swing 的颜色选择板。颜色选择板是现成的对话框,其上有一个调色板。这一部分内容将在第 15 章“文件选择框和颜色选择框”中介绍
JComboBox	Swing 中的组合框是对 AWT 组件 Choice 的一个增强。组合框是一个下拉列表,它能够在一个显示区域内显示出多个选项。也可以在显示区域里输入一项并将它编入列表中(请参看第 9 章,“列表框和组合框”)
JDesktop	这是一种附在 Swing 的内部框架上的容器。在第 4 章“框架、窗格和面板”中可以找到有关该组件更多的内容(也可以在本表中找到 JInternalFrame 的定义)
JDialog	这个类代表了 Swing 版本的对话框。JDialog 扩展了 AWT 的对话框,它要求将其组件添加到它的内容窗格上(请参看第 14 章,“对话框和选项窗格”)
JFileChooser	JFileChooser 类是一个对话框,它允许打开或保存信息内容。Swing 的文件选择框将在第 15 章中讨论

表 1-1 Swing 组件

(续表)

组件	定义
JFrame	JFrame 代表 Swing 的框架类, 它比 AWT 的框架类更为复杂。你可以分层向其中添加组件、菜单条, 或者[通过叫做玻璃窗格(glass pane)的组件]在上面绘制组件。有关框架类的内容将在第 15 章中讨论
JInternalFrame	内部框架是附在应用程序主框架上的。在应用程序中, 利用内部框架可以启动一次会话(session)过程。内部框架类的内容将在第 4 章中讨论
JLabel	这个类代表 Swing 中的标签, 它能够在某个适当位置显示文本串或图标。标签对象可以用下面的语句来创建: <code>JLabel labelRef = new JLabel(text, icon, JLabel.CENTER);</code> 这一条语句使用指定的 text、icon 以及按照水平方向对齐的要求创建了一个标签(请参看第 7 章, “图标、标签和边界”)
JLayeredPane	这个类代表 Swing 中的分层窗格。分层窗格建立了一个沿组件深度方向的维(dimension)。可以将组件放在分层窗格的每一层上。Swing 小程序和框架都缺省地包含了分层窗格。分层窗格的内容将在第 4 章中讨论
JList	JList 类代表能够在用户界面中显示一系列条目的组件。该组件用到了 M-U/I(改进的 MVC)体系结构并允许针对复杂程序设计中的列表模型进行处理(参看第 9 章)
JMenu	该类代表 Swing 的菜单(参看第 13 章, “菜单、工具条和工具提示”)
JMenuBar	该类代表 Swing 小程序或者框架上的菜单条, 第 13 章详细讨论了有关的内容
JMenuItem	该类代表 Swing 的菜单项。菜单项是类似于按钮的对象, 它能够被程序的特定动作所触发。Swing 的菜单项将在第 13 章中讨论
JOptionPane	该类代表 Swing 中的选项窗格。选项窗格是一些对话框, 这些对话框为用户显示了一些信息, 或者从用户那里获取信息, 以便程序根据用户所做的选择执行进一步的动作。选项窗格在第 14 章中进行讨论
JPanel	该类代表 Swing 的面板, 它除了能够组合其他的组件外, 还具有双缓冲(double buffering)的功能。窗格类将在第 4 章中进行讨论
JPasswordField	这个类代表一个文本字段, 它不显示输入的文本, 而是在输入过程中显示特殊的字符(比如 * )。口令字段的讨论放在第 11 章中进行
JPopupMenu	这个类代表 Swing 的弹出式菜单, 它包含了接受鼠标单击的功能(参看第 13 章)
JProgressBar	该类代表 Swing 的进度条类, 可以用进度条表示某些操作的状态, 第 17 章“定时器和进度指示器”中将讨论有关进度条的内容
JRadioButton	该类代表 Swing 的单选钮。利用单选钮可以从给定的一列互斥选项中选择一个。单选钮的讨论放在第 8 章中进行
JRadioButtonMenuItem	单选钮菜单项为表示互斥的菜单项, 有关的讨论见第 13 章
JScrollBar	滚动条用于水平或垂直滚动一个容器内的子组件。滚动条见第 12 章“滚动条和滑动条”中的讨论

表 1-1 Swing 组件

(续表)

组件	定义
JScrollPane	Swing 的滚动窗格是一个具有滚动功能的容器,滚动窗格的讨论见第 12 章
JSlider	该类创建一个可滑动对象,这个滑动对象允许从一个指定区间内选取特定的值。关于 Swing 的滑动条的讨论见第 12 章
JSplitPane	该类代表 Swing 的拆分窗格类。拆分窗格可以将位于其中的两个组件分割开来,有关的讨论见第 4 章
JTabbedPane	该类代表 Swing 所使用的卡片式窗格类。这个窗格类能够显示子组件,并且可以节省空间。子组件可以在多个卡片中一次一个地进行显示。卡片式窗格类将在第 4 章中讨论
JTable	JTable 类代表 Swing 中的表格,第 10 章“表格”中将讨论有关内容
JTextArea	该类代表 Swing 的文本区组件。文本区组件允许编辑多纯文本信息。可以在第 11 章中找到有关内容
JTextPane	该类可用来在 Swing 应用程序中创建复杂的编辑器。文本窗格既可以作为图像编辑器,也可作为文本编辑器。文本窗格组件将在第 11 章中详细讨论
JToggleButton	该类代表 Swing 的切换按钮。初次操作切换按钮,它会保持被按下的状态;再次操作后它便保持弹起状态。切换按钮将在第 11 章讨论
JToolBar	该类代表 Swing 的工具条对象。工具条是一个条形区域,可以为其添加一系列重要的按钮或者其他组件,以便更容易操作它们。工具条可以被做成浮动的。第 13 章中详细讨论了 Swing 的工具条
JToolTip	该类代表 Swing 的工具提示对象。“工具提示”可以显示一小段文字,用来指示一个组件的名称或说明该组件的目的。工具提示的内容将在第 13 章中讨论
JTree	该类用来显示一个用户界面中分层存储的数据,它代表了 Swing 的树对象。有关的内容将在第 16 章“树”中进行讨论。

## 1.5 辅助选项支持

辅助技术(也称为辅助选项技术)帮助那些身有残疾的用户更容易地使用计算机。辅助技术的例子包括为视力受损的用户所提供的屏幕放大功能,为盲人提供的能够将屏幕内容同语音进行合成的屏幕阅读器,以及对为那些不能正常使用键盘的人所提供的基于屏幕的键盘装置,等等。

Java 辅助技术支持提供了一个紧凑、简洁并且易于使用的可扩展 API 集合。利用辅助选项的界面和类,可以把 Java 小程序或 Java 应用程序中的用户界面组件精简成为可访问的或辅助技术(系统),这一技术提供了使用小程序和应用程序的途径。附录 D“辅助选项快速参考”中可以找到关于辅助选项 API 的快速参考。实现辅助选项支持的应用程序结构如图 1-3 所示。

除了辅助选项 API 之外,辅助技术还需必要的辅助选项工具。这些工具可以寻找实现辅助选项 API 的对象,支持将它们装入 Java 虚拟机,监视 AWT 事件和 JFC 事件,计数已打开的顶层 Java 窗口,获取鼠标的光标位置以及获取焦点的组件等等。

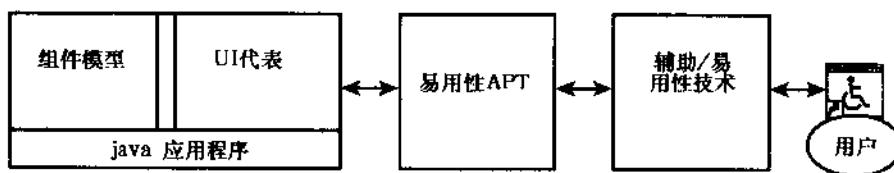


图 1-3 使用易用性支持的应用程序结构

**说明：**

作为应用程序开发者，通常不会用到辅助选项工具；这些工具是给辅助技术系统的开发人员所使用的。

## 第 2 章 Swing 小程序

这一章是有关使用 Swing 小程序的快速介绍。其中讨论了创建 Swing 小程序过程中所需的基本内容,以及 Swing 小程序是如何区别于标准小程序的。首先,会看到一个 Swing 小程序,“Hello! Swing World”;然后再研究这个 Swing 小程序的技术细节。

### 2.1 Hello! Swing World

清单 2.1 包含了一个简单的 Swing 程序 THello.java。这个程序只是简单地输出问候语“Hello! Swing World”(不是以前的问候语“Hello! World”)。“Hello! Swing World”按小程序和应用程序两种方式编写了相应的函数,所以你会在程序中发现一些额外的代码语句,事实上,该程序只有一条核心语句,具体如下:

```
getContentPane().add(new JLabel(  
    "Hello! Swing World",  
    new ImageIcon("AnimSpeaker.gif"),  
    JLabel.CENTER));
```

读过后面几章以后,这个语句的含义就会逐渐清晰的。现在可以看见动画图标和问候语“Hello! Swing World”显示在小程序或者应用程序框架中。另外,你可能已经注意到了分别代表 Swing 小程序和 Swing 框架的类标识符 JApplet 和 JFrame。这两个类分别从 AWT 类 Applet 和 Frame 继承而来。图 2-1 和图 2-2 显示了该程序分别作为小程序和应用程序的运行结果。注意到小程序的运行结果显示的标题栏指明了该程序是在小程序查看器中运行的。还可以看到状态栏中的状态信息“Applet Started”。



图 2-1 在小程序查看器中运行的 THello 小程序



图 2-2 THello 作为一个应用程序运行

**清单 2.1 演示 JApplet 的一个简单的 Swing 小程序/应用程序(THello.java)**

```
(THello.java)  
// Your first Swing program.  
  
/*  
 * <Applet code= THello width = 300 height = 150 >  
 * </Applet >
```

```
/*
import javax.swing.*; // Import the Swing classes
import java.awt.event.*; // for the WindowAdapter and WindowEvent classes

// This is your Swing applet (by extending JApplet).
public class THello extends JApplet {

    public void init() {
        // Here is the only program statement.

        // Create a string label to display the text as shown here
        // with an icon made from the animated GIF file.
        // The text and icon are centered by using JLabel.CENTER
        // The label is added to the content pane
        // of the applet (see Code Details).
        getContentPane().add(new JLabel(
            "Hello! Swing World",
            new ImageIcon("AnimSpeaker.gif"),
            JLabel.CENTER));
    }

    // When you run this program as an application.

    // The main method to instantiates the application frame.
    public static void main(String[] args) {
        // Create a Swing frame.
        JFrame f = new JFrame("THello");

        // Create an instance of the THello applet.
        THello helloApplet = new THello();

        // Initialize the applet instance.
        helloApplet.init();

        // Add the applet to the JFrame (to it's content pane).
        f.getContentPane().add(helloApplet);

        // Add the listener to close the frame.
        f.addWindowListener(new WindowEventHandler());

        // Assign the frame size and display it.
        f.setSize(300, 150); // frame: width = 300, height = 150
        f.show(); // Display the frame
    }
}

// Class to close the frame and exit the application.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
}
```

## 代码详析

如前所述,该程序既可以用做小程序,也可以作为应用程序。这样,就可以在同一程序中同时使用 init() 和 main() 这两种方法。其中 init() 方法包含创建一个文本标签“Hello! Swing World”和一个动画图标(正发送声波的扬声器)的语句。然后该标签被加入到小程序中。init() 方法在此程序中作为小程序方式运行时被调用。

在分析 init() 方法内的代码之前,应该注意到程序中出现的两个更重要的特征。如你所见,输入语句包含了 Swing 包 javajx.swing.\*。这是包含 Swing 类 JApplet 和 JFrame 的关键程序包。在整个 Swing 编程过程中,编程者会不断地在所有程序的顶端使用这个输入语句。

另一个有趣的地方是,类 THello 扩展了 JApplet。这是因为该程序是一个 Swing 小程序,所以它扩展了 Swing 小程序类 JApplet。在程序顶端的小程序标志 <Applet>...</Applet> 允许使用小程序查看器来运行程序 THello.java。这就避免了在测试程序时创建另一个名为 THello.html 的文件。注意,本书中的所有小程序都是按此惯例来测试的。

接下来,我们考察一下程序内部的 main() 方法,该方法一般用在应用程序的代码中。为了创建一个 Swing 应用程序,需要使用一个 JFrame 对象来包含这条语句。然而,在这个程序中,含有“Hello! Swing World”标题的小程序被加到了框架中。请看下面的代码语句:

```
// Create a Swing frame.  
JFrame f = new JFrame("THello");  
  
// Create an instance of THello applet.  
THello helloApplet = new THello();  
  
// Initialize the applet instance.  
helloApplet.init();  
  
// Add the applet to the JFrame (to its content pane).  
f.getContentPane().add(helloApplet);
```

其余的代码语句显示了具有指定大小的框架。该框架也具有窗口监听程序的特性,亦即在关闭框架时使得程序完全退出。最后一点,像 label 这样的组件被加到所谓的“内容窗格”里,而不是直接加到小程序或框架里。对于内容窗格的引用可以使用 getContentPane() 方法获得。在通读本书后就能学到更多有关内容窗格的知识了。

### 技巧:

可以将本程序(允许程序作为小程序或应用程序运行)中的外包装(Wrapper)代码用于其他需要类似功能的程序之中。

## 2.2 Swing 小程序集

Swing 集引入了它自己的小程序版本,其前身是 AWT 小程序。Swing 小程序的设计更加复杂,它能支持菜单条和在深度维数上的组件分层,并且允许在已经位于小程序内部的组件上

描绘。

由于 Swing 小程序支持菜单条,所以它可以拥有一个菜单条。组件的分层法本质上就是将组件重叠放置在多个层面上。这样就可指定组件放在某一层上,从而使其具有纵深感。Swing 小程序包含一个叫做分层窗格(layered pane)的专用组件来达到这一目的。

允许在已经位于小程序内部的组件上绘制的另一特性可以通过使用一个叫做玻璃窗格(glass pane)的组件来完成。玻璃窗格是一个透明组件,它使得背景组件是可见的。编程者可以在玻璃窗格上绘制文字或图像,并使得前景上的图像仍然是可见的。

Swing 小程序由类 JApplet 表示,该类存储在 javax.swing 程序包中。可以通过创建 JApplet 的扩展类来建立 Swing 小程序。当在其中增加子类时,需要将它们加入到所谓的内容窗格中,该窗格接受小程序的子类。Swing 小程序引入了这样一个中间容器,以便处理在小程序中混合轻量级组件和重量级组件所带来的复杂性。下面的代码语句显示了一个组件是怎样加到 Swing 小程序里的内容窗格上的:

```
/ Get a handle on the applet's content pane.  
Container contentPane = this.getContentPane();  
  
// You may want to assign a new layout.  
contentPane.setLayout(someLayoutObject);  
  
// Add the component to the applet.  
contentPane.add(someComponent);
```

通过调用小程序实例的以下几个方法,可以获取对于小程序分层窗格、玻璃窗格和菜单条的引用。

```
Public void getLayeredPane()  
Public void getClassPane()  
Public void getJMenuBar()
```

### 2.2.1 JApplet 程序示例

清单 2.2 TJApplet.java 示例演示了实现一个 Swing 小程序的过程,该程序显示了一个带边界的简单标签,这个程序产生的小程序结果见图 2-3。



图 2-3 Swing 小程序,显示一个带简单边界的标签

#### 清单 2.2 带简单边界的 Swing 小程序(TJApplet.java)

```
// Demonstrates the Swing applets  
/*
```

```
* <Applet code = TJApplet width = 200 height = 100>
* </Applet>;
*/
import javax.swing.*;
import java.awt.*;
public class TJApplet extends JApplet {
    public void init() {
        // 1. Get a reference to the content pane of the applet.
        Container contentPane = getContentPane();
        // 2. Create a label with an icon and text.
        JLabel label = new JLabel("This is a Swing applet", // text
            new ImageIcon("Metal.gif"), // icon
            JLabel.CENTER); // horiz. position
        // 3. Assign a matte border by using an icon and the specified insets.
        label.setBorder(BorderFactory.createMatteBorder(
            10, // top inset
            10, // left inset
            10, // bottom inset
            10, // right inset
            new ImageIcon("border.gif"))); // border icon
        // 4. Add the label to the applet's content pane.
        contentPane.add(label);
    }
}
```

## 代码详析

TJApplet 是 Swing 类 JApplet 的扩展类,从而使其自身成为一个 Swing 小程序。在 init() 方法里,程序片段 1 得到此小程序内容窗格的一个句柄引用。这个引用可用来将任意组件加到小程序内。

程序中的片段 2 创建了一个具有指定文本标签和图标的 Swing 标签。片段 3 给这个标签赋予了一个边框。该边框的类型是简单边界,并为这个不光滑模型使用了一个图标。标签边框四条边的宽度分别被赋予相应宽度(insets)。片段 4 将标签加到小程序中。注意,必须将标签加到小程序的内容窗格里,而不是小程序里。

## 第3章 事件处理

在响应用户输入时,用户接口元素激活事件。这些事件被传播给应用程序,然后由应用程序去执行特定的功能。如果没有定义功能,则该事件不产生任何效果。

JDK 1.1 中的 AWT 引入了代表事件模型来处理用户接口事件。由于 Swing 组件集是 AWT 的扩展,所以 Swing 组件也是基于这些代表事件模型而起作用的。

本章首先讨论了事件模型的概念,并介绍了由 Swing 组件所产生的高级事件。本章还解释了事件监听程序和用于创建事件监听程序的类。注意,大多数的 Swing 组件都使用了 AWT 的事件和事件监听程序类。由鼠标、键盘和其他设备所产生的底层事件将在后面的章节中讨论。

### 3.1 代表事件模型

代表事件模型支持明确划分内核程序与其用户界面之间的界限。由于在编译阶段做了更强的检查,因此代表模型就使得事件处理更为稳定而极不容易出错。此外,代表事件模型改进了事件处理的性能,因为这个工具包可以过滤掉不会执行任何功能的那些不期望出现的事件(比如频繁发生的鼠标事件)。

代表事件模型由三个分离的对象完成对事件的处理:事件源、事件以及事件监听程序。事件源是任何用户接口组件,例如按钮、文本域、滚动条以及滑动条等。事件源产生并引发事件,然后事件被广播出去,直到最后由它们的监听程序监听到为止。

事件是从 `java.util.EventObject` 类扩展出来的特定类的对象。

事件监听程序是从 `java.util.EventListener` 根接口扩展出来的实现特定事件监听接口的对象。监听类内部所实现的方法包含了操作用户接口组件时所需要执行的功能。

为了能够监听到事件源所产生的事件,监听对象必须以该事件源进行注册。这可用 `setEventTypeListener()` 方法来完成,或者更多的时候用 `addEventTypeListener()` 方法来完成,这是因为该函数支持多目广播。当一个源目标支持多目广播时,多个监听对象就可以同时登记或同时删除。

### 3.2 Swing 事件、源和监听对象

这一节将解释 Swing 源及其产生的事件,还将解释为了处理事件而实现的监听对象界面。

正如上一章中所讨论的,Swing 库支持很多复杂的接口组件,比如按钮、复选框、单选按钮、文本域等等。所有这些组件都会激活与其相应的监听类处理的事件。表 3.1 给出了 Swing 事件源及其相应的监听对象。

**说明:**

除了依赖于 Swing 库所支持的事件和监听对象外,Swing 组件也依赖于 AWT 事件及其监听对象。

**表 3-1 Swing 事件源及其相应的监听对象**

Swing 源	事件监听对象
AbstractButton JTextField Timer JDirectoryPane	ActionListener
JScrollBar	AdjustmentListener
JComponent	AncestorListener
DefaultCellEditor	CellEditorListener
AbstractButton DefaultCaret JProgressBar JSlider JTabbedPane JViewport	ChangeListener
AbstractDocument	DocumentListener
AbstractButton JComboBox	ItemListener
JList	ListSelectionListener
JMenu	MenuListener
AbstractAction JComponent TableColumn	PropertyChangeListener
JTree	TreeSelectionListener
JPopupMenu	WindowListener

附录 B“AWT 事件类快速参考”中列出了由这些事件类和监听对象界面封装的方法。在第 8 章至第 17 章介绍各种源组件的过程中,我们将详细讨论 Swing 组件所产生的特定事件及其相应的监听对象。

为处理某些特定的事件,需要创建一个监听对象来实现与之相关的监听接口。如下所示,下面的监听类提供了接口方法的代码:

```
class MyListener implements SomeEventListener {
    public void interfaceMethod1 (SomeEvent sel) {
```

```

    // functionality to be executed
}

public void interfaceMethod2 (SomeEvent se2) {
    // functionality to be executed
}

}

...
...
}

```

### 3.2.1 ActionEvent 代码示例

清单 3.1 演示了由按钮组件所产生的动作事件。该程序创建了一个 Swing 小程序，其上有一个按钮。这个按钮是产生 ActionEvent 类型的事件类的源对象。

当用户单击按钮时，就会执行 actionPerformed() 方法。该方法包含了按钮操作时所要执行的基本操作。在这个例子中，它所包含的功能是响铃。清单 3.1 的输出结果如图 3-1 所示。

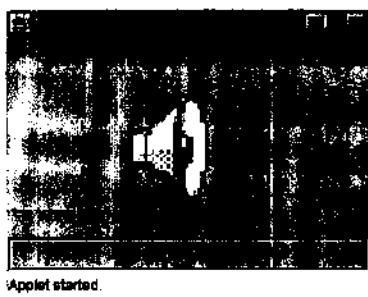


图 3-1 演示事件源、事件及其监听程序的 ActionEvent 小程序

清单 3.1 有一个按钮的 ActionEvent 小程序示例，用于演示源、事件及其对象 (TActionEvent.java)

```

/*
 * <Applet code = TActionEvent width = 300 height = 200 >
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TActionEvent extends JApplet {
    Container container;
    JLabel label;
    Icon speaker;

    public void init() {
        // 1. Get the handle on the applet's content pane.
        container = this.getContentPane();

```

```
// 2. Create a speaker icon, add it to a Swing
// label, and add the label to the applet.
speaker = new ImageIcon("speaker.gif");
label = new JLabel(speaker);
//label.repaint();
container.add(label);

// 3. Create a source (button) for the action event.
JButton source = new JButton("Ring the bell!");
container.add(source, BorderLayout.SOUTH);

// 4. Register the action listener with the source.
source.addActionListener(new ButtonListener());
}

// 5. Define the listener class.
class ButtonListener implements ActionListener {
    // 6. Interface method that has been implemented.
    public void actionPerformed(ActionEvent ae) {
        // Ring the bell...
        int i = 0;
        while (i < 10) {
            Toolkit.getDefaultToolkit().beep();
            try {
                Thread.currentThread().sleep(1000);
            } catch(InterruptedException ie) {
                System.out.println("Sleep Interrupted");
            }
            i++;
        }
    }
}
```

## 代码详析

小程序的开始处声明了几个对象作为其数据成员。在 Swing 小程序中,组件并非直接加到小程序中,而是要将它们加到小程序的基础内容窗格(underlying content pane)中。因此,代码片段 1 包含了对内容窗格的一个引用。代码片段 2 向其加了一个带有扬声器图标的装饰性标题框。

代码片段 3 创建了一个 Swing 按钮,用作产生 ActionEvent 类型事件的源对象。代码片段 4 中,源对象通过激活 addActionListener()方法由监听对象进行了登记。

代码片段 5 定义了实现 ActionListener 界面的监听类。这个类提供了界面中 actionPerformed()方法的实现。

formed()方法的代码。请注意,actionPerformed()方法以 ActionEvent 作为其参数。于是,按照源与其监听对象之间的这样一条通路,每当源按钮被按下的时候,actionPerformed()方法内的功能就得以执行了。

### 3.3 底层事件

所谓底层事件是指那些由鼠标和键盘等低级输入设备或者 GUI 组件上的窗口系统事件所产生的事件。表 3-2 包括了底层事件及其相应监听程序的列表。

表 3-2 包括了底层事件和监听程序

事件	监听程序
java.awt.event.ComponentEvent	java.awt.event.ComponentListener
java.awt.event.ContainerEvent	java.awt.event.ContainerListener
java.awt.event.FocusEvent	java.awt.event.FocusListener
java.awt.event.KeyEvent	java.awt.event.KeyListener
java.awt.event.MouseEvent	java.awt.event.MouseListener
	java.awt.event.MouseMotionListener
java.awt.event.WindowEvent	java.awt.event.WindowListener

附录 A“JFC Swing 快速参考”和附录 B 中给出了这些类中的 API 方法的快速参考。

#### 3.3.1 底层鼠标事件代码示例

清单 3.2 演示了底层鼠标事件。该代码创建了一个由鼠标监听对象注册的一个小程序。鼠标监听类实现了 MouseListener 接口。该小程序显示了鼠标进入和离开小程序时的状态。而且,如果单击、按下或释放鼠标按键,则鼠标按键的状态及鼠标的位置也会在该小程序上显示出来。程序的输出结果如图 3-2 所示。

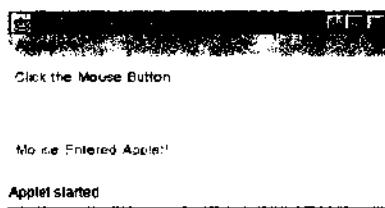


图 3-2 演示底层鼠标事件的 Swing 小程序

#### 清单 3.2 底层鼠标事件(TMouse.java)

```
// Demonstrates the mouse events (which are low-level events)

/*
 * <Applet code = TMouse width = 400 height = 200>
 * </Applet>
```

```
/*
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

// 1. Create a Swing applet that implements mouse listener.
public class TMouse extends JApplet
implements MouseListener {
    Container container;
    int width, height;
    int x, y;
    int flag;
    String mouseStatus;

    public void init() {
        // 2. Get a reference to the applet's content pane.
        container = this.getContentPane();

        // 3. Initialize the data members.
        x = 0; y = 0;
        width = 2; height = 2; // For a small square to be drawn
        flag = 0;

        // 4. Register the mouse listener with the applet.
        container.addMouseListener(this);
    }

    /*
     * NOTE: The class JApplet contains the update() method
     * to override the same method from the class
     * Component that repairs the container
     * background with its background color and calls
     * the paint() method. The update() in JApplet does not
     * repair the background of the container. It calls the
     * paint() method. You need to use an update() method that repairs
     * the background.
    */

    // 5. The update() method to repair the applet's background.
    public void update(Graphics g) {
        g.setColor(this.getBackground());
        g.fillRect(0, 0, // x and y coordinates
                  getWidth(), // Get the applet's width
                  getHeight()); // Get the applet's height
        paint(g);
    }

    // 6. The paint() method to paint the applet based on the flag.
```

```
public void paint(Graphics g) {
    g.setColor(Color.blue);
    g.drawString("Click the Mouse Button...", 5, 20); // at x = 5 & y = 20
    g.setColor(Color.red);
    if(flag == 1)
        g.drawString("Mouse Entered Applet!", 5, 80);
    else if(flag == 2)
        g.drawString("Mouse Exited Applet!", 5, 80);
    else if(flag == 3) {
        g.drawString("Mouse Entered Applet!", 5, 80);
        g.fillRect(x, y, width, height);
        g.drawString("Clicked Here!", x, y);
    }
    else if(flag == 4) {
        g.drawString("Mouse Entered Applet!", 5, 80);
        g.fillRect(x, y, width, height);
        g.drawString("Pressed Here!", x, y);
    }
    else if(flag == 5) {
        g.drawString("Mouse Entered Applet!", 5, 80);
        g.fillRect(x, y, width, height);
        g.drawString("Mouse Released!", x, y);
    }
}
```

7. Listener interface method, called when the mouse enters the applet.

```
public void mouseEntered(MouseEvent me) {
    flag = 1;
    repaint();
}
```

8. Listener interface method, called when the mouse exits the applet.

```
public void mouseExited(MouseEvent me) {
    flag = 2;
    repaint();
}
```

// 9. Listener interface method, called when the mouse button is clicked.

```
public void mouseClicked(MouseEvent me) {
    flag = 3;
    x = me.getX();
    y = me.getY();
    repaint();
}
```

```
}

// 10. Listener interface method, called when the mouse button is pressed.
public void mousePressed(MouseEvent me) {
    flag = 4;
    x = me.getX();
    y = me.getY();
    repaint();
}

// 11. Listener interface method, called when the mouse button is released.
public void mouseReleased(MouseEvent me) {
    flag = 5;
    x = me.getX();
    y = me.getY();
    repaint();
}

}
```

## 代码详析

代码片段 1 是小程序的开始。TMouse 类扩展了 JApplet, 它通过实现 MouseListener 接口而被用作鼠标监听程序。在 init() 方法内, 代码片段 2 获得了对小程序内容窗格的一个引用。代码片段 3 将数据成员初始化为适当的值。在小程序上有鼠标按键操作时, width 和 height 分别显示小矩形的宽度和高度。代码片段 4 中的语句用于向小程序注册鼠标监听对象(小程序本身)。

代码片段 5 是 update() 方法, 必须在此方法内完成当鼠标状态改变时对于小程序的一些修复工作。画一个填充了小程序背景色的矩形, 就可以将小程序界面完全清干净。这种方法会使系统性能降低, 从而导致屏幕出现闪烁。控制屏幕闪烁的有效办法是, 利用 Swing 窗格所提供的双缓冲。关于 Swing 窗格的内容将在第 4 章“框架、窗格和面板”中进行讨论。

代码片段 6 是小程序的 paint() 方法。每当鼠标状态发生改变时就会调用此方法。基于一个指示鼠标状态的特殊标志, 小程序就可以根据状态信息对窗口进行重绘。

代码片段 7~11 对鼠标状态标志进行赋值。这些代码片段实现了 MouseListener 接口的基本方法。mouseEntered() 和 mouseExited() 方法只是简单地对标志值做初始化并调用了 repaint() 方法。当鼠标进入或离开小程序时, 这些方法就会被激活。其他的方法(mouseClicked(), mousePressed() 和 mouseReleased()) 在执行鼠标操作时被激活。这些方法通过调用 MouseEvent 对象中的 getX() 和 getY() 方法取得鼠标的坐标值。这些信息将被用来在小程序上显示鼠标的各种活动。

### 3.3.2 底层鼠标移动代码示例

清单 3.3 演示了怎样处理底层鼠标移动事件——该事件是由鼠标的移动或拖动所产生的。为了识别通常的鼠标移动或鼠标键按下时的移动(拖动鼠标的情况), 必须处理 MouseMo-

`tionListener` 接口。监听程序通过定义接口方法 `mouseMoved()` 和 `mouseDragged()`。这个程序是一个小程序，它识别鼠标的移动状态并显示出此状态。程序的输出结果如图 3-3 所示。

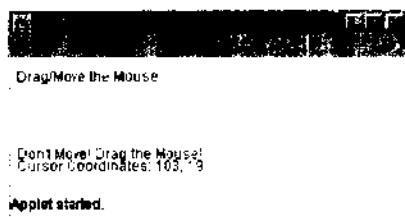


图 3-3 演示鼠标移动事件的小程序

### 清单 3.3 鼠标移动事件(TM MouseEvent.java)

```

// Demonstrates mouse motion events.

/*
* <Applet code = TMouseMotion width = 400 height = 200 >
* </Applet >
*/

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TMouseMotion extends JApplet {
    int x, y;
    int flag;

    public void init() {
        // 1. Create an object of the custom listener and register it with
        // the applet.
        CustomListener ct = new CustomListener(this);
        this.addMouseMotionListener(ct);
    }

    // 2. The update() method to repair the applet's background.
    // Warning: This can reduce the performance and lead to flickering.
    // The workaround is to use a JPanel attached to the applet.
    public void update(Graphics g) {
        g.setColor(this.getBackground());
        g.fillRect(0, 0, getWidth(), getHeight());
        paint(g);
    }

    // 3. The paint() method to display the mode of mouse motion
    // and its location, depending on the flag value.
    public void paint(Graphics g) {
        g.setColor(Color.blue);

```

```

g.drawString("Drag/Move the Mouse...", 5, 20);
g.setColor(Color.red);

if(flag == 1) {
    g.drawString("Don't Move! Drag the Mouse!", 5, 85);
    g.drawString("Cursor Coordinates: " + x + ", " + y, 5, 95);
}
else if(flag == 2) {
    g.drawString("Don't Drag! Move the Mouse!", 5, 85);
    g.drawString("Cursor Coordinates: " + x + ", " + y, 5, 95);
}
}

// 4. The mouse motion listener.
class CustomListener implements MouseMotionListener {

TMouseMotion tm;

public CustomListener(TMouseMotion tm) {
this.tm = tm;
}

// Executed when the mouse is moved.
public void mouseMoved(MouseEvent me) {

```

### 代码详析

在小程序的 init()方法内,代码片段 1 创建了一个鼠标移动监听对象。该对象被注册到小程序,以便小程序能够识别鼠标的移动。代码片段 2 是 update()方法,这个方法负责修复小程序的背景并调用 paint()方法。代码片段 3 定义了 paint()方法,该方法根据标志的取值画出鼠标的移动状态。

代码片段 4 定义了鼠标移动监听对象,该对象实现了 MouseMotionListener 接口。此时需要实现两个方法,即 MouseMoved()和 MouseDragged()方法。实现的代码需要设置鼠标移动或拖动的状态标志值,并获取鼠标光标的坐标。小程序调用 repaint()方法来画出鼠标坐标。

## 3.4 匹配类

大多数监听对象的接口都支持多个事件子类型(multiple event subtypes)。因此,它们都包含了多个需要由其监听类实现的抽象方法。例如,java.awt.event.MouseListener 接口就包含了 mouseClicked()、mouseEntered()、mouseExited()、mousePressed()和 mouseReleased()。

如果不想在一个接口内实现所有方法的功能,那就会有一个或更多的方法没有实际程序体。为避免空方法所带来的混乱,Swing 中引入了匹配类(adapter)。所谓匹配类是指那些用空方法实现了接口的类。在这种情况下,监听类只需简单地扩展适当的匹配类,并且仅仅实现

必要的方法去覆盖相应匹配类中的那些方法即可。这样就给出了处理那些必要事件的一种清晰的方法。下面是 Swing 监听类所用到的匹配类：

```
java.awt.event.ComponentAdapter  
java.awt.event.ContainerAdapter  
java.awt.event.FocusAdapter  
java.awt.event.KeyAdapter  
java.awt.event.MouseAdapter  
java.awt.event.MouseMotionAdapter  
java.awt.event.WindowAdapter
```

### 3.4.1 匹配类的代码示例

清单 3.4 演示了用于事件处理以及实现键盘监听对象的匹配类。这个小程序包括一个文本域，它是一个组件，可以在其中输入或显示单行文本。小程序显示时，文本域接受了输入焦点，可以在其中输入一些文字。所输入的文字只简单地由 Swing 的标签组件显示出来。

你会注意到，KeyListener 类扩展了 KeyAdapter 类，并且只实现了 KeyTyped()一个方法，它覆盖了匹配类中的那个方法。注意，如果监听类要实现 KeyListener 类，则必须实现 keyPressed()、keyReleased()和 keyTyped()这样三个方法。程序的输出结果如图 3-4 所示。

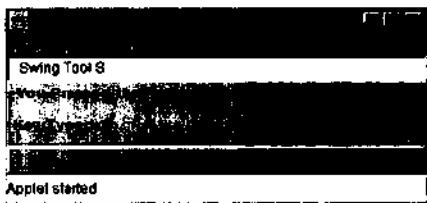


图 3-4 演示键盘匹配类的 Swing 小程序

#### 清单 3.4 用于事件处理的匹配类(*TKeyEvent.java*)

```
// Demonstrates adapters and key events.  
  
/*  
* <Applet code= TKeyEvent width= 350 height= 100 >  
* </Applet >  
*/  
  
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class TKeyEvent extends JApplet {  
  
    Container contentPane;  
    JLabel label;  
    JTextField textField;  
  
    public void init() {
```

```
// 1. Get the handle on the applet's content pane.  
contentPane = this.getContentPane();  
  
// 2. Create a text field and add a key listener to it.  
textField = new JTextField(25); // of 25 char width  
textField.addKeyListener(new MyKeyListener());  
  
// 3. Create a button object and register an action  
// listener.  
Button button = new Button("Clear");  
button.addActionListener(new ButtonListener());  
  
// 4. Create a label with the titled border.  
label = new JLabel("Key Typed: Null");  
label.setBorder(BorderFactory.createTitledBorder(  
    "You Pressed the Following Key"));  
  
// 5. Add the text field and button to the applet's  
// content pane.  
contentPane.setLayout(new BorderLayout());  
contentPane.add("North", textField);  
contentPane.add(label);  
contentPane.add("South", button);  
  
// 6. Get the focus on to the text field.  
// Note: You can do this only after you add  
// the text field to the container.  
textField.requestFocus();  
}  
  
// 7. Create the key listener class.  
class MyKeyListener extends KeyAdapter {  
    public void keyTyped(KeyEvent e) {  
        char c = e.getKeyChar();  
        label.setText ("Key Typed: " + c);  
    }  
}  
  
// 8. Create the button listener class.  
class ButtonListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        //Reset the text components.  
        textField.setText("");  
  
        //Return the focus to the text field.  
        textField.requestFocus();  
    }  
}
```

```
}
```

### 代码详析

在小程序的 init()方法内,代码片段 1 获得了一个对小程序内容窗格的引用参数。代码片段 2 创建了一个文本域并为它增加了一个监听对象。当用键盘在此文本域中输入文本时,键盘监听类就会作出响应。代码片段 3 创建了一个由动作监听对象注册的按钮。每次单击此按钮时,文本域就会被清空。

代码片段 4 创建了一个带有边框的标签。然后,在代码片段 5 中,所有组件都被添加到了小程序的内容窗格上。代码片段 6 中的语句说明,当小程序启动时,允许文本域接收输入焦点。

代码片段 7 中是键盘监听类,它实现了 KeyAdapter 匹配类。该类仅实现了 keyTyped()方法,该方法覆盖了匹配类中的同名方法。可以激活 KeyEvent 类型的对象上的 getKeyChar()方法,来取得所键入的字符。类似地,诸如修改键这样其他的一些按键可以利用 KeyEvent 类中的 API 方法来获得。代码片段 8 说明了动作监听程序 actionPerformed()方法的代码。当操作小程序上的按钮时,就会执行该方法。



# 第二部分 技术参考

第 4 章 框架、窗格和面板

第 5 章 布局管理器

第 6 章 Swing 设计

第 7 章 图标、标签和边界

第 8 章 按钮和复选框

第 9 章 列表和组合框

第 10 章 表格

第 11 章 文本小工具

第 12 章 滚动条和滑动条

第 13 章 菜单、工具条和工具提示

第 14 章 对话框和选项窗格

第 15 章 文件选择器和颜色选择器

第 16 章 树

第 17 章 定时器和进度指示器



## 第 4 章 框架、窗格和面板

本章将要讨论怎样实现 Swing 中的三类容器：框架、窗格和面板。下一章再说明怎样按照某个特定的布局要求将各种组件放置在容器里。

### 4.1 框架

Swing 框架是一个容器，其作用是作为使用 Swing 组件的程序主窗口。Swing 框架拥有一个标题、一个边界以及一些按钮，例如有的按钮用于将框架最小化成图标，最大化或关闭框架。

Swing 框架由 `JFrame` 类表示，它是 AWT 类 `Frame` 的扩展（见图 4-1）。注意，Swing 框架同其父类一样，是一个重量级容器。一个 `JFrame` 实例可以利用 `JFrame` 类和 `Frame` 类的特性。

#### 警告：

尽管可以在 `JFrame` 对象上调用 AWT 框架中的 API 方法，但是一定要小心可能出现不兼容的情况，比如将一个框架中的组件作不适当的打包等等。

Swing 中引入了内容窗格（content pane），用于解决轻量级组件和重量级组件一同工作时带来的复杂性。内容窗格基本上是完成 Swing 框架的内部管理，这就意味着可以将组件添加到框架的窗格上，而不是将它们直接加到框架本身。在本章讲解过程中，你将会学到更多有关窗格的内容。

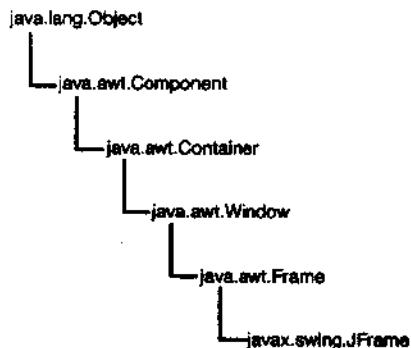


图 4-1 `JFrame` 类的层次结构图

#### 4.1.1 `JFrame` 构造函数

下面是一段典型的代码，它实现了 `JFrame` 对象并且向其上面添加了一个组件：

```
JFrame frame = new JFrame(); // Create a frame object
Container contentPane = frame.getContentPane(); // get the content pane
ContentPane.setLayout(new SomeLayout()); // if you want to assign
ContentPane.add(SomeComponent); // add the component
```

注意，内容窗格的缺省设置为有边界的布局。布局是用于在容器内显示组件的模型（关于布局以及怎样将布局赋予容器的有关内容，请参看第5章，“布局管理器”）。下面是 **JFrame** 类的构造函数：

```
public JFrame()
public JFrame(String title)
```

第一个构造函数创建了一个初始不可见、没有标题的新框架，第二个构造函数创建了一个不可见、但具有指定标题的框架。可以通过调用 `show()` 方法使框架成为可见的。关闭框架时，可以执行定义在 `WindowConstants` 接口中的如下两种关闭操作：

```
DO NOTHING ON CLOSE
DISPOSE ON CLOSE
```

第一个常数表示，当执行 `windowsClosing()` 方法时什么也不做。如需要，必须在 `windowsClosing()` 方法中处理必要的操作。第二个常数表示，调用任何注册的监听程序对象后会自动隐藏并且释放框架。

可以通过将这些常数作为参数来调用 `setDefaultCloseOperation()` 方法，将这些常数赋给一个框架。该方法的缺省参数是 `HIDE ON CLOSE`，该常量指示，简单地关闭框架。

#### 4.1.2 **JFrame** 代码示例

清单4.1给出了一个显示Swing框架的应用程序。该框架包含了一个标签，这是该框架的唯一组件。框架被注册到窗口监听对象，用于处理框架的关闭操作。程序的输出结果如图4-2所示。

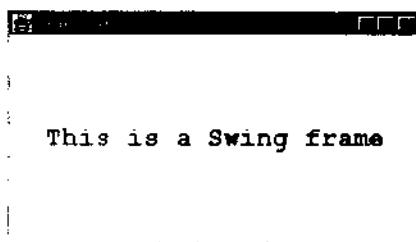


图4-2 带有一个标签的Swing框架(**JFrame**)

清单4.1 处理框架关闭动作的带有一个标签和窗口监听对象的**JFrame** (*TJFrame.java*)

```
Frame (TJFrame.java)
// Demonstrates the Swing frames.

import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;  
  
class TJFrame extends JFrame {  
    Container container = null;  
  
    // 1.Create a Swing frame.  
    public TJFrame(String title) {  
        super(title); // set the title for the frame  
  
        // 2.Get the content pane of the frame and  
        // configure the frame using its content pane.  
        container = this.getContentPane();  
        container.setBackground(Color.white);  
        container.setForeground(Color.blue);  
  
        // 3.Create a label and add it to the frame.  
        JLabel label = new JLabel("This is a Swing frame",JLabel.CENTER);  
        label.setFont(new Font("Sans",Font.PLAIN,22));  
        container.add(label);  
  
        // 4.Add the window listener to close the frame  
        // and display it with the specified size.  
        addWindowListener(new WindowEventHandler());  
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
        setSize(350,200); // width = 350,height = 200  
        show(); // Display the frame  
    }  
  
    // 5.The main method.  
    public static void main(String[] args) {  
        TJFrame frame = new TJFrame("Swing Frame");  
    }  
  
    // 6.The listener class to handle closing of the frame.  
    class WindowEventHandler extends WindowAdapter {  
        public void windowClosing(WindowEvent evt) {  
            System.exit(0);  
        }  
    }  
}
```

### 代码详析

TJFrame 是 Swing 框架类 JFrame 的扩展类。代码片段 1 显示了 TJFrame 类的构造函数。可以用 super() 调用构造函数 JFrame (title) 来给框架赋予一个标题。代码片段 2 取得一个对于框架中内容窗格的引用。然后，它修改了框架的背景色和前景色。

代码片段 3 创建了一个标签对象，并利用内容窗格把它加到 Swing 框架的中央。代码片段 4 将窗口监听对象加到框架中来执行框架的关闭操作。这个片段也给出了用 show() 方

法进行框架大小调整和框架显示的代码。另一种方法 `setDefaultCloseOperation()` 则指定关闭框架时所需进行的操作。在这种情况下，关闭框架的操作就会转交给其他的方法去完成。代码片段 5 给出了主方法，代码片段 6 则给出了处理框架关闭操作的监听程序类。

## 4.2 Swing 的基本容器

Swing 中引入了 4 个新的组件，用于处理在同一个容器内混合使用轻量级组件和重量级组件时所造成的复杂情况。这 4 个新组件是内容窗格、分层窗格、玻璃窗格和一个可选的菜单条。另外，根窗格是一个虚拟容器，它包含了内容窗格、分层窗格、玻璃窗格和菜单条。

### 4.2.1 根窗格

像 `JApplet`、`JFrame`、`JWindow`、`JDialog` 以及 `JInternalFrame` 这样的 Swing 容器代表了它们对于根窗格所担负的责任，这个根窗格是由 `JRootPane` 类表示的。注意，`JInternalFrame` 是轻量级容器，其他几个组件则是重量级的。由于根窗格是由内容窗格、分层窗格、玻璃窗格和菜单条构成的，因此内容必须加到这些根窗格成员之一，而不是加到根窗格本身。图 4-3 描述了这些根窗格成员在根窗格中所处的位置。可以利用 `getRootPane()` 方法取得对容器潜在根窗格的引用。

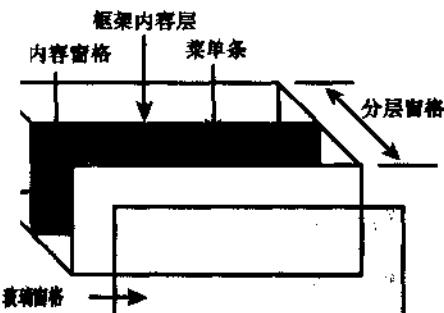


图 4-3 根窗格类的组件

根窗格及其成员在 Swing 容器设计中都被认为是 basic 容器。本章将要讨论内容窗格、玻璃窗格和分层窗格。至于菜单条的实现，则放在第 13 章“菜单、工具条和工具提示”中详细讨论。

#### 说明:

根窗格类不能有任何子类。因此，不能向根窗格对象中加入任何组件，而应该向根窗格成员中加入组件。

### 4.2.2 内容窗格

内容窗格实际上位于分层窗格的某一层上。分层窗格包含了若干层，这些层根据覆盖要求的不同，分别显示不同的 Swing 组件。图 4-3 表明了内容窗格在分层窗格中所处的位置。内容窗格的位置是由分层窗格常数 `FRAME_CONTENT_LAYER` 确定的。事实上，菜单条

和内容窗格是位于框架 - 内容层上的。

大多数 Swing 组件都加到了内容窗格上。在前面 JFrame 示例中已经显示了组件加到内容窗格上的处理方式。下面这个典型的代码段用来将组件加入到具有特定布局的内容窗格中：

```
Container contentPane = swingContainer.getContentPane();
contentPane.setLayout(new SomeLayout());
contentPane.add(new SomeComponent());
```

**说明：**

内容窗格缺省地设置为带边界的布局管理器。

### 4.2.3 玻璃窗格

玻璃窗格是根窗格的成员，用于在某个已经包含有一些组件的区域上进行描绘。玻璃窗格也可以用来捕捉鼠标事件，因为它位于内容窗格和菜单条之上。图 4-3 给出了玻璃窗格在根窗格内所处的位置。由于玻璃窗格是根窗格的成员，所以它也就存在于某个 Swing 容器之中。但缺省情况下，玻璃窗格是不可见的。

**说明：**

通过调用玻璃窗格对象上的 setVisible (true) 方法，可以使玻璃窗格成为可见的。调用玻璃窗格对象上的 getGlassPane () 方法则可以取得对玻璃窗格的引用。

### 4.2.4 菜单条

菜单条添加在菜单条区域内，该区域是一个很窄的长方形区域，位于分层窗格中的框架内容窗格这一层的上部边缘。其余的区域由内容窗格占据。图 4-3 给出了菜单条在分层窗格内所处的位置。为取得对 Swing 框架上的选项菜单条的引用，可以调用容器对象上的 getJMenuBar () 方法。有关创建菜单条的详细内容，请参看第 13 章。

### 4.2.5 分层窗格

分层窗格是一种容器，它包含有多个层面，每一层可用来加入处于不同深度（第 3 维上）的一类组件。组件的深度叫做它的 Z - 序定位（Z - order positioning）。用 Integer 类型的对象指定要加入的组件所处的深度。深度整数值大的组件位于深度整数值小的组件之上。位于这些层上的组件可以相互重叠。

分层窗格对象用 Swing 的 JLayeredPane 对象表示，它扩展了 JComponent 类，如图 4-4 所示。这个类存放在 javax.swing 包中。

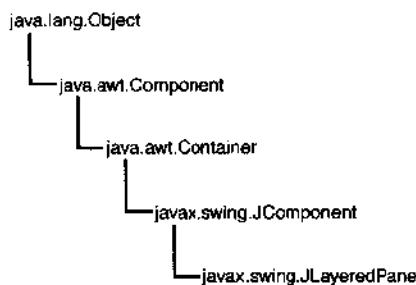


图 4-4 JLAYEREDPANE 类的层次结构图

#### 4.2.6 关于层的详细论述

除了用整数指定组件的深度外，还可以使用由静态常数定义的层的名字指定组件的深度。下面定义了 5 个便于使用的层：

- 缺省层 — 这一层所处的深度由 new Integer(0) 对象指定。常数 DEFAULT\_LAYER 表示该层。顾名思义，这是一个标准层，缺省情况下，任何组件都会被加入到该层。缺省层也是标准层的最底下一层。
- 调色板层—调色板层由常数 PALETTE\_LAYER 表示，它所处的深度由 new Integer(100) 对象指定。该层位于缺省层之上。调色板层对于添加调色板或者浮动工具条是很有用的。正如第 13 章中所述，工具条是可浮动的。
- 模式层—模式层位于调色板层之上，由常数 MODAL\_LAYER 表示。它所处的深度由 new Integer(200) 对象指定。模式层对于模式对话框的位置是很有用的。
- 弹出层—弹出层有助于显示类似于弹出式菜单、工具提示等一些对象，而这些对象应该出现在位于前面所讨论的那些层内的组件上方。弹出层由常数 POPUP\_LAYER 表示，它的深度值是 new Integer(300)。
- 拖动层—拖动层由常数 DRAG\_LAYER 表示，其深度值是 new Integer(400)。当拖动一个组件时，会把该参数重新赋给拖动层，从而保证它位于其他所有组件之上。拖放完成时，该组件又会被赋给其原来所处的层上。

除了上述几层以外，还有一个特殊的层被称为框架 ~ 内容层，用来放置根窗格中的内容窗格和选项菜单。这个层由常数 FRAME\_CONTENT\_LAYER 表示，它的深度值是 new Integer(~30000)。

#### 4.2.7 JLAYEREDPANE 代码示例

清单 4.2 给出了一个演示 Swing 分层窗格的程序。该应用程序上有一些按钮，操作某个按钮时，它就会向分层窗格的一个特定层中加入一个内部框架。程序中还有一个 Clear 按钮，负责清除主框架中的所有内部窗格。程序的输出结果如图 4-5 所示。

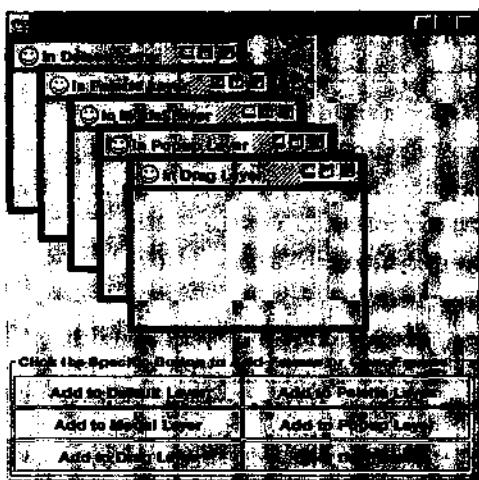


图 4-5 附在 Swing 分层窗格的内部框架 (JLayeredPane)

**清单 4.2 附加了按钮和内部框架的 JLayeredPane (TJLayeredPane.java)**

```
// Demonstrates the Swing layered panes.  
import javax.swing.*;  
import javax.swing.border.*;  
import java.awt.*;  
import java.awt.event.*;  
import java.util.*;  
  
public class TJLayeredPane extends JFrame  
    implements ActionListener {  
    JButton button;  
    JLayeredPane layeredPane;  
  
    static int frameCount = 0;  
    static final int xOffSet = 25; // each layer pane x offset  
    static final int yOffSet = 25; // each layer pane y offset  
  
    int numFrames = 1;  
    // Define arrays for the layered pane constants and their names.  
    Integer[] layerConstants = { JLayeredPane.DEFAULT_LAYER,  
        JLayeredPane.PALETTE_LAYER, JLayeredPane.MODAL_LAYER,  
        JLayeredPane.POPUP_LAYER, JLayeredPane.DRAG_LAYER };  
  
    String[] layerNames = { "Default Layer", "Palette Layer",  
        "Modal Layer", "Popup Layer",  
        "Drag Layer" };  
    Vector framesVector = new Vector();  
  
    // Constructor.  
    public TJLayeredPane() {
```

```
super("TUIlayeredPane"); // Assign a title to the frame.  
// 1.Create a panel and assign grid layout with 3 row and 3 columns.  
JPanel panel = new JPanel(new GridLayout(3,3));  
panel.setBorder( // Assign a title border around the panel.  
    BorderFactory.createTitledBorder(  
        "Click the Specific Button to Add Frames or Clear Frames"));  
// 2.Add the following control buttons to the panel.  
for (int i=0; i<layerNames.length+1; i++) {  
    if (i < layerNames.length) {  
        button = new JButton("Add to "+layerNames[i]);  
        button.setActionCommand(layerNames[i]);  
    } else {  
        button = new JButton("Clear the Frames");  
        button.setActionCommand("Clear");  
        button.setForeground(Color.red);  
    }  
    button.addActionListener(this);  
    panel.add(button);  
}  
// 3.Add the panel at the bottom portion of the frame.  
getContentPane().add(panel,BorderLayout.SOUTH);  
// 4.Obtain a handle on the layered pane of the frame.  
layeredPane = getLayeredPane();  
// 5.Code to configure the frame.  
addWindowListener(new WindowEventHandler());  
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
setSize(400,400); // width=400,height=400  
show(); // Display the frame  
}  
  
// 6.Whenever a button is clicked.  
public void actionPerformed(ActionEvent e) {  
    JButton tempButton = (JButton) e.getSource();  
    if (tempButton.getActionCommand() == layerNames[0]) { // for default layer.  
        addInternalFrame(0);  
    }  
    else if (tempButton.getActionCommand() == layerNames[1]) { // for palette layer.  
        addInternalFrame(1);  
    }  
    else if (tempButton.getActionCommand() == layerNames[2]) { // for Modal layer.  
        addInternalFrame(2);  
    }  
    else if (tempButton.getActionCommand() == layerNames[3]) { // for pop-up layer.  
        addInternalFrame(3);  
    }  
}
```

```
}

else if (tempButton.getActionCommand() == layerNames[4]) { // for Drag layer.
    addInternalFrame(4);
}

else if (tempButton.getActionCommand() == "Clear") {
    //Enumerate the vector elements and dispose all of them.
    for (Enumeration enum = framesVector.elements();
         enum.hasMoreElements();)
        ((JInternalFrame) enum.nextElement()).dispose();
}

// Set the frame count to zero.
frameCount = 0;
}

}

// 7. Adds an internal frame to the specified layer.
public void addInternalFrame(int index) {
    JInternalFrame iFrame = new JInternalFrame(
        "In " + layerNames[index],
        true, // can be resized
        true, // can be closed
        true, // can be maximized
        true); // can be iconified

    // 8. Register the new frame in a vector.
    framesVector.addElement(iFrame);

    // 9. Set the location of the frame.
    iFrame.setLocation (xOffSet * (frameCount),
                        yOffSet * (frameCount++));
    iFrame.setSize(200,150); // Assign suitable size
    iFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    iFrame.setFrameIcon(new ImageIcon("smiley.gif"));
    layeredPane.add(iFrame,layerConstants[index]);
}

// 10. The listener class to handle closing of the frame.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}

// 11. The main method.
public static void main(String[] args) {
    TUILayeredPane frame = new TUILayeredPane();
}
```

## 代码详析

程序 TJLayeredPane 中为分层窗格常数和各层的名字定义了相应的数组作为其数据成员。在其构造函数内，代码片段 1 创建了一个窗格来放置一些控制按钮，这些按钮用以创建内部框架并把它们加到分层窗格内各层。代码片段 2 将这些按钮加到窗格上。按钮上包含了各层的名字作为其标签。当单击某个特定按钮时，内部框架就会加到相应的层上。代码片段 3 把窗格加到了应用程序框架的底部。代码片段 4 取得一个与主框架相关联的分层窗格的引用。代码片段 5 给出了用于配置主框架的代码语句。

代码片段 6 是 actionPerformed() 方法，单击按钮时就会被激活。正如你所看到的，当单击某个特定按钮时，利用 if 条件语句，可以将一个内部框架加到与之对应的层中。当单击 Clear 按钮时，所有已经创建的内部框架就会从主框架上清除掉。

代码片段 7 是创建内部框架并将其加到某个特定层上的实际代码。代码片段 8 在一个向量中注册每一个新的内部框架。这些信息在必要的时候用于清除所有的内部框架。代码片段 9 定义了每一个内部框架的位置，当新创建的框架下面已经有框架时，就需要确定新框架的偏移位置。代码片段 10 给出了处理框架关闭动作的监听类。代码片段 11 是创建主框架对象的主方法。

## 4.3 内部框架

内部框架是一个容器，它与 Swing 框架类似，但有一点很重要的差别：内部框架是轻量级的。内部框架支持很多特性，比如拖放、关闭、图标化（最小化）以及大小调整等，内部框架也可以拥有标题和菜单条。内部框架常用于在程序中初始化一个新的任务。例如，在字处理程序中，当开始编辑一份新的文档时，它就会被赋给一个内部框架。

内部框架用 JInternalFrame 类表示，它扩展了 JComponent 类，如图 4-6 所示。JInternalFrame 类存放在 javax.swing 程序包中。

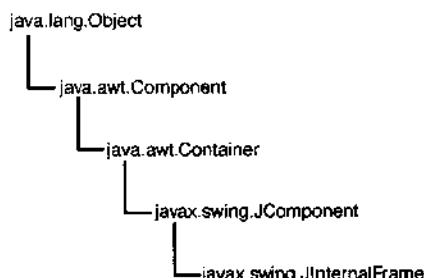


图 4-6 JInternalFrame 类的层次图

### 4.3.1 JInternalFrame 类构造函数

下面是 JInternalFrame 类所支持的构造函数列表：

```
public JInternalFrame()
```

这个构造函数创建一个不带标题的 JInternalFrame 对象。此框架不能改变大小，不能关闭，不能最大化，也不能最小化。

```
public JInternalFrame (String title)
```

这个构造函数创建一个带标题的 JInternalFrame 对象。此框架不能改变大小，不能关闭，不能最大化，也不能最小化。

```
public JInternalFrame (String title, boolean resizable)
```

这个构造函数创建一个带标题的内部框架。可以指定此框架是否能够改变大小，但此框架不能关闭，不能最大化，也不能最小化。

```
public JInternalFrame (String title,  
                      boolean resizable,  
                      boolean closable)
```

这个构造函数创建一个带标题的内部框架。可以指定此框架是否能够改变大小，以及是否能够关闭。然而，此框架不能最大化，也不能最小化。

```
public JInternalFrame (String title,  
                      boolean resizable,  
                      boolean closable,  
                      boolean maximizable)
```

这个构造函数创建一个带标题的内部框架。可以控制它能否改变大小，能否关闭，能否最大化。

```
public JInternalFrame (String title,  
                      boolean resizable,  
                      boolean closable,  
                      boolean maximizable,  
                      boolean iconifiable)
```

这个构造函数创建一个带标题的 JInternalFrame 对象。它可以控制自己能否改变大小，能否关闭，能否最大化。

## 4.4 桌面窗格

桌面窗格是特殊类型的分层窗格，用来在 Java 应用程序中创建虚拟的桌面。桌面窗格的主要目的是为了添加内部框架。桌面窗格负责管理重叠的内部框架并根据其内容将它们按正确的顺序放置。桌面窗格也支持内部框架的桌面管理器。可以用它的构造函数 JDesktopPane () 创建桌面窗格。

### 4.4.1 桌面管理器

桌面管理器负责管理桌面窗格和内部框架的外观效果。接口函数 DesktopManager 给出

了桌面管理器的体系结构。这个接口对象实现了桌面窗格的最小化、最大化、图标化以及从图标状态恢复成窗口状态等操作的代码。桌面窗格还包括了一个桌面管理器对象的实现。

由于内部框架是由桌面窗格管理的，因此，对于桌面管理器的操作也施加到了内部框架上。DefaultDesktopManager 类是 DesktopManager 接口的缺省实现。缺省管理器对象可以管理位于其任何父类中的桌面的外观效果。

#### 4.4.2 使用 JInternalFrame 类的桌面窗格代码示例

清单 4.3 给出了一个应用程序示例，该程序创建了一个 Swing 框架，该框架底部有一个按钮，框架的中央显示了一个内部框架。每次单击按钮时，程序会再创建一个内部框架，并在相对于前一个内部框架的某个偏移位置上，在主框架中显示内部框架。程序的输出结果如图 4-7 所示。

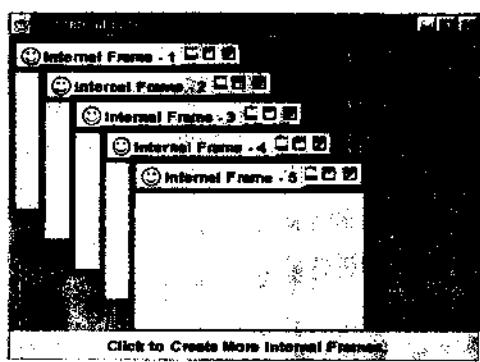


图 4-7 显示内部框架的桌面窗格

清单 4.3 JInternalFrame 示例——显示带有内部框架的桌面窗格 (*TJInternalFrame.java*)

```
// Demonstrates the Swing internal frames.

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class TJInternalFrame extends JFrame {
    Container container;
    JButton button;
    JDesktopPane desktop;
    JInternalFrame internalFrame;

    static int frameCount = 1;
    static final int xOffset = 25;
    static final int yOffset = 25;

    // 1. Constructor of the frame class.
    public TJInternalFrame() {
        // 2. Give a title to the frame and get its content pane.
```

```
super("JInternalFrame");
container = this.getContentPane();

// 3.Create a button and add it at the lower portion of the
// frame; also add an action listener.
button = new JButton("Click to Create More Internal Frames");
button.addActionListener(new ButtonListener());
container.add(button, BorderLayout.SOUTH);

// 4.Create a desktop pane and add an internal frame.
desktop = new JDesktopPane(); // holds the internal frame
container.add(desktop); // add the desktop to the main frame
createInternalFrame(); // create an internal frame

// 5.Add the window listener, set the frame size, default close
// operation and make the frame visible.
addWindowListener(new WindowEventHandler());
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
setSize(400,300); // width = 400,height = 300
show(); // Display the frame
}

// 6.Creates an internal frame and adds it to the desktop pane.
// Takes care of displaying frames with overlap offsets when called
// multiple times.
public void createInternalFrame() {
    // 7.Use a suitable internal frame constructor.
    JInternalFrame iFrame = new JInternalFrame(
        "Internal Frame - " + (frameCount
        ),
        true, // can be resized
        true, // can be closed
        true, // can be maximized
        true); // can be iconified

    // 8.Set the location and size, and add it to the desktop pane.
    iFrame.setLocation(xOffset * (frameCount - 2),
                      yOffset * (frameCount - 2));
    iFrame.setSize(200,150);
    desktop.add(iFrame);

    // 9.Let the frame be selected.
    try {
        iFrame.setSelected(true);
    } catch (java.beans.PropertyVetoException ex) {
        System.out.println(
            "Exception while selecting an internal frame");
    }
}
```

```

}

}

// 10. The button (action) listener.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        createInternalFrame();
    }
}

// 11. The listener class to handle closing of the frame.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}

// 12. The main method.
public static void main(String[] args) {
    TJInternalFrame frame = new TJInternalFrame();
}
}

```

### 代码详析

这个程序创建了带有标题 `TJInternalFrame` 的主框架。代码片段 1 给出了主框架构造函数，代码片段 2 为主框架设置了标题并取得其内容窗格句柄。代码片段 3 创建了一个按钮并把它添加到框架底部，还注册了按钮的动作监听程序。

代码片段 4 创建了桌面窗格，以添加内部框架。此内部框架是由 `createInternalFrame()` 方法创建的。代码片段 5 添加了一个窗口监听程序用于处理关闭框架的操作，这段代码还通过设置大小及缺省的关闭操作对主框架进行了配置，代码段最后使此框架成为可见的。

代码片段 6 – 9 创建内部框架并把该内部框架加入到桌面窗格内。内部框架放在特定位上。每次调用 `createInternalFrame()` 方法时，都会创建一个新的内部框架，并以一定的偏移位置把它添加到桌面窗格上。该方法同时还选定被创建的新内部框架。注意，`setSelected()` 会抛出 `PropertyVetoException` 异常，所以需要将该方法放在 `try...catch` 语句内。

代码片段 10 是按钮监听程序，它激活了 `createInternalFrame()` 方法。代码片段 11 是关闭框架的窗口监听程序。代码片段 12 给出了主方法。

## 4.5 面板

面板是一个 Swing 容器（它是 `JComponent` 组件的扩展），常用来组合小程序或框架某个区域内的若干组件。面板还可以用来组合其他的面板。

Swing 面板支持双缓冲，这种技术有助于避免出现动画中的闪烁问题。如果面板在对象显示之前被设置成了双缓冲，那么该对象就会先写入屏幕缓冲区之外，然后再被快速地翻转

到面板上。这是一种处理计算密集型显示问题的有效技术。

Swing 面板由 JPanel 类所表示，这个类存放在 javax.swing 程序包中。JPanel 类的层次结构如图 4-8 所示。注意 JPanel 类是一个轻量级容器。

#### 说明：

缺省情况下，Swing 面板采用双缓冲技术，并且设置成流式布局结构。

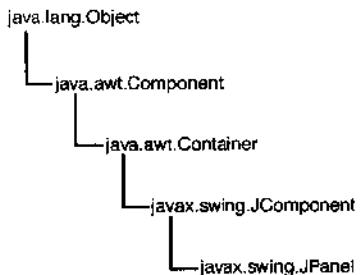


图 4-8 Swing 窗格类 JPanel 的层次结构图

#### 4.5.1 JPanel 类构造函数

Swing 面板对象可以用 JPanel 类中的四个构造函数之一来创建。构造函数既提供了面板布局，又提供了对双缓冲区的控制。下面是构造函数代码：

```
public JPanel()
```

这个构造函数创建一个具有流式布局的双缓冲面板。

```
public JPanel(boolean isDoubleBuffered)
```

该构造函数创建一个流式布局面板；然而，可以通过传递参数 true 来控制该面板为双缓冲面板，或者传递参数 false 来控制该面板为非双缓冲的。

```
public JPanel(LayoutManager layout)
```

这个构造函数创建一个双缓冲面板，它允许你将感兴趣的布局结构赋给它。

```
public JPanel(LayoutManager layout, boolean isDoubleBuffered)
```

这个构造函数以一个特定的布局结构创建面板。能够通过传递参数 true 或 false 来控制该面板是否为双缓冲的。

#### 4.5.2 JPanel 代码示例

清单 4.4 给出了一个小程序，它使用 4 个面板显示一些标签（参见图 4-9）。每个面板都设置成了网格布局，并缺省地设置为双缓冲。注意，面板是用来组合组件的，其用法在代码注释中作了说明。

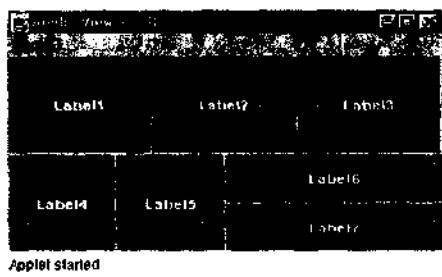


图 4.9 显示 Swing 面板的小程序

## 清单 4.4 用 JPanel 创建带有标签的四个面板 (TJPanel.java)

```
// Demonstrates the Swing panels.

/*
* < Applet code = TJPanel width = 400 height = 200 >
* </Applet>
*/

import javax.swing.*;
import java.awt.*;

public class TJPanel extends JApplet {
    Container container = null;

    public void init() {
        // 1. Get a reference to the applet's content pane and
        // assigns a grid layout.
        container = getContentPane();
        container.setLayout(new GridLayout(2,1,2,2));

        // 2. Create four panels with grid layout and specified number
        // of rows and columns. The first two numbers in the
        // GridLayout constructors are the number of rows and columns
        // The next two numbers represent the horizontal and vertical
        // spacing between the components.
        JPanel panel1 = new JPanel(new GridLayout(1,3,2,2));
        JPanel panel2 = new JPanel(new GridLayout(1,2,2,2));
        JPanel panel3 = new JPanel(new GridLayout(2,1,2,2));
        JPanel panel4 = new JPanel(new GridLayout(1,2,2,2));

        // 3. Prepare panel1 with labels 1,2, and 3.
        setLabel(panel1,"Label1");
        setLabel(panel1,"Label2");
        setLabel(panel1,"Label3");

        // 4. Prepare panel2 with labels 4 and 5.
        setLabel(panel2,"Label4");
        setLabel(panel2,"Label5");
    }

    void setLabel(JPanel panel, String label) {
        Label l = new Label(label);
        panel.add(l);
    }
}
```

```
// 5. Prepare panel3 with labels 6 and 7.  
setLabel(panel3, "Label6");  
setLabel(panel3, "Label7");  
  
// 6. Add panel2 and panel3 to panel4.  
panel4.add(panel2);  
panel4.add(panel3);  
  
// 7. Finally, add panel1 and panel4 to the content pane.  
container.add(panel1);  
container.add(panel4);  
  
}  
  
// 8. Supporting method to create and attach a label.  
public void setLabel(Container panel, String text) {  
    JLabel label = new JLabel(text, JLabel.CENTER);  
    label.setOpaque(true);  
    label.setForeground(Color.white);  
    label.setBackground(Color.gray);  
    panel.add(label);  
}
```

### 代码详析

代码片段 1 取得其小程序内容窗格的引用，并且向程序赋予了网格布局。代码片段 2 创建了四个面板，每个面板都定义了行数和列数的网格布局。

代码片段 3–5 准备了已定义好标签的窗格。代码片段 8 所示的方法 setLabel() 则完成了这个操作。代码片段 6 将面板 2 和面板 3 加到了面板 4 上。代码片段 7 又将面板 1 和面板 4 加到了小程序的内容窗格上。

## 4.6 拆分窗格

拆分窗格是一个轻量级的 Swing 容器，它将置于其中的组件从图形形式上作了划分。拆分窗格只能包含两个组件。每个组件的显示区域都可以通过应用程序用户进行交互式调整。如果要显示两个以上被划分开的组件，就需要拆分窗格和组件。

Swing 拆分窗格由 JSplitPane 类表示。如图 4-10 所示，这个类扩展了 JComponent 类，它存放在 javax.swing 程序包中。

拆分窗格有两种类型：水平拆分窗格和垂直拆分窗格。水平拆分窗格用一个拆分条水平地分割两个组件。也就是说，两个组件水平方向相邻，中间被一个可调整的拆分条隔离开来。拆分条可以沿水平方向移动。可以利用静态整型常数 HORIZONTAL\_SPLIT 来为水平拆分类指定拆分窗格对象。

在支持垂直拆分的窗格中，组件是垂直放置的（一个位于另一个之上）。拆分条可以沿垂直方向移动。可以利用静态整型常数 VERTICAL\_SPLIT 来为垂直拆分类指定拆分窗格

对象。

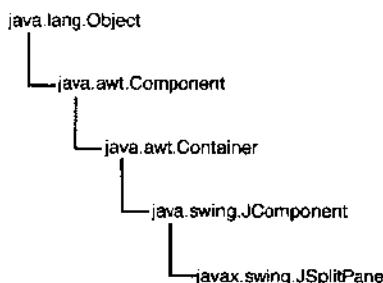


图 4-10 JSplitPane 类的层次或结构图

#### 4.6.1 JSplitPane 类构造函数

JSplitPane 类提供了五个构造函数。每个构造函数允许指定不同属性的拆分窗格对象。可以指定拆分的方向、要加入的子组件，以及子组件大小的改变是否连续，等等。下面是构造函数代码：

```
public JSplitPane()
```

这个构造函数创建一个对象，并设置该对象的子组件一个挨一个地水平排列，非连续布局，并且每个组件都各有两个按钮。

```
public JSplitPane(int newOrientation)
```

这个构造函数创建一个具有指定方向并且非连续布局的对象。参数 newOrientation 取以下值之一：

```
static int HORIZONTAL_SPLIT
static int VERTICAL_SPLIT

public JSplitPane (int newOrientation
                  boolean newContinuousLayout)
```

该构造函数创建一个 JSplitPane 对象，该对象具有指定方向和子组件尺寸改变样式。第一个参数与上一个构造函数中的参数一样。第二个参数指定子组件尺寸是否随拆分条位置的改变而连续变化（参数为 true 或 false）。

```
public JSplitPane(int newOrientation
                  Component newLeftComponent,
                  Component newRightComponent)
```

该构造函数创建一个具有指定方向和指定组件的对象。子组件的绘制不是连续的。

```
public JSplitPane(int newOrientation
                  boolean newContinuousLayout,
                  Component newLeftComponent,
```

```
Component newRightComponent)
```

该构造函数创建一个 JSplitPane 对象，该对象具有特定的子组件及其子组件的方向和可改变尺寸的类型。

#### 4.6.2 连续布局属性

随着水平拆分窗格或垂直拆分窗格中拆分条的调整，在最终位置上停止之前，子组件的尺寸可以随着拆分条每一个位置的改变而连续地变化，或者只是在最终位置处才停止。拆分窗格的这一性质叫做连续布局属性。

连续布局属性可以开启或关闭。当传递给 setContinuousLayout() 方法的布尔参数值为 true 时，子组件的尺寸就可以改变。如果上述布尔参数值为 false，子组件的尺寸仅在拆分条移到最终位置时才改变。

#### 4.6.3 JSplitPane（水平或垂直拆分窗格）代码示例

清单 4.5 给出了一个小程序，它使用 2 个嵌套的拆分窗格（一个是水平的，另一个是垂直的）来显示三幅图像。之所以要用 2 个拆分窗格，是因为一个拆分窗格只能包含两个组件。可以通过调整拆分条，来改变图像的大小。程序的输出结果如图 4-11 所示。



图 4-11 在拆分窗格中显示图像的 Swing 小程序

清单 4.5 具有一个水平拆分窗格和一个垂直拆分窗格的 JSplitPane (TJSplitPane.java)

```
/*
 * <Applet code = TJSplitPane width = 400 height = 300 >
 * </Applet >
 */

import javax.swing.*;
import java.awt.*;

public class TJSplitPane extends JApplet {
    static int HORIZSPLIT = JSplitPane.HORIZONTAL_SPLIT;
    static int VERTSPLIT = JSplitPane.VERTICAL_SPLIT;
```

```
boolean continuousLayout = true;
Icon icon1 = new ImageIcon("saravan2.gif");
Icon icon2 = new ImageIcon("saravan3.gif");
Icon icon3 = new ImageIcon("ISaravan.gif");

public void init() {
    // 1. Create a label to display icon1 and add it to a scroll pane.
    JLabel label1 = new JLabel(icon1);
    JScrollPane topLeftComp = new JScrollPane(label1);

    // 2. Create another label to display icon2 and add it to another scroll pane.
    JLabel label2 = new JLabel(icon2);
    JScrollPane bottomLeftComp = new JScrollPane(label2);

    // 3. Create a third label to display icon3 and add it to one more scroll pane.
    JLabel label3 = new JLabel(icon3);
    JScrollPane rightComp = new JScrollPane(label3);

    // 4. Add the scroll panes displaying icon1 and icon2 to a split pane.
    JSplitPane splitPanel = new JSplitPane(VERTISPLIT,
                                             continuousLayout,
                                             topLeftComp,
                                             bottomLeftComp);
    splitPanel.setOneTouchExpandable(true); // Provide a collapse/expand widget.
    splitPanel.setDividerSize(2); // Divider size.
    splitPanel.setDividerLocation(0.5); // Initial location of the divider.

    // 5. Add the previous split pane and the scroll pane displaying
    // icon3 to an outer split pane.
    JSplitPane splitPanel2 = new JSplitPane(HORIZSPLIT,
                                             splitPanel, // left comp
                                             rightComp);
    splitPanel2.setOneTouchExpandable(true); // Provide a collapse/expand widget.
    splitPanel2.setDividerLocation(0.4); // divider size
    splitPanel2.setDividerSize(2); // Initial location of the divider.

    // 6. Add the outer split pane to the content pane of the applet.
    getContentPane().add(splitPanel2);
}
```

## 代码详析

在小程序的 Init() 方法内，代码片段 1 创建了一个标签对象来显示图像 1，并把它加到了可滚动窗格上。代码片段 2 和代码片段 3 也分别创建了标签对象来显示图像 2 和图像 3，并把它们加到了相应的滚动窗格上。

代码片段 4 将显示图像 1 和图像 2 的滚动窗格又加入到了一个拆分窗格中。代码片段 5 将这个拆分窗格和显示图像 3 的滚动窗格加入到另一个拆分窗格中。后一个拆分窗格最后被加入了小程序的内容窗格之中。

## 4.7 卡片式窗格

当选中一个卡片时，卡片式窗格便从一组组件中显示出该选中的组件。此卡片的扩展部分可以有一个标题、一个图标，也可以同时拥有两者。对于从多个组件的组合中显示每个组件这一点来说，卡片式窗格可以节省空间。

Swing 卡片式窗格由 JTabbedPane 类表示，它扩展了 JComponent 类。JTabbedPane 实现了 Serializable、Accessible 以及 SwingConstants 接口。

### 4.7.1 JTabbedPane 类的构造函数

下面是 JTabbedPane 类所支持的构造函数代码：

```
public JTabbedPane()
```

这个构造函数创建一个空的卡片式窗格。可以用成员函数 addTab() 向卡片式窗格中添加卡片，该成员函数取位置常数之一。这些位置常数的用法请参看下一个构造函数。

```
public JTabbedPane(int tabPlacement)
```

这个构造函数根据指定的卡片位置创建一个空的卡片式窗格。卡片位置参数可取 TOP、BOTTOM、LEFT 和 RIGHT 之一。

### 4.7.2 Change 事件和监听程序

注意，卡片的索引号从 0 开始到 (n - 1)。每当选择状态发生变化时，卡片式窗格对象就会激活 ChangeEvent 类型的事件。这些事件必须由实现了 ChangeListener 接口的相应监听程序来捕获。要执行的代码包含在 stateChanged() 方法内。

### 4.7.3 JTabbedPane 代码示例

清单 4.6 演示了如何实现卡片式窗格。该卡片式窗格显示出三架飞机不同的演示图片。可以单击某个标签来观察相应的图片。你会注意到，当用鼠标单击标签时，卡片式窗格是如何通过逐次地显示图片来节省空间的。程序的输出结果如图 4-12 所示。

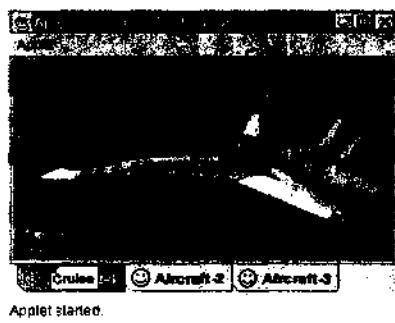


图 4-12 使用卡片式窗格的 Swing 小程序（JTabbedPane）

**清单 4.6 用 JTabbedPane 创建带有三个标签的窗格 (TJTabbedPane.java)**

```
// 演示怎样实现卡片式窗格  
// Demonstrates how to implement a tabbed pane.  
  
/*  
* <Applet code = TJTabbedPane width = 400 height = 300 >  
* </Applet >  
*/  
  
import javax.swing.*;  
import javax.swing.event.*;  
import java.awt.*;  
  
public class TJTabbedPane extends JApplet {  
    JTabbedPane tabbedPane;  
    String[] aircraft = {"Aircraft1.jpg",  
                        "Aircraft2.jpg",  
                        "Aircraft3.jpg"};  
    String[] tips = {"Cruise", "Banking", "Take-off"};  
  
    public void init() {  
        // 1. Create a tabbed pane object.  
        tabbedPane = new JTabbedPane(JTabbedPane.BOTTOM);  
        tabbedPane.addChangeListener(new TabbedPaneListener());  
  
        // 2. Add tabs to the tabbed pane. Each tab displays an aircraft.  
        for (int i = 0; i < aircraft.length; i++) {  
            JLabel label = new JLabel(new ImageIcon(aircraft[i]));  
            tabbedPane.addTab("Aircraft - " + (i + 1), // Tab text  
                            new ImageIcon("smiley.gif"), // Tab icon  
                            label, // component to be displayed  
                            tips[i]); // Some tip at the tab  
        }  
  
        // 3. Add the tabbed pane to the applet.  
        getContentPane().add(tabbedPane);  
    }  
  
    // 4. Tabbed Pane listener class.  
    class TabbedPaneListener implements ChangeListener {  
        int selectedIndex = -1; // -1 = default number less than 0  
        JTabbedPane tp;  
  
        public void stateChanged(ChangeEvent ce) {  
            tp = (JTabbedPane) ce.getSource();  
  
            // 5. Code to disable a tab that has been selected, and  
            // enable the tab that has already been disabled.  
            if (selectedIndex == -1 || // if pressed for first time
```

```
selectedIndex != tp.getSelectedIndex()) {  
    tp.setEnabledAt(tp.getSelectedIndex(), false);  
    if (selectedIndex != -1) // if not the first press  
        tp.setEnabledAt(selectedIndex, true);  
}  
// Store the index of the newly selected tab.  
selectedIndex = tp.getSelectedIndex();  
}  
}  
}
```

### 代码详析

飞机图片文件名和飞机演习方式的名称定义成字符串数组，并作为小程序的数据成员保存。在 Init () 方法内，代码片段 1 创建了一个卡片式窗格。构造函数的参数值指定卡片标签应位于卡片式窗格的下部。该卡片式窗格也由 change 事件监听程序注册。这个 Change 事件监听程序在标签选择状态发生变化时激活。

代码片段 2 创建了显示飞机名称，并且用 addTab () 方法把它们加到了卡片式窗格的不同卡片上。该方法要用到四个参数：要显示在标签上的文本及图标、位于标签中央的组件（标签），以及相应的提示文本串。

代码片段 3 给出了把卡片式窗格加到程序中的代码。代码片段 4 是关于 Change 事件监听程序的代码。监听类实现了 ChangeListener 接口，并包含了 stateChanged () 方法中所执行的功能。

### 说明：

不必处理任何因单击标签而发生的从一个标签切换到另一个标签这样的事件。

但是，如果希望在单击按钮时执行某项功能，可以使用 ChangeListener 对象。stateChanged () 方法中所实现的代码使得已经被选中的标签成为不可用的，同时也使得先前被改成不可用的标签现在成为了可用的。代码片段 5 显示了这个功能。

# 第 5 章 布局管理器

布局 (Layout) 是指如何在小程序或框架内排列用户接口组件。要按照特定的布局要求把组件放置在某个位置, Swing 工具集主要是依靠原先的抽象窗口工具包 (AWT) 所提供的布局类来完成的。AWT 支持布局管理器的核心技术, 能够根据给定的布局要求来控制每一个组件位置。

AWT 布局管理器有五种: 流布局管理器、网格布局管理器、边界布局管理器、卡片布局管理器以及无序网格布局管理器。而 Swing 布局管理器只有四种。如果不熟悉 AWT 的无序网格布局, 那么框式布局就是最为重要的了, 它可以节省时间。其他的 Swing 布局管理器还有覆盖布局管理器, 滚动窗格布局管理器以及视口布局管理器。这些布局都是由 Swing 组件使用, 用户不必直接处理它们。

本章首先说明怎样创建布局管理器, 并把它赋给一个小程序或者一个框架, 随后的内容讨论每一种布局管理器。对 AWT 和 Swing 中每一个布局管理器的详细讨论都提供相应的例子。本章包含的材料来自 AWT, 因为 Swing 组件都是按照 AWT 中所支持的布局进行安排的。

## 5.1 布局管理器

抽象窗口工具包 (AWT) 支持五个处理不同布局外观的类: `FlowLayout`、`GridLayout`、`BorderLayout`、`CardLayout` 和 `GridBagLayout`。所有这些类都实现了接口函数 `java.awt.LayoutManager`, 该接口函数被当作布局管理器加以引用。应用程序可以利用上述任何一种布局管理器, 来控制组件的排列位置及大小, 从而保存用户界面的外观。

流布局管理器和网格布局管理器支持简单布局。用流布局管理器能够将组件按照其自身的大小显示在一行上, 并且在容器的边界处折到下一行。而网格布局管理器则适合于按行和列来显示大小相同的组件。

边界布局管理器和卡片布局管理器用于复杂的编程工作。边界布局管理器依照容器的北、南、东、西、中这些地理方位来放置组件。卡片布局管理器则将多个组件相互重叠放置, 使得任何时候只有一个卡片是可见的。

无序网格布局是 AWT 布局管理器中最复杂、最通用的布局组件。它学起来有点复杂, 需要一些几何学的知识, 以便掌握其内在的思想。可以利用无序网格布局获得任何布局模式。

除了这些布局管理器之外, Swing 工具集还提供了框式布局 (Box Layout), 重叠式布局 (Overlay Layout), 滚动窗格布局 (Scroll Pane Layout) 以及视口布局 (View Port Layout)。框式布局很重要, 因为可以用它来创建与通过无序网格布局所生成的相同的布局。可以用框式布局来代替无序网格布局。Swing 的其他布局在这里就不是很重要了, 这是由于它们仅仅被 Swing 组件本身所使用; 而在构造应用程序时, 实际上不用它们。

### 5.1.1 为容器指定布局管理器

当创建了一个特定的布局管理器对象之后，需要将它赋予某容器，以便按照布局模式将组件加到容器上。Container类（在java.awt包内）提供了以下两种方法来将布局管理器赋予某个容器，或者检索已经被赋予某个容器的布局对象：

```
public void setLayout(LayoutManager layoutObject)  
public void getLayout()
```

#### 说明：

当把布局管理器赋予 Swing 小程序或者框架时，必须将布局对象赋予小程序或框架的内容窗格，而不是把它直接赋予小程序或框架对象。

### 5.1.2 安排组件

一旦将布局管理器赋予了某个容器，就可以利用下面的来自 Container类的方法将其上的组件放置在相应的位置：

```
public void add(Component comp)  
public void add(Component comp, Object Constraints)  
public void add(Component comp, int index)  
public void add(Component comp, Object Constraints, int index)
```

这些方法中的参数 comp 是要被放置在容器内的组件。参数 Constraints 是一个对象，当使用象无序网格布局这样的布局对象时，Constraints 对象就会对组件施加一定的限制。参数 index 指定组件加入到容器内的顺序。如果没有指定 index，则组件被加到组件堆栈表的底部。

也可以使用下面的方法将组件从容器中去掉，这些方法用到的参数与先前所做的说明相同。

```
public void remove(int index)  
public void remove (Component comp)  
public void removeAll()
```

这几个方法中，removeAll（）方法会把调用该方法的容器中的所有组件都删除。

#### 说明：

正象布局管理器要被赋予 Swing 小程序或者框架一样，组件也要加到内容窗格上，或者从内容窗格上去掉。

## 5.2 流布局管理器

流布局管理器（Flow Layout Manager）支持的布局结构为，组件是按照其自身的或者某个预选的大小显示在一行上。如果组件不能在一行内排列完，那么流布局管理器就会将组件

折到下一行上。这个过程会连续地从上到下进行，直到所有的组件都被放置在了容器内。也可以指定组件是按照左对齐、右对齐或者居中排列。缺省情况下，组件在每一行上是居中排列的。

### 5.2.1 FlowLayout 构造函数

可以用下面任何一个构造函数创建流布局：

```
public FlowLayout()
public FlowLayout(int alignment)
public FlowLayout(int alignment, int horizGap, int vertGap)
```

其中，参数 alignment 取以下三种值之一：

```
FlowLayout.LEFT = 0
FlowLayout.CENTER = 1
FlowLayout.RIGHT = 2
```

参数 horizGap 和 vertGap 是以像素为单位指定的水平间隔和垂直间隔。水平间隔是一行中两个相邻组件之间的间隔，垂直间隔是两个相邻行的组件之间的间隔。二者的缺省值为 5 个像素。

### 5.2.2 FlowLayout 代码示例

清单 5.1 包含一个演示流布局怎样管理组件的 Swing 小程序。该程序创建了几个按钮，并把它们加到了程序上（参看图 5-1）。

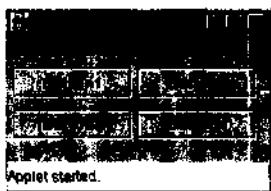


图 5-1 演示 FlowLayout 的 Swing 小程序

清单 5.1 FlowLayout (TFlowLayout.java)

```
/*
 * <Applet code=TFlowLayout width=300 height=100>
 * </Applet>
 */
import javax.swing.*;
import java.awt.*;

public class TFlowLayout extends JApplet {
    Container container = null;
    public void init() {
```

```
// 1. Get the handle on applet's content pane.  
container = this.getContentPane();  
  
// 2. Create the flow layout object and set it for the applet.  
FlowLayout fl = new FlowLayout(FlowLayout.LEFT, 5, 10);  
// horizontal gap = 5 pixels, vertical gap = 10 pixels  
  
container.setLayout(fl);  
  
// 3. Create and add some buttons to the applet..  
for (int i = 0; i < 4; i++)  
{  
    JButton button = new JButton("Button" + (i + 1));  
    button.setPreferredSize(new Dimension(100, 25)); // width = 100, height = 25  
    container.add(button);  
}  
}  
}
```

### 代码详析

代码片段 1 取得一个小程序内容窗格的句柄。代码片段 2 中，利用所创建的流布局管理器对象将小程序内容窗格设置为流布局。代码片段 3 是一个控制循环体，这里按钮被加到了程序中。注意，预先选定的按钮尺寸得到调整以便容纳稍宽一些的标签，因为按钮不会填充任何额外的空间。

注意，流布局将 Button3 和 Button4 折入了第二行中，因为它不能够把这些按钮按照预选的大小全部放在第一行内。但是，如果象图 5-2 中所示那样改变小程序窗口的大小，则可将第二行中的按钮放在第一行内。该程序也显示出组件是左对齐的，水平间隔和垂直间隔都是 5 个象素。

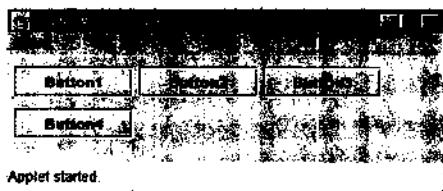


图 5-2 改变了大小的流布局小程序

### 说明：

流布局管理器总是尽可能地将组件放在一行内。每当程序窗口的大小改变时，组件布局的外观就会随之改变。

## 5.3 网格布局管理器

网格布局管理器 (Grid Layout Manager) 将组件放在附属于容器的网格的空格中。网格中所有空格的大小都是一样的，并且网格中空格的个数可由网格的行数和列数决定。例如，一个三行两列的网格能够产生六个大小相等的空格，它们可以用来放置六个组件。每一个用户接口组件填满一个空格。

即使改变容器的大小，布局仍然会保持原样，而组件的大小也会随之改变以填满网格空间。当重新设置布局时，组件之间可以增加额外的间隔。

### 5.3.1 网格布局构造函数

`GridLayout` 类型的对象可以用下面的构造函数之一来创建：

```
Public GridLayout(int rows, int columns)
Public GridLayout(int rows, int columns,
                  int horizGap, int vertGap)
```

参数 `rows` 和 `columns` 指定所创建网格的行数和列数。这两个参数中必须至少有一个非零。`rows` 或 `columns` 为零，则表示在一行或一列中可以排列任意多个组件。参数 `horizGap` 和 `vertGap` 分别指定了网格之间的空格大小（像素的个数）。

### 5.3.2 GridLayout 代码示例

清单 5.2 演示了网格布局管理器的布局效果。该程序创建一个小程序，它使用网格布局管理器构成了六个单元（3 行 2 列）。每个网格单元中放了一个 Swing 按钮。注意，每个按钮都填满了网格单元的整个空间（参看图 5-3）。

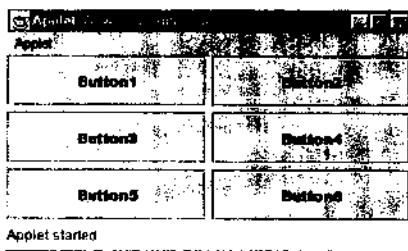


图 5-3 演示 GridLayout 的 Swing 小程序

清单 5.2 GridLayout (TGridLayout.java)

```
/*
 * <Applet code = TGridLayout width = 350 height = 150 >
 * </Applet >
 */
import javax.swing.*;
```

```
import java.awt.*;  
  
public class TGridLayout extends JApplet {  
    Container container = null;  
  
    public void init() {  
        // 1. Get a handle on the applet's content pane.  
        container = this.getContentPane();  
  
        // 2. Assign the grid layout to the content pane.  
        GridLayout grid = new GridLayout(3,2,5,5);  
        // rows = 3, cols = 2, horizontal gap = 5, vertical gap = 5  
        container.setLayout(grid);  
  
        // 3. Create and add components to the content pane.  
        for (int i = 0; i < 6; i++) {  
            container.add(new JButton("Button" + (i + 1)));  
        }  
    }  
}
```

### 代码详析

代码片段 1 取得一个小程序内容窗格的句柄。代码片段 2 创建了一个 `TGridLayout` 类型的对象。构造函数需要四个参数：行数、列数、水平间隔和垂直间隔。

代码片段 3 将 Swing 按钮加到所产生的六个网格单元中。读者可以试着改变程序窗口的大小，注意观察此时产生的额外空间只是简单地被组件填充了。

## 5.4 边界布局管理器

边界布局管理器（Border Layout Manager）依照容器的北、南、东、西、中这些地理方位来放置组件。组件会以其本身的大小或选定的大小填满相应的位置，并且它们之间不留间隔。缺省情况下，Swing 小程序的内容窗格对象和框架都带有边界。

即使改变容器的大小，布局仍然会保持原样。南北方向的组件会沿水平方向伸缩，而在垂直方向上已经选定的高度保持不变。类似地，东西方向的组件会沿垂直方向伸缩，而在水平方向上已经选定的宽度却保持不变。处于中间位置的组件则将填满水平和垂直方向上的全部空间。

### 5.4.1 边界布局构造函数

`BorderLayout` 类的实例可以用下面的构造函数之一来创建：

```
Public BorderLayout()  
Public BorderLayout(int horizGap, int vertGap)
```

参数 `horizGap` 和 `vertGap` 分别指定了组件之间的空格大小（象素的个数）。

对于任何使用边界布局管理器的容器，都可使用 `java.awt.Container` 类中带两个参数的方法 `add(Component comp, Object location)` 来添加组件。

### 5.4.2 BorderLayout 代码示例

清单 5.3 用一个 Swing 小程序演示了边界布局管理器的用法。这个 Swing 小程序已经缺省地采用边界布局效果，因此，不必再明确对它进行设置。该程序仅仅简单地往其中添加了几个 Swing 按钮（参看图 5-4）。

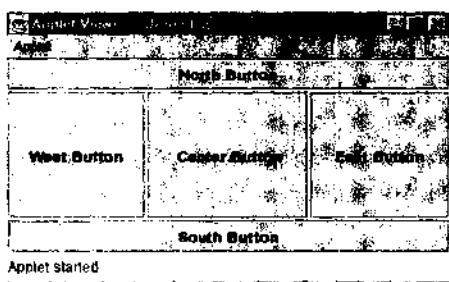


图 5-4 使用边界布局管理器的 Swing 小程序

清单 5.3 边界布局管理器 (`TBorderLayout.java`)

```

/*
<Applet code = TBorderLayout width = 400 height = 250>
</Applet>
*/
import java.awt.*;
import javax.swing.*;

public class TBorderLayout extends JApplet {
    Container container = null;

    public void init() {
        //1. Get a handle on the applet's content pane.
        container = this.getContentPane();
        // container.setLayout(new BorderLayout());
        /* Note: Don't have to explicitly set border layout,
           because the content pane uses border layout
           by default */

        // 2. Give some horizontal and vertical gaps between buttons.
        ((BorderLayout) container.getLayout()).setHgap(2);
        ((BorderLayout) container.getLayout()).setVgap(2);

        // 3. Array that contains border layout constants.
        String[] borderConsts = { BorderLayout.NORTH,
            BorderLayout.SOUTH,
            BorderLayout.EAST,
            BorderLayout.WEST,
            BorderLayout.CENTER };
        ...
    }
}

```

```
BorderLayout.EAST,  
BorderLayout.WEST,  
BorderLayout.CENTER );  
  
// 4.Array that contains button names.  
String[] buttonNames = { "North Button", "South Button",  
                        "East Button", "West Button",  
                        "Center Button" };  
  
// 5.Create and add buttons to the container.  
for (int i = 0; i < borderConsts.length; i++) {  
    JButton button = new JButton(buttonNames[i]);  
  
    // You may wish to reset the preferred size of the east  
    // and west buttons to display wide labels.  
    if (buttonNames[i] == "East Button" ||  
        buttonNames[i] == "West Button") {  
        button.setPreferredSize(new Dimension(115, 25));  
        // width = 115, height = 25  
    }  
    container.add(button, borderConsts[i]);  
}  
}
```

## 代码详析

代码片段1取得小程序内容窗格的句柄。因为程序的缺省布局格式为带边界的布局，所以，不必再对它重新进行设置。代码片段2在各组件之间留出一些间隙。GetLayout()方法返回一个必须强制转化成边界布局的对象。另外两个方法setHgap()和setVgap()则负责创建按钮之间的水平间隔和垂直间隔。

代码片段3和片段4创建了两个分别包含边界布局常数和按钮名称的数组。代码片段5利用这些信息创建Swing按钮，并将它们加到程序的内容窗格上。请注意，可能需要改变东西方向按钮的大小，以便它们能够容纳按钮上的按钮名。

## 5.5 卡片布局管理器

CardLayout类支持把各种GUI组件放置分别在不同的卡片上的布局管理，而这些卡片可以相互重叠并共享同一个显示区域。一个时间只能看到一个卡片——位于最上面的那张卡片。可以通过实现CardLayout类中的方法把其他的卡片调到前面显示。

### 5.5.1 CardLayout构造函数

CardLayout类的实例可以用下面的构造函数之一来创建：

```
public CardLayout()
```

```
public CardLayout(int horizGap, int vertGap)
```

第二个构造函数按照参数 `horizGap` 和 `vertGap` 指定的水平间隔和垂直间隔创建一个布局。水平间隔放在容器的左右两侧，而垂直间隔放在容器的上下边。

### 5.5.2 翻动卡片

卡片布局把几个窗格关联起来了，而组件通常又是组合在窗格中的。下面的方法支持翻到容器内的第一张卡片和最后一张卡片：

```
public void first(Container target)
public void last(Container target)
```

下面的方法翻到当前所显示卡片的后一张或前一张：

```
public void next(Container target)
public void previous(Container target)
```

也可以指定卡片的名字来显示相应卡片。如果不存在指定名字的卡片，则不会产生任何效果。下面这个来自 `CardLayout` 类的方法就能够根据名字显示卡片：

```
public void show(Container target, String name)
```

以上各种方法的参数 `target` 是容纳布局的容器。

### 5.5.3 CardLayout 代码示例

清单 5.4 创建了一个 Swing 小程序，其中显示了两张卡片。每张卡片都包含若干个链接到某个 Web 站点的按钮。第一张卡片上的按钮分别链接到 JavaSoft、Sun、IBM 和 Netscape。第二张卡片上的按钮分别链接到几个航空组织。可以把这几个按钮的具体实现作为练习去完成；还可以用 `Next` 和 `Previous` 按钮翻动卡片（参看图 5-5）。

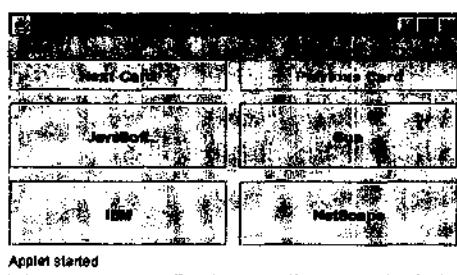


图 5-5 演示卡片布局管理器的小程序

#### 清单 5.4 CardLayout (`TCardLayout.java`)

```
/*
 * <Applet code = TCardLayout width = 400 height = 200 >
 * </Applet >
 */
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class TCardLayout extends JApplet {
    // 1. Declare some panel and button references.
    JPanel controlPanel;
    JPanel cardsPanel1, swCompPanel, aeroCompPanel;
    JButton nextButton, previousButton;
    JButton javaSoftButton, sunButton,
            ibmButton, netscapeButton;
    JButton nasaButton, boeingButton,
            lockheedButton, northropButton;
    URL currentSite = null;

    // 2. Define the string arrays for the sites and site urls.
    String[] swCompSites = {"JavaSoft", "Sun",
                           "IBM", "NetScape"};
    String[] aeroCompSites = {"NASA", "Boeing",
                           "Lockheed", "Northrop"};
    String[] siteURLs = { "http://www.javaSoft.com/",
                          "http://www.sun.com/",
                          "http://www.ibm.com/",
                          "http://www.netscape.com/" };

    Container container = null;

    public void init() {
        // 3. Get a handle on the applet's content pane.
        container = this.getContentPane();

        // 4. Create a control panel object.
        controlPanel = new JPanel();
        controlPanel.setLayout(new GridLayout(1, 2, 10, 10));

        // 5. Create and add the next and previous control buttons.
        ButtonListener listener = new ButtonListener();
        nextButton = new JButton("Next Card");
        nextButton.setActionCommand("Next Button");
        nextButton.addActionListener(listener);
        controlPanel.add(nextButton);

        Listing 5.4 continued
        previousButton = new JButton("Previous Card");
        previousButton.setEnabled(false);
        previousButton.setActionCommand("Previous Button");
        previousButton.addActionListener(listener);
```

```
controlPanel.add(previousButton);

// 6.Create a panel to contain the cards.
/* Each card will display a set of buttons to
visit a specific Web site. */
cardsPanel = new JPanel();
cardsPanel.setLayout(new CardLayout());
swCompPanel = new JPanel();
aeroCompPanel = new JPanel();
swCompPanel.setLayout(new GridLayout(2,2,10,10));
aeroCompPanel.setLayout(new GridLayout(2,2,10,10));

// 7.Add buttons to the these cards.
for (int i = 0; i < swCompSites.length; i++) {
    JButton b = new JButton(swCompSites[i]);
    b.addActionListener(listener);
    b.setActionCommand(swCompSites[i]);
    swCompPanel.add(b);
}

for (int i = 0; i < aeroCompSites.length; i++) {
    JButton b = new JButton(aeroCompSites[i]);
    b.addActionListener(listener);
    b.setActionCommand(aeroCompSites[i]);
    aeroCompPanel.add(b);
}

// 8.Add the company - list panels to cards panel.
cardsPanel.add("Next Card", swCompPanel);
cardsPanel.add("Previous Card", aeroCompPanel);

// 9.Finally, add the control and the cards panels to
// the applet's content pane.
((BorderLayout) container.getLayout()).setVgap(10);
container.add("North", controlPanel);
container.add("Center", cardsPanel);
}

// 10.Functionality to be executed when the next - button
// is activated.
public void showNextCard() {
    nextButton.setEnabled(false);
    ((CardLayout) cardsPanel.getLayout()).next(cardsPanel);
    previousButton.setEnabled(true);
}

// 11.Functionality to be executed when the previous - button
// is activated.
```

```
public void showPreviousCard() {
    previousButton.setEnabled(false);
    ((CardLayout) cardsPanel.getLayout()).previous(cardsPanel);
    nextButton.setEnabled(true);
}

// 12. Functionality to display the specific Web page.
public void toDestination(String aSite) {
    System.out.println("Connecting to the web site...");
    try {
        currentSite = new URL(aSite);
    }
    catch (MalformedURLException e) {
        System.out.println("Unknown URL Address: " + aSite);
    }
    getAppletContext().showDocument(currentSite);
}

// 13. The action listener class.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        JButton tempButton = (JButton) ae.getSource();

        // 14. If the next or previous buttons are pressed...
        if (tempButton.getActionCommand() == "Next Button") {
            showNextCard();
        }
        else if (tempButton.getActionCommand() == "Previous Button") {
            showPreviousCard();
        }

        // 15. If the Web site buttons are pressed...
        for (int i = 0; i < swCompSites.length; i++) {
            if (tempButton.getActionCommand() == swCompSites[i])
                toDestination(siteURLs[i]);
        }
    }
}
```

### 代码详析

代码片段 1 声明几个对象的引用，它们作为小程序的数据成员。控制窗格中包含了 Next 和 Previous 按钮，用来翻动卡片。而卡片窗格中又包含了很多窗格，上面的按钮将把你带到这些站点。代码片段 2 创建了两个数组，用来存放计算机名称和几个航空公司的名称。引用容器提供了程序内容窗格的句柄。

在 init() 方法里, 代码片段 3 对内容窗格的引用进行初始化。代码片段 4 创建了网格布局控制窗格。代码片段 5 创建了控制按钮, 用于变换选择卡片。这些按钮已经加到了控制窗格之上。这里, 你不必去管操作按钮对象的方法。但你可能注意到, 以 true 参数调用 setEnabled() 方法时, 可以使按钮成为可用的如果以 false 参数调用 setEnabled() 方法, 按钮就变成不可用的了。通过取得字符串参数, 用 SetActionCommand() 方法能够识别指定的按钮, 也可以象代码片段 13 那样进行其他操作。

代码片段 6 创建了一个卡片窗格, 并将其设置成卡片布局形式, 而这个窗格本身又可以包含其他几个窗格。其中的代码将公司窗格加到卡片窗格上。这里的公司窗格上又包括有几个链接到特定站点的按钮。代码片段 7 给出了对应于各个站点的按钮的具体实现。两个 for 循环分别针对计算机和航空公司站点对按钮进行初始化。

代码片段 8 把公司窗格加到了卡片窗格上。代码片段 9 将控制窗格和卡片窗格加到程序内容窗格的北侧和中间位置。

代码片段 10 和代码片段 11 是显示前一张或后一张卡片(当它们被激活时)方法的具体实现。这些方法也可以依据文本使按钮成为可用或者不可用的。代码片段 12 给出了显示 Web 站点(当激活了卡片上的按钮时)的方法。这个方法以 Web 站点的 URL 字符串作为参数创建其 URL 对象。该对象再使用 showDocument() 方法在程序内容窗格上显示出相关的 Web 页面。

代码片段 13 是按钮监听程序段, 它根据单击的按钮激活所需的方法。如果单击了 Next 按钮, 则会激活 showNextCard() 方法。为了识别单击了哪个按钮, 要用到 getActionCommand() 方法来取得已由 setActionCommand() 方法设置的动作命令字符串。然后, 这个字符串要与引用字符串作比较。代码片段 14 为其具体作法。代码片段 15 根据单击的按钮装载某个特定的 Web 页面。

## 5.6 无序网格布局管理器

无序网格布局(Grid Bag Layout)是最灵活, 同时也是最复杂的布局管理器, 其布局机制多少有些复杂并且令人感到棘手。无序网格布局可以使组件扩充到多行多列, 而无须所有组件都具有同样的大小。这可以通过设置一个与容器关联的单元的矩形网格并把每个组件放在一个或多个单元内这个办法来实现。这些单元形成了组件的显示区域。

### 5.6.1 GridBagLayout 构造函数

无序网格布局使用两个类来排列组件: java.awt.GridBagLayout 和 java.awt.GridBagConstraints。而构造函数:

```
public GridBagLayout()
```

创建一个新的无序网格布局管理器, 再用 setLayout(GridBagLayout gridbag) 方法将其设置成一个容器。

### 5.6.2 无序网格约束

无序网格布局管理器放置组件的模式依赖于下面这个方法施加在组件上的约束条件:

```
public void GridBagConstraints(Component comp,
    GridBagConstraints constraint);
```

这里的约束条件由 `GridBagConstraints` 类中的一组公共字段表示。这些字段可以被赋以特定的值，或者被赋以该类中已有的一组值。下一节中将描述 `GridBagConstraints` 类中的这些公共字段。

### `gridx` 和 `gridy`

放置一个组件时需要两个坐标 ( $x, y$ )，例如，确定位于网格单元中某个按钮 `B1` 的位置时就需要有坐标。然后，就可以把该按钮的某一角，最好是左上角，放在这个特定的坐标位置 ( $x, y$ ) 了（请参看图 5-6）。

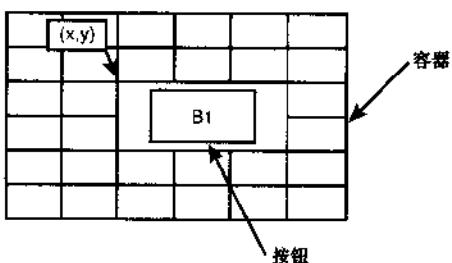


图 5-6 演示卡片布局管理器的小程序

字段 `gridx` 和 `gridy` 分别表示  $x$  坐标和  $y$  坐标。`gridx` 表示组件显示区域左侧单元的个数，其中，最左边单元的 `gridx = 0`。`gridy` 表示组件显示区域上方单元的个数，其中，最上面单元的 `gridy = 0`。上图 5-6 中的  $(x, y)$  是  $(2, 2)$ 。

`gridx` 和 `gridy` 也可以取 `GridBagConstraints` 类中所提供的常数 `RELATIVE`。该常数规定，将新加入的组件按照特定的坐标方向放在上一个加入的组件的下一个位置。缺省地，`gridx` 和 `gridy` 都取数值 `RELATIVE`。

### `gridwidth` 和 `gridheight`

当按照  $(x, y)$  坐标放置了组件之后，就可以确定组件的大小了。组件的大小可以用 `gridwidth` 和 `gridheight` 表示，它们分别表示沿行和列方向所扩充的单元数。在前面的图 5-6 中，`gridwidth` 和 `gridheight` 的值分别是 3 和 2。

也可以用 `REMAINDER` 作为 `gridwidth` 和 `gridheight` 的取值，分别表示该组件扩充到最后一行或最后一列。`gridwidth` 和 `gridheight` 可取的另一常数是 `RELATIVE`，它表示，按照特定的行列值将该组件放在上一个加入的组件的后一个位置。缺省地，`gridwidth` 和 `gridheight` 的值均为 1。

### `weightx` 和 `weighty`

当改变容器大小时，字段 `weightx` 和 `weighty` 用于分别指定怎样分配多余的水平空间和垂直空间。为了取得所需的用户界面效果，这两个值起着关键的作用。它们可以设置为界于 0.0 和 1.0 之间任何双精度类型的值。如果组件的某个权数取了较大的值，那么该组件就会

占据该方向上额外的空间。权数为零的组件不占据额外的空间。缺省地, weightx 和 weighty 都设置为 0。

#### fill

当组件的尺寸比其显示区域小时, 字段 fill 就可以发挥作用了, 它指定一个组件在其显示区域内是如何分布的。可以将 NONE、HORIZONTAL、VERTICAL 以及 BOTH 中的任何一个值赋给字段 fill。当把 fill 设为 NONE 时, 它表示组件将不会填充显示区域。第二个常数 HORIZONTAL 使得组件的宽度能够达到仅沿水平方向填满显示区域。常数 VERTICAL 使得组件的高度能够达到仅沿垂直方向填满显示区域。最后一个常数 BOTH 使得组件沿水平方向和垂直方向完全填满显示区域。缺省地, fill 的值设置为 0。

#### anchor

字段 anchor 指定当组件小于显示区域时停靠的位置。停靠位置按照地理方位确定。该字段可能的取值有 NORTH、NORTHEAST、EAST、SOUTHEAST、SOUTH、SOUTHWEST、WEST、NORTHWEST 以及 CENTER。缺省地, 组件位于显示区域的中心。

#### insets

insets 是 AWT 中提供的类 Insets 的类型字段。这个字段用于指定组件的外部填充区域的大小 (external padding) ——界于组件与其显示区域的边界之间的空间。通过利用 Insets 类的构造函数 Insets (int top, int left, int bottom, int right) 创建 Insets 对象, 可以设置这个字段的值。参数 top、left、bottom、right 分别是四个方向上相对于显示区域边界留出的空间。缺省地, insets 字段已经设置成对象 Insets (0, 0, 0, 0), 也就是没有外部填充区域。

#### ipadx 和 ipady

字段 ipadx 和 ipady 指定组件的内部填充区域 (internal padding) ——加在组件现有宽度和高度上的额外宽度和高度值。扩充的部分对组件两边的宽度和高度同时起作用。因此, 额外的宽度和高度分别是 ipadx 和 ipady 的两倍。缺省情况下, 组件没有内部填充区域。

### 5.6.3 GridLayout 代码示例

清单 5.5 演示了怎样用无序网格布局在容器内放置组件 (参看图 5-7)。程序使用了数值坐标, 而不是常数 RELATIVE 和 REMAINDER。

清单 5.5 GridLayout (TGridLayout.java)

```
/*
 * < Applet code = TGridLayout width = 200 height = 200 >
 * < /Applet >
 */
import javax.swing.*;
import java.awt.*;
```

```
public class TGridLayout extends JApplet {
    Container container = null;
    public void init() {
        // 1. Get a handle on the applet's content pane.
        container = this.getContentPane();
        // 2. Set the container to the grid bag layout and
        // define a constraint object.
        GridBagLayout gridbag = new GridBagLayout();
        container.setLayout(gridbag);
        GridBagConstraints c = new GridBagConstraints();
        // 3. Common settings for constraint object instant variables.
        c.fill = GridBagConstraints.BOTH;
        // 4. Settings for button B1.
        c.insets = new Insets(5,5,5,5);
        c.gridx = 0; c.gridy = 0;
        c.gridwidth = 2; c.gridheight = 2;
        c.weightx = 1.0; c.weighty = 1.0;
        makeButton("B1",gridbag,c);
        // 5. Settings for button B2.
        c.insets = new Insets(0,0,0,0);
        c.gridx = 2; c.gridy = 0;
        c.gridwidth = 1; c.gridheight = 3;
        makeButton("B2",gridbag,c);
        // 6. Settings for button B3.
        c.gridx = 0; c.gridy = 2;
        c.gridwidth = 1; c.gridheight = 1;
        c.weightx = 1.0; c.weighty = 0.5;
        makeButton("B3",gridbag,c);
        // 7. Settings for button B4.
        c.gridx = 1; c.gridy = 2;
        makeButton("B4",gridbag,c);
    }
    // 8. Define the function to create and add a button
    // according to the constraints.
    public void makeButton(String name,
                          GridBagLayout gridbag,
                          GridBagConstraints c) {
        JButton button = new JButton(name);
        gridbag.setConstraints(button,c);
        container.add(button);
    }
}
```

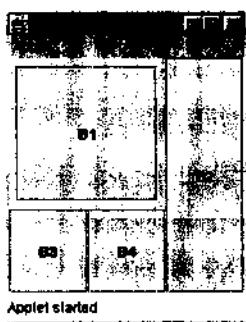


图 5-7 无序网格布局管理器的 Swing 小程序

### 代码详析

清单 5.5 是一个 Swing 小程序，它按照无序网格的布局结构在窗格上放置了四个按钮。本例的目的是，按照图 5.7 所示的布局配置放置按钮。希望 B1 为正方形按钮，B2 位于容器的右边，并且从顶部扩展到底部。B3 和 B4 也是正方形按钮，它们从容器左下角相邻排列。

`Init()` 方法内的代码片段 1 得到内容窗格的句柄。代码片段 2 创建了无序网格布局对象并把它赋给小程序。该程序段为小程序提供了如图 5-6（在说明 `gridx` 和 `gridx` 字段时）所示的网格模式。

随后，定义了一个无序网格 `constraint` 对象，以便能够在一定的约束条件下放置组件。通常，只要为组件创建一个简单的 `constraint` 对象并对约束字段做相应的初始化就够了。也可以为若干个组件或一组组件使用几个 `constraint` 对象。选择哪一种方法取决于组件的个数以及布局的复杂程度。本例中只用到了一个 `constraint` 对象（叫做 `c`），它的字段都是用来控制不同按钮的。

代码片段 3 对约束字段 `fill` 进行初始化，这是对所有组件的共同操作。而其他字段，如 `gridx`、`gridy`、`gridwidth`、`gridheight`、`weightx`、`weighty` 等，则要针对不同的按钮赋以相应的值。

如果检查一下图 5-7 中布局所需的样式，那么各个按钮显示区域的坐标和大小就可以按照以下方式进行编码：

```
'B1' button:
    c.gridx = 0; c.gridy = 0;
    c.gridwidth = 2; c.gridheight = 2;

'B2' button:
    c.gridx = 2; c.gridy = 0;
    c.gridwidth = 1; c.gridheight = 3;

'B3' button:
    c.gridx = 0; c.gridy = 2;
    c.gridwidth = 1; c.gridheight = 1;

'B4' button:
```

```
c.gridx = 1; c.gridy = 2;  
c.gridwidth = 1; c.gridheight = 1;
```

这些约束设置在 `init()` 方法的其余部分给出。最后，最关键的一步是，设置 `weightx` 和 `weighty` 的值，并把它们调整为合适值。这一步就创建了所需的布局外观。

约束字段 `weightx` 和 `weighty` 的值定义在 0 到 1.0 之间。如果所有组件的权值设置为 0 的话，组件就会象图 5-8 所示那样相互紧挨在一起，并位于容器中央。最简单的方法是，先把所有的权值都设置为 1.0，然后再对它们进行微调。图 5-9 是当所有的权值都设置为 1.0 时的布局结果。

对权值的微调是在考虑到用户界面外观效果的基础上，进行反复试验的过程。比较一下图 5-7 和图 5-9，需要将 B1 按钮沿 y 方向（即高度）改变其尺寸，以使其成为一个正方形。注意，按钮的外观还依赖于容器的形状（在这里，容器的宽度和高度是相等的）。如果仔细地观察一下程序代码，可以发现按钮 B1 的 `weighty` 一直保持为常数，而按钮 B3 和 B4 的 `weighty` 都减小到了 0.5。如果想使按钮 B1 的高度是按钮 B3 和 B4 高度的两倍，那么这些值就是很容易实现的。最终的权值将产生图 5-7 所示用户界面的期望效果。

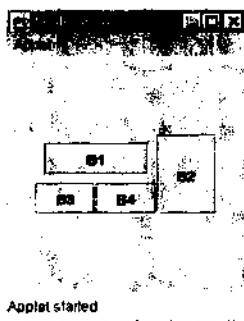


图 5-8 无序网格布局管理器的 Swing 小程序（所有权值为 0）



图 5-9 无序网格布局管理器的 Swing 小程序（所有权值为 1）

代码片段 8 处理具有约束条件的按钮的排列问题。在这个约束关系中，方法 `setConstraints (Component comp, GridBagConstraints c)` 用于把事先约定的约束条件施加到按钮之上。方法 `makeButton ()` 中包含了这条语句。

makeButton() 利用 add() 方法把每个按钮加到了程序中。

图 5-10 给出了当约束字段 fill 设置为 NONE 时的布局结果。由于已经调整了权值，按钮出现在显示区域中的相对位置上，并且保持适当的大小，但它们没有填满整个区域。这说明，对各个组件单独赋以相应的权值，能够控制额外水平空间和垂直空间的分布。

## 5.7 框式布局

利用前面讨论过的无序网格布局，几乎可以产生任意复杂的布局模型。除了这一个 AWT 布局管理器之外，Swing 还引入了另外一个叫做框式布局（Box layout）的结构。框式布局能够生成复杂的布局结构，但却没有象使用无序网格布局那样麻烦。编程者不必处理任何权值，也不用采用反复试验的办法。使用框式布局，可以象使用流布局一样简单地将组件安排在一个容器内。包含框式布局的容器可以嵌套使用，最终达到类似于无序网格布局那样的效果。

框式布局模式有两种类型：水平模式和垂直模式。水平模式把组件在容器内从左到右排列。垂直模式把组件在容器内从上到下进行排列。框式布局所取的水平方向和垂直方向分别由 x 轴和 y 轴表示。因此，对于一个给定的框式布局来说，x 轴和 y 轴将成为主轴方向。

任何时候，当用框式布局把一个组件加到容器内时，该组件的大小将会保持不变。这就是说，框式布局与网格布局不同，它允许组件沿主轴方向占据不同的空间大小。

在非主轴方向上，如果几个组件大小不同，那么框式布局就会以尺寸最大的那个组件的大小来布置所有的组件。例如，在水平框式布局中，如果所有的组件的高度不同（非主轴方向上），则布局管理器就会力图使所有组件都与最高的组件具有同样的高度。

如果不能沿非主轴方向调整尺寸，则组件就会以该轴为中心线进行排列。也就是说，在非主轴方向上，任何一个组件的中心坐标与其他组件的中心坐标是一致的。

如果改变了容器的大小，框式布局不会把任何组件折到下一行去。例如，在一个包括一些按钮的水平框式布局中，改变当前容器的大小时，位于一行上的按钮将不会被折到下一行上。



图 5-10 无序网格布局管理器的 Swing 小程序（具有适当的权值，但 fill=NONE）

### 5.7.1 BoxLayout类

BoxLayout类实现了产生于抽象窗口工具库(AWT)的LayoutManager2接口。该类是Swing库的成员，存放在javax.swing包中。除java.lang.Object以外，这个类不作为其他任何类的子类。

通常，不用BoxLayout类来处理框式布局，而是用下一节将要介绍的Box类。然而，BoxLayout类有几个重要特性需加以注意。

要创建框式布局来从上到下或者从左到右排列组件，可以使用下面的构造函数：

```
public BoxLayout(Container target, int axis)
```

这里，target是布局的容器，axis指定框式布局的方向为水平的或者垂直的，并取下列静态常数之…：

```
static int X_AXIS  
static int Y_AXIS
```

### 5.7.2 Box容器

Box类表示一个轻量级容器，它只能使用框式布局。试图把任何其他类型的布局管理器赋给它时，将会产生一个AWTError。Box容器允许组件根据指定的方向水平排列组件或者垂直排列。Box对象由取axis参数的构造函数或者用很方便的静态方法创建。下面是Box类的构造函数：

```
Public Box(int axis)
```

参数axis取值为BoxLayout.X\_AXIS或BoxLayout.Y\_AXIS。下面是两个很方便的静态方法，它们无需任何参数就可分别创建水平的或垂直的方框容器：

```
public static Box CreateHorizontalBox()  
public static Box CreateVerticalBox()
```

Box类还提供了有用的方法，这些方法增加了一些特性，如产生组件内部的间隔、当容器改变大小时将组件重新定位，等等。这些特性的实现都用到了诸如glues、fillers、struts、rigid区域这样的不可见组件(invisible components)。

glue组件可扩充到必要的大小，以便填满框式布局中相邻组件之间的空间。这就导致在水平布局中会产生最大宽度，在垂直布局中产生最大高度。利用glues组件，可以使组件与组件之间以及组件与容器之间的额外空间得到平均分布。可以用以下两个静态方法分别创建针对水平布局和垂直布局的glue组件：

```
public static Component CreateHorizontalGlue()  
public static Component CreateVerticalGlue()
```

strut大小是固定的。可以用这个组件向框式布局内的两个组件之间加入固定数量的间隔。例如，在一个水平布局中，这个组件就可以在两个组件之间加入一定的间隔。strut也可以用来使方框本身具有一定的大小。例如，在一个垂直放置的方框中，strut可用来使方

框的宽度具有某个特定的值。Strut 的尺寸在没有指定的方向上基本上是无限制的。所以，对水平方向的方框来说，其高度就没有限制。

**rigid** 区域占有一定的尺寸。**rigid** 区域组件的功能类似于 struts 组件。可以指定该组件的宽度和高度。下面是创建 **rigid** 区域组件的静态方法：

```
public static Component CreateRigidArea(Dimension d)
```

**fillers** 组件允许指定尺寸的最小值、最大值以及某个期望的值。过滤器是一些内部类 **Box.Filler** 对象。当把一个过滤器添加到水平方框中的两个组件之间时，它就会保持最小的宽度值，并且保证容器的高度最小。可以用以下所示的方法创建过滤器对象：

```
Box.Fill(new Dimension(w1,h1), //minimum size  
new Dimension(w2,h2), //prefered size  
new Dimension(w3,h3)) //maximum size
```

### 5.7.3 BoxLayout 代码示例 (利用不可见的组件放置其他组件)

清单 5.6 用框式布局创建了一个小程序，其中分别在不同的窗格里显示了几对按钮：B1 和 B2，B3 和 B4，等等，以此演示象 glue、fillers、rigid areas 以及 struts 这样的不可见组件的效果。每一个窗格都有一个显示其演示类型的标题条（参看图 5-11）。不可见组件已经被加到了按钮与按钮之间，按钮与窗格之间，以及窗格与窗格之间。

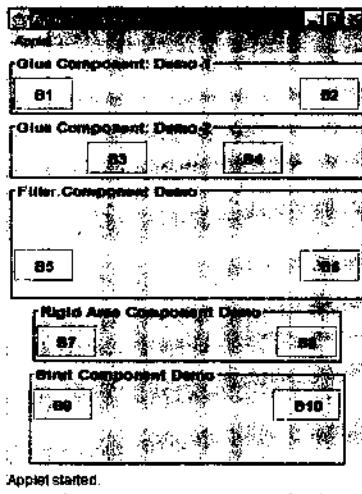


图 5-11 框式布局能够用不可见组件放置其他的组件

#### 清单 5.6 使用不可见组件的 *BoxLayout* (*Tbox1.java*) 示例

```
// Demonstrates the positioning of components in a box using  
// invisible components  
  
/*  
* <Applet code = TBox1 width = 300 height = 300 >  
* </Applet >
```

```
*/  
  
import javax.swing.*;  
import javax.swing.border.*;  
import java.awt.*;  
  
public class TBox1 extends JApplet {  
    Container container = null;  
  
    public void init() {  
        // 1. Get the handle on the applet's content pane.  
        container = getContentPane();  
  
        // 2. Create a vertical box and add it to the applet.  
        Box baseBox = Box.createVerticalBox();  
        container.add(baseBox);  
  
        // 3. Create the demo panel1 with box layout to display  
        // buttons B1 and B2.  
        // a) Create the panel and set the box layout.  
        JPanel glueDemoPanel1 = new JPanel();  
        glueDemoPanel1.setLayout(new BoxLayout(glueDemoPanel1,  
                                              BoxLayout.X_AXIS));  
  
        // b) Set the title border around the panel.  
        TitledBorder border1 = new TitledBorder(  
            new LineBorder(Color.black),  
            "Glue Component: Demo - 1");  
  
        border1.setTitleColor(Color.black);  
        glueDemoPanel1.setBorder(border1);  
  
        // c) Add the buttons B1 and B2 with a glue component  
        // between them.  
        glueDemoPanel1.add(new JButton("B1"));  
        glueDemoPanel1.add(Box.createHorizontalGlue());  
        glueDemoPanel1.add(new JButton("B2"));  
  
        // d) Add the panel to the base box.  
        baseBox.add(glueDemoPanel1);  
  
        // 4. Create the demo panel - 2, assign box layout and add  
        // buttons B3 and B4.  
        // a) Create the glue panel 2 and assign the box layout.  
        JPanel glueDemoPanel2 = new JPanel();  
        glueDemoPanel2.setLayout(new BoxLayout(glueDemoPanel2,  
                                              BoxLayout.X_AXIS));  
  
        // b) Add a titled border to the panel.  
        TitledBorder border2 = new TitledBorder(  
            new LineBorder(Color.black),
```



```
// b) Assign the title border around the panel.  
TitledBorder border4 = new TitledBorder(  
    new LineBorder(Color.black),  
    "Rigid Area Component Demo");  
border4.setTitleColor(Color.black);  
rigidADemoPanel.setBorder(border4);  
  
// c) Add buttons B7 and B8 to the rigid area panel.  
// Also add a rigid area in the middle of the buttons.  
rigidADemoPanel.add(new JButton("B7"));  
rigidADemoPanel.add(Box.createRigidArea(new  
Dimension(150,0)));  
rigidADemoPanel.add(new JButton("B8"));  
  
// d) Add the panel to the base box.  
baseBox.add(rigidADemoPanel);  
  
// 7. Create the strut panel, assign the box layout, and  
// add the buttons B9 and B10.  
// a) Create the panel and assign the box layout  
JPanel strutDemoPanel = new JPanel();  
strutDemoPanel.setLayout(new BoxLayout(strutDemoPanel,  
    BoxLayout.X_AXIS));  
  
// b) Set the titled border around the panel.  
TitledBorder border5 = new TitledBorder(  
    new LineBorder(Color.black),  
    "Strut Component Demo");  
border5.setTitleColor(Color.black);  
  
// c) Add the buttons B9 and B10 to the panel. Also assign  
// the horizontal strut in the middle of the buttons.  
strutDemoPanel.setBorder(border5);  
strutDemoPanel.add(new JButton("B9"));  
strutDemoPanel.add(Box.createHorizontalStrut(150));  
strutDemoPanel.add(new JButton("B10"));  
  
// d) Add the panel to the base box.  
baseBox.add(strutDemoPanel);  
}  
}
```

## 代码详析

小程序 TBox1 中的代码片段 1 获得了一个对于其内容窗格的引用。代码片段 2 创建了一个垂直方向的方框，下面代码片段中创建的窗格将被加到这个方框中。

代码片段 3 创建了演示 glue 组件的窗格 1，并给它赋予水平框式布局。窗格上加了一个

带标题条的边框，相应的标题说明了演示的类型。该窗格上还加了按钮 B1 和 B2。这两个按钮之间添加了一个 glue 组件。可以调用 Box() 类中的静态方法 createHorizontalGlue()，来创建水平方向的 glue 组件。注意，水平方向的 glue 组件产生的效果是，它把 B1 和 B2 这两个按钮推到了容器的两边（参看图 5-11）。

代码片段 4 类似于代码片段 3，不同之处是，它沿水平方向在按钮（B3 和 B4）和窗格之间添加了一个 glue 组件。这个 glue 组件产生的实际效果是，将两个按钮放置在具有相同间隔的位置上（参看图 5-11）。

代码片段 5~7 使用了 fillers、struts 和 rigid areas 这几个不可见组件。代码片段 5 把按钮 B5 和 B6 加到了框式布局的窗格上。另外，还在按钮之间加入了一个具有指定最小值、最大值和期望值的 fillers 组件。该组件维护的按钮之间最小间隔至少是 50 个象素。Filler 组件也确保窗格的最小高度为 75 个象素（参看图 5-11）。

代码片段 6 把按钮 B7 和 B8 加到了框式布局中的相关窗格上。按钮之间加入了宽度为 200 个象素的水平方向（高度为 0 象素）的 rigid area 组件。于是，rigid area 组件就始终保持按钮之间有 200 个象素的固定间隔（参看图 5-11）。

代码片段 7 使用了位于代码片段 6 中所描述的 rigid area 区域中水平方向的 strut 组件。注意，strut 组件沿水平方向也会产生同 rigid area 一样的效果。唯一的不同之处是，水平方向的 strut 组件没有高度限制，窗格能够扩展到所有剩余的空间上。

#### 5.7.4 BoxLayout 代码示例（使用多重嵌套）

清单 5.7 演示了多重嵌套的框式布局能够产生无序网格布局的效果。这里的布局模型类似于无序网格布局演示程序给出的模型。因此，本程序的目的是将各个组件进行布局，最后得到图 5-12 所示的外观结果。

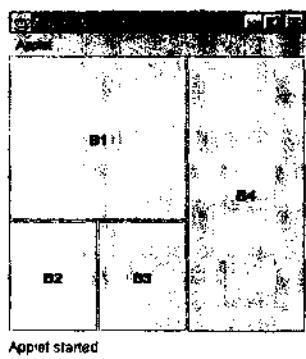


图 5-12 利用框式布局得到的类似于无序网格布局效果

##### 清单 5.7 使用多重嵌套的 BoxLayout (Tbox2.java) 示例

```
// Multiple-nesting of boxes can produce the effect of
// grid bag layout
/*
 * <Applet code = Tbox2 width = 250 height = 250 >
```

```
* </Applet>
*/
import javax.swing.*;
import java.awt.*;
public class TBox2 extends JApplet {
    Container container = null;
    public void init() {
        // 1. Get the handle on the applets' content pane
        // and set the background color to white.
        container = getContentPane();
        container.setBackground(Color.white);
        // 2. Create a horizontal box and add it to the applet.
        Box baseBox = Box.createHorizontalBox();
        container.add(baseBox);
        // 3. Create a vertical box and add it to the base box.
        Box vBox = Box.createVerticalBox();
        baseBox.add(vBox);
        // 4. Create button B1 and add it to vBox.
        JButton b1 = new JButton("B1");
        b1.setAlignmentX(Component.CENTER_ALIGNMENT);
        b1.setMaximumSize(new Dimension(150, 150));
        vBox.add(b1);
        // 5. Create a horizontal box and add it to vBox.
        Box hBox = Box.createHorizontalBox();
        vBox.add(hBox);
        // 6. Create button B2 and add it to hBox.
        JButton b2 = new JButton("B2");
        b2.setAlignmentY(Component.CENTER_ALIGNMENT);
        b2.setMaximumSize(new Dimension(75, 100));
        hBox.add(b2);
        // 7. Create another button B3 and add it to hBox.
        JButton b3 = new JButton("B3");
        b3.setAlignmentY(Component.CENTER_ALIGNMENT);
        b3.setMaximumSize(new Dimension(75, 100));
        hBox.add(b3);
        Listing 5.7 continued
        // 8. Create the vertical box (vBox1) and
        // add it to the base box.
        Box vBox1 = Box.createVerticalBox();
        baseBox.add(vBox1);
```

```
// 9.Create the button b4 and add it to vBox1.  
JButton b4 = new JButton("B4");  
b4.setAlignmentX(Component.CENTER_ALIGNMENT);  
b4.setMaximumSize(new Dimension(100, 250));  
vBox1.add(b4);  
}  
}
```

## 代码详析

小程序 TBox1 中的代码片段 1 获得了对其内容窗格的引用，并重置了容器的背景色。代码片段 2 创建了一个水平方向的方框，并把它加到了一个内容窗格上。这个方框作为基本方框，用来添加其他的组件。

代码片段 3 创建一个垂直方框，并将其加到了基本方框中。该方框中要装载一个按钮和一个水平方框。代码片段 4 创建并放置按钮 B1。注意，你可能不得不根据需要设置按钮的对齐方式。按钮的大小调整到了期望的尺寸。

### 说明：

如果组件放入到一个方框容器中时没有象所期望的那样显示出来，那么需检查一下对齐方式，并对组件大小进行调整。

代码片段 5 创建一个水平方框，并将其加到了垂直方框中。水平方框里加入了 B2 和 B3 两个按钮。代码片段 6 和片段 7 创建了按钮 B2 和 B3，并以适当的对齐方式和大小把它们加入到方框中。

代码片段 8 创建一个垂直方框，并将其加到了基本方框中。该垂直方框里加入了按钮 B4，该按钮从程序空间的顶部一直扩展到了底部。代码片段 9 创建了按钮 B4，并以适当的对齐方式和大小将它加入到垂直方框中。

# 第 6 章 Swing 设计

Swing 组件集允许 Java 开发人员利用现成的、复杂的构件快速建立应用程序前端。Swing 组件集的基础植根于其早先的抽象窗口工具包 (AWT)，而且新的组件集也与 AWT 中的 API 相兼容。最初的目标之一是，提供一个能够支持本机系统 (native system) “外观” (look-and-feel) 的 API，这是因为作为不依赖于平台的 Java 小程序和 Java 应用程序必须能够运行在各种客户机上。除此之外，语言本身也要有特殊的环境“外观”。

为了实现这个目标，Swing 开发小组一开始就采用了模型 - 视图 - 控制器 (Model-View-Control，简记为 MVC) 体系结构。这种建立组件的方法还允许使用模型驱动的程序设计技术，因此，开发人员就可以通过组件建立自己的数据模型来显示数据，这样也隐藏了提取应用程序数据这样的复杂操作。例如，假设要利用模型驱动程序设计技术初始化一个树组件，用它来显示计算机文件系统的目录结构，那么，获得文件系统数据的方法就可以打包在树的模型中。

随着使用 MVC 体系结构的工作逐步发展，Swing 开发小组认为需要修改 MVC 的设计，以避免将有紧密耦合关系的视图与控制程序分割开来。本章第一节将讨论这个修改过的设计。

这一章的后面几节将介绍组件模型和“组件 UI 代表” (component UI delegate) 的基础内容，这里的“组件 UI 代表”是当一个模型被删除时所剩余的那部分组件。本章最后给出了一个模拟 Swing 按钮及其 M - UI 体系结构的例子。

## 6.1 M - UI：简化的 MVC

模型 - 视图 - 控制器 (MVC) 体系结构已经通过一个叫做 SmallTalk 的语言引入到了程序设计中。利用 MVC 方法，一个具有可视化接口的软件程序可以分解为三个相互分离的部分 (参见图 6-1)：



图 6-1 模型 - 视图 - 控制器 (MVC) 形成的分离效果

- 模型，它包含了要编程的系统数据
- 视图，它以可视化方式显示数据

- 控制器，它通过对数据的操作把用户在视图上输入的数据传送给模型  
MVC 体系结构具有以下几个优点：
  - MVC 体系结构可以大大减轻多视图数据表示系统编程的工作量。
  - MVC 体系结构也提供了构成程序各个部件之间的清晰划分。这样就允许每一个域范围内的问题能够相互独立地加以处理。
  - 模型与视图之间的绑定可以在运行时进行，而不是在编译时进行。这就允许在运行程序的任何时候将用户界面改变成另外一个外观。
  - 最后，与单模块方法相比较，MVC 体系结构支持基于组件的软件开发。

由于 MVC 体系结构具有上述优点，Swing 开发小组开始把这个方法应用于他们想设计的每一个组件中。注意，这是 MVC 体系结构对于整个软件的传统应用方法相比较而言的。但是很快，Swing 开发小组就看到了控制器与视图之间紧密的耦合关系。这是由于有这样一个事实：要写一个能处理所有类型视图的控制器是极为困难的。这就导致开发小组寻求将视图和控制器协调起来的方法。每个组件的协调部分就叫做该组件的“UI - 代表”。这样，MVC 体系结构就修改成了 M - UI 体系结构。

## 6.2 组件模型

组件的模型表示它的数据。在 Swing 中，分离的设计界面定义成表示每一个组件的数据模型。该界面允许使用 Swing 组件的自定义模型。对于显示某个特定应用程序数据的组件来说，自定义模型是非常有用的。例如，可以定义 Swing 树模型的用户实现形式，以此显示计算机上文件系统的层次结构。类似地，在文本窗格内，可以通过定制模型的实现方式来按照相应的格式显示信息内容。

### 6.2.1 模型类型

组件的模型可分为三类。有的组件模型表示主应用程序的数据。例如，TreeModel, TableModel,ListModel, Document 和 ComboBoxModel 界面就支持应用程序数据模型。这些模型界面在建立能够减少编程复杂性的定制模型时很有用处的。

第二种类型的模型只存放与 GUI 相关的数据，比如按钮状态、选择信息等等。这样的模型有：ButtonModel、ListSelectionModel 以及 TableColumnModel。第三种类型的模型存放应用程序的数据和 GUI 数据。BoundedRangeModel 就是这种类型的模型。

### 6.2.2 模型改变和通知

通过用户的交互动作，组件模型的数据值会发生变化。这些变化必须在数据视图中反映出来。为此目的，模型要通过激活事件来通知那些它所感兴趣的监听程序。Swing 模型根据 JavaBeans 所引入的事件模型发送通知。

### 6.2.3 更改数据视图

要为一个给定的数据模型生成多重视图，该视图就不应该依赖于模型。然而，如果模型数据发生了任何变化，却应在视图中将此变化反映出来。要更新视图，监听程序必须检查由

模型所激活的事件并完成必要的操作。

#### 说明：

一般情况下，没有必要在所写的每一个 Swing 程序中都使用模型 API。但在某些场合，模型有助于减少程序设计的复杂性。特别是在处理表示应用程序数据而不是 GUI 数据的模型时，情况尤其如此。

## 6.3 “UI-代表”

如果是从组件中把数据模型删除的，那么组件的剩余部分就是“UI-代表”。UI-代表是协调在一个实体内的视图和控制程序。在 Swing 中，可见的 UI-代表是 javax.swing.plaf 包里的 ButtonUI、ComboBoxUI、TreeUI、TableUI 等。

### 6.3.1 ComponentUI 类

所有 UI 代表的超级类是 javax.swing.plaf.ComponentUI。这个类中的方法完成 UI 的安装和卸载。这些方法还要处理组件的几何结构及对它们的绘制。installUI() 方法负责为组件的属性（如字体、颜色以及边界等）赋予缺省值，它还安装组件的布局并添加事件监听器。方法 uninstallUI() 将会撤消 installUI() 方法所执行过的所有操作。

## 6.4 可插入外观

对于象 Java 这样的能够运行在不同平台上的语言来说，改变视图以适应当前平台，可以使用户熟悉 Java 应用程序，可插入外观使得用户无需重新启动应用程序就可以切换其外观。它还允许程序识别其所依附的平台，并且把应用程序的外观改为与此平台相似的外观。

开发人员可以创建这样的组件：它们能够自动地具有某种样式和行为，而不管其运行在哪一种操作系统平台上。Swing 组件是用 Java 语言写的，其中没有用到任何底层的组件。这使我们可以得到不依赖于本机系统的可插入外观。可插入外观是 MVC 设计的一个自然结果。由于数据视图和数据模型之间所具有的这种可分离性，所以，把组件插入到其他视图就很简单而且直截了当。

Swing 的可插入外观 API 支持以下特性：

- 跨平台，或缺省外观——这也叫做 Java 外观（以前被称为 Metal 外观）。跨平台外观使得程序无论运行在哪一种平台上，都能够保持 Java 外观。当前端由 Swing 创建时，它被缺省地设置为具有 Java 外观（除非编程时把缺省值改成了其他的外观）。
- 系统外观——这是底层操作系统的外观。基于 Swing 的前端可以设置成任何底层平台的外观，比如 Motif、Windows 或者 Macintosh。
- 动态改变外观——这一特性是 MVC 设计的结果。可以在应用程序运行时改变程序前端的外观（也许是通过单击工具条上的按钮）。

### 6.4.1 外观程序包

Java JDK 1.2 的 Swing 库中有四种外观类型。这其中就包括 Java 外观程序包，它也作为缺省外观。其他外观程序包有：Motif 类型、Windows 类型和 Macintosh 类型。这里的 Macintosh 类型是 JDK 1.2 中新引入的，其支持库需要单独获得，因为它没有与 JDK 一同发布。

Java 外观程序包的名字是 javax.swing.plaf.metal.\*。JDK 1.2 还包含 Windows 类型和 Macintosh 类型的程序包。这两种类型的包的名字是：

```
com.sun.java.swing.plaf.motif.*  
com.sun.java.swing.plaf.windows.*
```

这种命名约定表示相应的外观与 Sun 公司以外的操作系统有关。

### 6.4.2 UIManager 类

UIManager 类提供了控制应用程序外观的句柄。该类包含把一种特定的外观赋予应用程序或者取得应用程序当前外观的静态方法。要把特定外观作为缺省外观，可以调用 UIManager 类中以下的静态方法：

```
public static void setLookAndFeel(String className)
```

如果所赋的外观没有被发现，该方法就会抛出 ClassNotFoundException 异常。它还可能抛出其他异常，比如 InstantiationException（如果新的实例不能被创建），IllegalAccessException（如果这个类不能被访问），UnsupportedLookAndFeelException（如果不支持此种外观）。

```
public static void setLookAndFeel(LOOKAndFeel newLF)
```

如果不支持所赋的外观，该方法就会抛出 UnsupportedLookAndFeelException 异常。

其他（有益）的方法可以获得外观的类型，现说明如下：

```
public static String getSystemLookAndFeelClassName()
```

该方法返回实现本地系统外观的外观类名称。如果没有本地外观，则返回 Java 外观的名称。

```
public static String getCrossPlatformLookAndFeelClassName()
```

该方法返回 Java 外观类的名称。

### 6.4.3 LookAndFeel 类

LookAndFeel 是一个抽象类，它包含了所有不同外观的基本信息。该类是一个根类，其他特定的外观类都是扩展它得到的，如图 6-2 所示。LookAndFeel 类有三个主要的任务：识别当前的外观，确定支持该外观的平台，以及建立缺省的外观表。想建立自己外观的程序员必须建立其自己的外观子类。

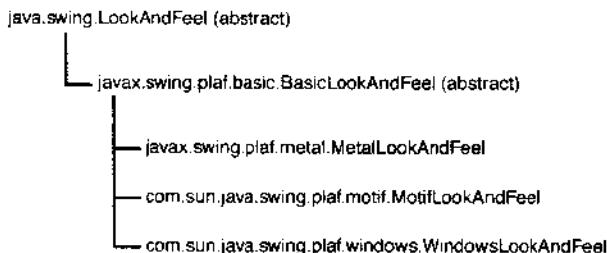


图 6-2 特定的外观类的层次结构

在这个层次图中，MetalLookAndFeel、MotifLookAndFeel 以及 WindowsLookAndFeel 初始化了系统颜色的缺省值（比如桌面背景色、标题条的颜色、文本的颜色，等等），并为每个组件指定了类名，例如 com.sun.java.swing.plaf.motif.MotifButtonUI。

#### 6.4.4 外观的动态更新

当应用程序运行时，其外观可以动态地改变。但是，由于组件是在其构造函数内对 UI 代表进行初始化的，所以运行时对外观的改变不能自动地传给 UI 代表。这样就必须沿着组件的包含层次关系通过编程来改变 UI 代表。

为了处理组件的动态修改，Swing 在 SwingUtilities 类中提供了静态方法 updateComponentTreeUI (Component c)。这个方法要求包含层次中的每一个组件都要调用 updateUI ()。通常情况下，必须传递顶层容器作为 updateComponentTreeUI (Component c) 方法的参数值。调用这个方法之后，还需要确认容器调用了 validate () 方法。

### 6.5 Swing 设计的模拟代码示例

清单 6.1 是一个用简单代码模拟 Swing 设计的程序示例。这个程序提供了利用 Swing 组件进行完整设计的一个概览。注意，各个对象分别代表了不同的组件模型及其 UI 代表。程序中叫做 MyButton 的组件示例，能够通过使用可插入外观方法在两种样式（矩形和椭圆形）之间改变。图 6-3 显示了 MyButton 的缺省样式和椭圆样式。



图 6-3 在两种不同外观（缺省的和椭圆的）中的 MyButton 组件

这个程序由三个源文件组成，即 TMyButton.java、MyButton.java 和 MyUIManager.java。文件 TMyButton.java 包含了实现 MyButton 按钮的程序。源文件 MyButton.java 包含了利用 Swing 设计对 MyButton 组件的实现。文件 MyUIManager.java 则包含了处理组件外观的代码。

**清单 6.1 MyButton 组件的 Swing 设计 (TMyButton.java)**

```
// Application program that implements MyButton component.  
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
  
class TMyButton extends JFrame { public TMyButton() {  
    // 1. Assign a name to the frame, get the content pane of the  
    // frame, and assign white color to the background of the  
    // frame.  
    super("My Button Frame");  
    Container contentPane = this.getContentPane();  
    contentPane.setBackground(Color.white);  
  
    // 2. Assign an oval shaped look - and - feel to the buttons.  
    // In reality, only the look is altered; the feel is not changed!  
    // The default look - and - feel is rectangular and has the  
    // program identifier "Default".  
    // UIManager.setLookAndFeel("Default");  
    UIManager.setLookAndFeel("Oval");  
  
    // 3. Create a Play button object and set an action command.  
    // Also register an action listener.  
    MyButton playButton = new MyButton();  
    playButton.setText("Play");  
    playButton.setActionCommand("playbutton");  
    playButton.addActionListener(new ButtonListener());  
  
    // 4. Create a Stop button object and set an action command.  
    // Also register an action listener.  
    MyButton stopButton = new MyButton("Stop");  
    stopButton.setActionCommand("stopbutton");  
    stopButton.addActionListener(new ButtonListener());  
  
    // 5. Set the frame to the grid layout and add the Play and  
    // Stop buttons.  
    contentPane.setLayout(new GridLayout(1, 2));  
    contentPane.add(playButton);  
    contentPane.add(stopButton);  
  
    // 6. Add the window listener to close the frame  
    // and display it with the specified size.  
    addWindowListener(new WindowEventHandler());  
    setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
    setSize(300, 75); // width = 300, height = 75  
    show(); // Display the frame
```

```
}

// 7. Listener class to close the parent frame.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        dispose();
        System.exit(0);
    }
}

// 8. Action listener that receives events whenever the Play
// or Stop button is clicked.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent evt) {
        MyButton button = (MyButton) evt.getSource();

        if (button.getActionCommand().equals("playbutton")) {
            System.out.println("You clicked the Play button.");
        }
        else if (button.getActionCommand().equals("stopbutton")) {
            System.out.println("You clicked the Stop button.");
        }
    }
}

// 9. The main method.
static public void main(String[] args) {
    new TMyButton();
}

// File Name: MyButton.java
// 10. The component called MyButton that is built by using Swing design.
import javax.swing.*;
import javax.swing.plaf.*;
import java.awt.*;
import java.awt.event.*;

// 11. The component class MyButton is a subclass of JComponent. class MyButton
extends JComponent { MyButtonModel model = null;
    String text = "";
    Font font = null;
    FontMetrics fontMetrics = null;
    int border = 10;
    boolean buttonPressed = false;

    // 12. MyButton constructor without any arguments.
```

```
public MyButton() {
    this(null);
}

// 13. MyButton constructor that takes the text label argument.
public MyButton(String text) {
    // Assign the data model to the component.
    setModel(new DefaultMyButtonModel());
    // Initialize the text label.
    this.text = text;
    // Obtain the default font.
    font = getFont();
    // If there is no default font, assign a new font.
    if (font == null) {
        font = new Font("SanSerif", Font.BOLD, 14);
    }
    // Retrieve the font metrics of the font.
    fontMetrics = getFontMetrics(font);
    // Register a mouse listener.
    addMouseListener(new ButtonMouseListener());
    // Update the view of the button.
    updateUI();
}

// 14. Method definition of update.
public void updateUI() {
    // Assign proper look-and-feel for the view depending on the
    // specified look-and-feel.
    if (MyUIManager.getLookAndFeel().equals("Default")) {
        super.setUI(new MyButtonUI());
    }
    else if (MyUIManager.getLookAndFeel().equals("Oval")) {
        super.setUI(new MyButtonOvalUI());
    }
}

// 15. This class is the UI delegate. It has been defined as an
// inner class to simplify the program. This class provides the default
// view of the button.
class MyButtonUI extends ComponentUI {

// 16. Constructor.
public MyButtonUI() {
    repaint(); // Call repaint for further actions.
}
```

```
// 17. Retrieve the minimum size of the component.  
public Dimension getMinimumSize() {  
    int w = fontMetrics.stringWidth(text);  
    int h = fontMetrics.getHeight();  
    return new Dimension(w + 2 * border, h + 2 * border);  
}  
  
// 18. Retrieve the preferred size of the component.  
public Dimension getPreferredSize() {  
    return getMinimumSize();  
}  
  
// 19. Retrieve the maximum size of the component.  
public Dimension getMaximumSize() {  
    return new Dimension(Short.MAX_VALUE, Short.MAX_VALUE);  
}  
  
// 20. The update method definition.  
public void update(Graphics g, JComponent c) {  
    if (c.isOpaque()) {  
        g.setColor(c.getBackground());  
        g.fillRect(0, 0, getWidth(), getHeight());  
    }  
    paint(g, c);  
}  
  
// 21. The paint method definition.  
public void paint(Graphics g, JComponent c) {  
    g.setFont(font);  
  
    if (buttonPressed == false) {  
        buttonView(g, Color.lightGray, Color.black);  
    }  
    else {  
        buttonView(g, Color.gray, Color.white);  
    }  
}  
  
// 22. The buttonView method definition.  
public void buttonView(Graphics g, Color background, Color foreground) {  
    g.setColor(background);  
    g.fillRect(0, 0, getSize().width, getSize().height, true);  
    g.setColor(foreground);  
    int x = (getSize().width - fontMetrics.stringWidth(text))/2;  
    int y = (getSize().height + fontMetrics.getHeight()  
            - fontMetrics.getDescent()) / 2;  
    g.drawString(text, x, y);  
}
```

```
}

// 23. Implementation of oval shaped look - and - feel for MyButton.
// This view of the button is used as an alternative to its default view.
class MyButtonOvalUI extends ComponentUI {

    // 24. Constructor.
    public MyButtonOvalUI() {
        repaint();
    }

    // 25. Retrieve the minimum size of the component.
    public Dimension getMinimumSize() {
        int w = fontMetrics.stringWidth(text);
        int h = fontMetrics.getHeight();
        return new Dimension(w + 2 * border, h + 2 * border);
    }

    // 26. Retrieve the preferred size of the component.
    public Dimension getPreferredSize() {
        return getMinimumSize();
    }

    // 27. Retrieve the maximum size of the component.
    public Dimension getMaximumSize() {
        return new Dimension(Short.MAX_VALUE, Short.MAX_VALUE);
    }

    // 28. The update method definition.
    public void update(Graphics g, JComponent c) {
        if (c.isOpaque()) {
            g.setColor(c.getBackground());
            g.fillRect(0, 0, getWidth(), getHeight());
        }
        paint(g, c);
    }

    // 29. The paint method definition.
    public void paint(Graphics g, JComponent c) {
        g.setFont(font);

        if (buttonPressed == false) {
            buttonView(g, Color.gray, Color.white);
        }
        else {
            buttonView(g, Color.lightGray, Color.black);
        }
    }
}
```

```
}

// 30.The button view definition.
public void buttonView(Graphics g,Color background,Color foreground) {
    g.setColor(background);
    g.fillOval(0,0,getSize().width,getSize().height);
    g.setColor(foreground);
    int x = (getSize().width - fontMetrics.stringWidth(text))/2;
    int y = (getSize().height + fontMetrics.getHeight()
              - fontMetrics.getDescent())/2;
    g.drawString(text,x,y);
}

// 31.When the user presses or releases the mouse over a button,
// the functionality from the implementation methods from this
// mouse listener class are executed.
class ButtonMouseListener extends MouseAdapter {
    public void mousePressed(MouseEvent evt) {
        requestFocus();
        buttonPressed = true;
        ((MyButton) evt.getSource()).repaint();
    }

    public void mouseReleased(MouseEvent evt) {
        buttonPressed = false;
        ((MyButton) evt.getSource()).repaint();
        ActionEvent actionEvent = new ActionEvent(
            MyButton.this,
            ActionEvent.ACTION_PERFORMED,
            MyButton.this.getActionCommand());
        processEvent(actionEvent);
    }
}

//Supports the action listener.
ActionListener actionListener;

// 32.Method to register an action listener with the button.
public void addActionListener(ActionListener l) {
    actionListener = AWTEventMulticaster.add(actionListener,l);
}

// 33.To figure out the action events.
protected void processEvent(AWTEvent evt) {
    if (evt instanceof ActionEvent) {
        processActionEvent((ActionEvent) evt);
```

```
}

else {
    super.processEvent(evt);
}

}

// 34. To process the action events.
protected void processActionEvent(ActionEvent evt) {
    if (actionListener != null) {
        // Execute the action performed from the listener class.
        actionListener.actionPerformed(evt);
    }
}

// 35. Method to assign the button model.
public void setModel(MyButtonModel model) {
    this.model = model;
}

// 36. Retrieve the action model.
public MyButtonModel getModel() {
    return model;
}

// 37. Assign the text label to the button.
public void setText(String text) {
    this.text = text;
    invalidate();
    repaint();
}

// 38. Retrieve the text label of the button.
public String getText() {
    return text;
}

// 39. To assign the action command.
public void setActionCommand(String actionCommand) {
    getModel().setActionCommand(actionCommand);
}

// 40. To retrieve the action command.
public String getActionCommand() {
    String ac = getModel().getActionCommand();
    if (ac == null) {
        ac = getText();
    }
    return ac;
}
}
```

```
// 41.Define a simple button model interface.  
interface MyButtonModel {  
    public void setActionCommand(String actionCommand);  
    public String getActionCommand();  
}  
  
// 42.Define the default button model class that implements the  
// MyButtonModel interface.  
class DefaultMyButtonModel implements MyButtonModel {  
    protected String actionCommand = null;  
    // 43.Assign an action command.  
    public void setActionCommand(String actionCommand) {  
        this.actionCommand = actionCommand;  
    }  
    // 44.Retrieve the action command that has been assigned.  
    public String getActionCommand() {  
        return actionCommand;  
    }  
}  
  
// File Name: MyUIManager.java.  
// 45.The class MyUIManager.This class figures out the look - and - feel to be  
// assigned to a button object.The class contains static method that can  
// be invoked from the application class to assign the look - and - feel.  
class MyUIManager {  
    // Store the look - and - feel string.  
    static String lfType = "Default";  
    // 46.Assign a new look - and - feel.  
    static public void setLookAndFeel(String type) {  
        lfType = type;  
    }  
    // 47.Retrieve the current look - and - feel.  
    static public String getLookAndFeel() {  
        return lfType;  
    }  
}
```

## 代码详析

在程序 TMyButton.java 中，代码片段 1 为框架指定了标题，取得一个内容窗格的句柄，并把框架背景色设置为白色。代码片段 2 将椭圆形外观赋予框架上的按钮。代码片段 3 创建了 Play 按钮对象，并赋予它一个动作命令，该片段还为按钮注册了一个动作监听程序。代码片段 4 对 Stop 按钮完成类似的功能。

代码片段 5 准备了一个包含一行二列的网格框架，并把 Play 按钮和 Stop 按钮加入到其

上。代码片段 6 注册窗口事件处理器，其中包括关闭框架、对框架进行配置等处理程序。代码片段 7 给出了表示窗口事件处理器的内部类。代码片段 8 是动作监听类，它负责监听框架上的按钮激活事件。在 actionPerformed() 方法中，每一个按钮都能通过动作命令识别，程序会在标准输出设备上打印出该按钮被按下的信息。代码片段 9 给出了把 TMyButton 类实例化的主要方法。

代码片段 10（在 MyButton.java 源程序中）给出了利用 Swing 设计创建 MyButton 组件的代码。代码片段 11 声明了 MyButton 类，它是 JComponent 类的扩展。代码片段 12 和片段 13 是类的构造函数。注意，在进行类的初始化时，代码片段 13 调用了 updateUI() 方法。代码片段 14 给出了 updateUI() 方法的定义。基本上，能够找到把视图赋予组件的代码，这要依赖于所需的外观。

代码片段 15 定义了用代码片段 16 中的构造函数所创建组件的 UI 代表。代码片段 17~19 返回将要布局的组件大小的最小值，期望值和最大值。代码片段 20 是 update() 方法，它调用了 paint() 方法对组件的视图进行绘制。代码片段 21 定义了 paint() 方法。描述按钮图形形状是代码片段 22 中给出的另外一个函数 buttonView()。代码片段 22~30 给出了定义椭圆形外观的按钮视图的代码。

代码片段 31 是处理鼠标事件的监听类。该类的对象与按钮一同被注册，以便识别何时有按钮操作。这个类基本实现了 mousePressed() 和 mouseReleased() 方法。片段 32 中的代码注册了按钮的动作监听程序。代码片段 33 和片段 34 识别并处理这个动作事件，以激活与按钮对象一起注册的动作监听类中的 actionPerformed() 方法。

代码片段 35 和片段 36 中有两个方法，它们分别把已经被赋予按钮的数据模型再赋给另一个量，以及返回这个数据模型。代码片段 37 和片段 38 片断中的两个方法分别把显示在按钮组件上的文本标签赋给另一个量，以及返回这个文本标签。类似地，代码片段 39 和片段 40 中的两个方法则分别把按钮模型中的动作命令赋给另一个量，以及返回该动作命令。

代码片段 41 定义了 MyButtonModel 接口，该接口只包含两个抽象方法。代码片段 42 给出了缺省按钮模型的代码，它实现了 MyButtonModel 接口。代码片段 43 和 44 片段中的方法分别实现了接口中的相应方法。

代码片段 45 定义了按钮的 UI 管理器。UI 管理器是一个被称为 MyUIManager 的类，该类存放在一个叫做 MyUIManager.java 的单独文件中。UI 管理器包含了对按钮对象赋以外观值以及取得其外观值的两个静态方法。

# 第 7 章 图标、标签和边界

与以前的抽象窗口工具包 (AWT) 不同, Swing 工具箱支持可用于标签、按钮、复选框等各种组件的图标。

本章将讨论如何利用图形环境对象 (graphics context object) (`java.awt.Graphics` 类的对象) 所提供的功能来建立自己的图标。可以看到怎样用 Java 所支持格式 (如 GIF 和 JPEG 等) 的图像文件来建立图标。

Swing 常数用来把文本信息放在各种组件内, 或者设置组件自身的方向。这对整个 Swing 集合都是一样的, 适合于本章中所给出的各种组件。

在用户界面上, 一个简短的文本标签可以告诉用户某个特定组件的目的。Swing 标签的目标就在于此, 它支持文本字符串和图标。本章将探讨 Swing 标签的用法, 如何创建它们, 如何把图标连到文本上, 以及如何控制文本和图标的位置。本章将给出一个例子来说明怎样使用标签类所支持的 API。

Swing 库提供了若干个能够在应用程序或小程序中实现的现成的边界, 这些边界可以装饰用户界面的外观效果。本章将要讨论这些边界的有关内容; 还将讨论在 Swing 所提供的边界不能满足需要的情况下, 怎样实现定制的边界。

## 7.1 图标

图标可以显示在按钮、标签和菜单项上, 用以描述组件的用途。图标可以由 `java.awt.Graphics` 类中提供的一些功能函数 (如直线、矩形、椭圆等) 来建立, 也可以由 Java 支持的图像文件 (如 .gif 和 .jpeg) 创建。

### 7.1.1 Icon 接口

`Icon` 接口是一个设计级的抽象类, 它代表一个将要用于用户界面中的图标。该接口已经存在于 `javax.swing` 中。可以通过实现 `Icon` 接口的定制类来创建图标。图标具有固定的宽度和高度。以下是在接口中必须实现的三个方法:

```
public abstract void paintIcon(Component c,
                               Graphics g,
                               int x, int y);
public abstract void getIconWidth();
public abstract void getIconHeight();
```

方法 `paintIcon ()` 在图标的坐标位置上画图。在实际的代码中, `Graphics` 对象用于激活所需的绘图方法。`Component` 对象恢复了用于画背景色, 前景色等的属性。

### 7.1.2 创建图标

现在,我们用实现接口 `Icon` 的类创建一个图标。清单 7.1 是 `TestIcon` 类的一种典型实现方式,它实现了接口 `Icon` 的方法,用于创建一个简单的圆形图标来表示媒体播放器中的 Stop 功能(参看图 7-1)。

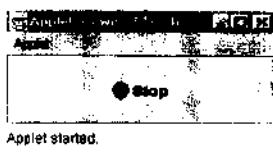


图 7-1 `TestIcon` 类型的 Stop 图标

清单 7.1 接口 `Icon` 的典型实现 (`//TestIcon.java`)

```
// Shows a class that implements the Icon interface

import javax.swing.Icon;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.Component;

class TestIcon implements Icon {
    int width = 20; // set to some default size
    int height = 20; // set to some default size

    // Constructor without any arguments
    public TestIcon() {
    }

    // Constructor to specify the icons width and height
    public TestIcon(int width, int height) {
        this.width = width;
        this.height = height;
    }

    // Paint the picture
    public void paintIcon(Component c,
                          Graphics g,
                          int x,
                          int y) {
        g.setColor(Color.red);
        g.fillOval(x,y,width,height);
        g.setColor(Color.gray);
        g.drawOval(x,y,width,height);
    }

    // To retrieve the icon width
    public int getIconWidth() {
        return width;
    }

    // To retrieve the icon height
    public int getIconHeight() {
        return height;
    }
}
```

```
public int getIconWidth() {
    return width;
}

// To retrieve the icon height
public int getIconHeight() {
    return height;
}
}
```

TestIcon 类实现了 paintIcon ()、getIconWidth () 和 getIconHeight () 这三个方法的完整代码。paintIcon () 方法中用到了图形环境中的两个方法来画一个填充了红色并具有灰色边界的圆。另外两个方法分别返回图标的宽度和高度。构造函数提供了可以创建期望大小的图标的选项。

然后，我们就可以用 TestIcon 类创建图标对象，如下所示：

```
Icon stopIcon = new TestIcon(15,15);
```

StopIcon 对象的类型为 Icon，它现在具有新的宽度和高度。图 7-1 显示了位于 Swing 按钮上的图标。

## 7.2 图像图标

利用 javax.swing 包中的 ImageIcon 类，可以利用图像文件（如 x.gif 或 x.jpg）创建图标。ImageIcon 类是 Icon 接口中已存在的实现形式，它由某个特定的图像生成图标。该类提供了一组构造函数，用来从 Image 类型的对象、图像文件或 URL 创建图像图标。ImageIcon 类是 java.lang.Object 类的一个直接后继。

### 7.2.1 ImageIcon 构造函数

下面的构造函数支持多种选择，以便用给定的 Java 所支持的图像文件创建图标：

```
public ImageIcon()
```

该构造函数创建了一个通用的 ImageIcon 对象，用于以后对特定的图象进行定制。可以通过在已经创建的 ImageIcon 对象上使用 setImage (Image image) 函数来对其进行定制。同一个对象中的 getImage () 方法将返回相关的图像对象：

```
public ImageIcon(Image image)
```

如果已经有一个从图像源创建的 Image 对象，那么就可以使用下述构造函数：

```
public ImageIcon(Image image, String description)
```

除了 description 参数外，这个构造函数类似于前一个构造函数。description 是描述 ImageIcon 对象或图像本身的一个简短的文本信息。这段文本不会在图像上显示，但是可以用 getDescription () 方法检索它，供以后任何时候引用。

```
public ImageIcon(String filename)
```

该构造函数根据图像文件名（如 stop.gif 或 stop.jpg）创建图标。

```
public ImageIcon(String filename, String description)
```

该构造函数类似于上一个构造函数，只是多了一个说明性字符串参数。

```
public ImageIcon(URL url)
```

该构造函数利用位于计算机网络上的图像文件创建图标。

```
public ImageIcon(URL url, String description)
```

该构造函数类似于上一个构造函数，只是多了一个用于描述图像的说明性字符串。

```
public ImageIcon(byte[] pixelArray)
```

该构造函数由 GIF 或 JPEG 图像中的象素数组来创建图像图标。下面给出的构造函数与此类型一样，只不过增加了用于描述图像的说明性字符串。

```
public ImageIcon(byte[] pixelArray, String description)
```

### 7.2.2 创建图像图标

由特定的图像文件创建图标比用自己实现的 Icon 接口要来得简单。要创建图像图标，可以象下面这样使用上述构造函数之一：

```
Icon imageIcon = new ImageIcon("bear.gif")
```

这个图像图标可以在后面的 Swing 组件中使用，它可以显示在组件之上。图 7-2 给出了在 Swing 标签上显示图像图标例子。下一节将讨论 Swing 标签，还将详细讨论怎样在标签上实现图标。

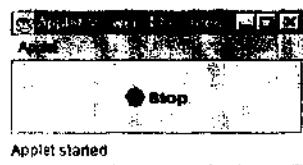


图 7-2 显示在 Swing 标签上的图像图标

### 7.2.3 关闭图像图标

每当组件（例如按钮或复选框）被关闭时，可以在该组件上显示一个变灰了的图标，以此表示其被关闭的状态。被关闭的图标是使用灰色过滤器（gray filter）变灰的。

#### 提示：

通常，每当 Swing 组件被关闭时，该组件负责把图标设置成关闭状态。然而，在有些场合，需要进行将图标设置为关闭的基础工作。

**警告：**

在定制实现创建图标的 Icon 接口时，需要单独地利用 java.awt.Graphics 类的图形环境对象功能，来为被关闭的图标编写代码。

Swing 库提供了图像过滤器类 GrayFilter 来将图像变灰。图像过滤器把图像变为灰度图像，并同时将图像像素加亮。为创建变灰的图像，可以传递图像对象作为参数，来激活 GrayFilter 中的 createDisabledImage() 方法。下面是一段典型的代码：

```
Image image = Toolkit.getDefaultToolkit().getImage("bear.gif");
Image grayImage = GrayFilter.createDisabledImage(image);
Icon grayIcon = new ImageIcon(grayImage);
```

图 7-3 给出了在 Swing 标签上被关闭图标的示例。注意，图标图像仍然反应出了图 7-2 中原来的颜色（图像上的白色和黑色），但图标上有灰影。

```
Image image = Toolkit.getDefaultToolkit().getImage("bear.gif");
Image grayImage = GrayFilter.createDisabledImage(image);
Icon grayedIcon = new ImageIcon(grayImage);
```

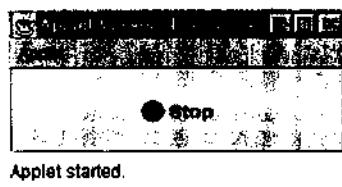


图 7-3 显示在 Swing 标签上的表示关闭的图像图标

#### 7.2.4 Swing 常数

Swing 库中的组件要求用常数安排其文本标签或图标的位置。javax.swing 包中包含有 SwingConstants 接口，该接口提供了一系列很有用的常数。Swing 常数还要控制组件是水平方向的还是垂直方向的。

Swing 常数分为三组，分别表示方框方位、地理方向以及水平或垂直方向。可以按照其用法使用这些常数。例如，常数 TOP 和 BOTTOM 就用来设置文本（针对标签或按钮）相对于其图标的位置。同样地，常数 HORIZONTAL 和 VERTICAL 则用来设置滚动条或滑动条的方向。下面是 Swing 常数的列表：

##### 方框方位常数

```
static int TOP
static int LEFT
static int BOTTOM
static int RIGHT
```

### 地理方向常数

```
static int NORTH
static int NORTH_EAST
static int EAST
static int SOUTH_EAST
static int SOUTH
static int SOUTH_WEST
static int WEST
static int NORTH_WEST
static int CENTER
```

常数 CENTER 是方框方向和地理方向的公用常数。下面两个常数共同用于组件的方向：

```
static int HORIZONTAL
static int VERTICAL
```

## 7.3 Swing 标签 (JLabel)

Swing 标签是一个短小的文本字符串，它可以与一个图标相关联，也可以不与图标相关联。Swing 标签可以用来表示其他组件（如文本域、滑动条等）的用途和目的。标签不能由用户直接修改，只能在程序内改变。

Swing 标签是由 JLabel 类的对象所表示的一个轻量级组件。JLabel 类是 JComponent 类的直接子类，存放在 javax.swing 包中（参看图 7-4）。JLabel 类还实现了 SwingConstants 接口。

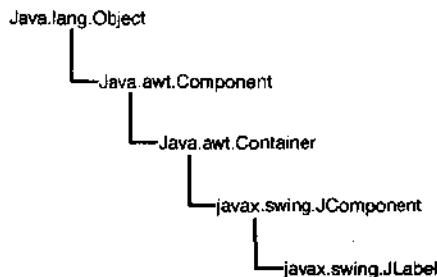


图 7-4 label 的类层次结构

Swing 标签不能获得键盘输入焦点，也不能产生任何类型的事件；它只是简单地显示文本消息和图片。标签上文本或图片的位置是通过指定水平对齐方式和垂直对齐方式来控制的。还可以控制文本相对于图标的位置。

按缺省设置，标签的内容在垂直方向上居中对齐。在仅有文本的标签里，文本消息串左对齐，而在仅有图像的标签里；图像则居中放置。缺省的排列方式为，文本位于图像的右边，两者沿垂直方向对齐。

### 7.3.1 JLabel 构造函数

要在程序中创建标签，可以根据需要选择适当的参数，并调用 JLabel 构造函数。例如，显示仅有文本的标签、仅有图标的标签，或者同时具有文本和图标的标签，等等。Swing 标签对象可以用以下的构造函数之一来创建：

```
public JLabel();
```

这个构造函数创建了一个不带图标和标签字符串的 JLabel 对象。标签字符串和图标可以用类中的 public void setIcon (Icon icon) 和 public void setText (String text) 这两个方法进行设置。

```
public JLabel(Icon icon);
```

该构造函数创建一个具有特定图标的 JLabel 对象。

```
public JLabel(Icon icon, int alignment);
```

该构造函数类似于前一个，只是多了一个使标签按水平方向对齐的参数。

```
public JLabel(String text);
```

该构造函数创建一个以特定文本作为其标签的 JLabel 对象。

```
public JLabel(String text, int alignment);
```

除了指定水平对齐方式外，该构造函数与前一个函数相似。

```
public JLabel(String text, Icon icon, int alignment);
```

这个构造函数创建一个具有特定的文本标签、图标以及水平对齐方式的 JLabel 实例。

创建标签之后，可以根据正在使用的布局要求，把标签放在一个容器内。还可以激活设置水平对齐方式和垂直对齐方式的函数，来控制标签的对齐方式：

```
public void setHorizontalAlignment(int alignment)
```

```
public void setVerticalAlignment(int alignment)
```

缺省地，标签被置于标签显示区域顶部的中央位置。纯文本标签沿水平方向左对齐，纯图标标签沿水平方向右对齐。

也可以用下面的方法指定文本相对于图标的位置。

```
public void setHorizontalTextPosition (int textPosition)
```

```
public void setVerticalTextPosition (int textPosition)
```

缺省情况下，文本和图标都是垂直居中的，但文本位于图标的右侧。

public void setIconTextGap (int IcontextGap) 方法支持在标签和图标之间空出以像素为单位的一定数量的间隔。这个参数的缺省值是 14 个象素。

### 7.3.2 JLabel 代码示例

清单 7.2 创建了一个简单的 Swing 小程序。该程序显示了一组带有 Java 图标和文本字

字符串的 Swing 标签 (见图 7-5)。每个标签中的文本字符串和图标都使用 Swing 常数放在了不同的位置上。该程序也演示了 JLabel 类中一些方法的用法。

```
public void setHorizontalTextPosition(int textPositon)
public void setVerticalTextPosition(int textPositon)
```

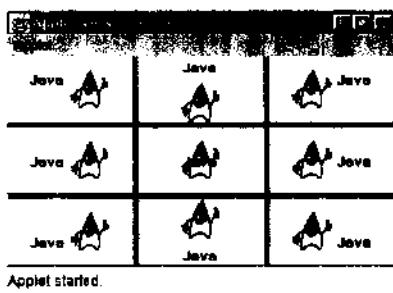


图 7-5 显示带有图标的 Swing 标签

#### 清单 7.2 带有标签、图标和常数的 JLabel (TJLabel.java)

```
// Demonstrates the Swing labels, icons, and constants.

/*
 * <Applet code = TJLabel width = 400 height = 200 >
 * </Applet >
 */

import java.awt.*;
import javax.swing.*;
public class TJLabel extends JApplet {
    Container container = null;
    Icon icon = new ImageIcon("metal.gif");

    public void init() {
        //1. Get a handle on the JApplet's content pane.
        container = getContentPane();

        //2. Create grid layout and set it for the applet and
        //assign a new background color.
        GridLayout grid = new GridLayout(3,3,5,5);
        container.setLayout(grid);
        container.setBackground(Color.gray);

        //3. Invoke the setLabel() method to create and add
        //labels with suitable text and icon positions.
        setLabel(JLabel.LEFT,JLabel.TOP);
        setLabel(JLabel.CENTER,JLabel.TOP);
        setLabel(JLabel.RIGHT,JLabel.TOP);

        setLabel(JLabel.LEFT,JLabel.CENTER);
```

```
setLabel(JLabel.CENTER, JLabel.CENTER);
setLabel(JLabel.RIGHT, JLabel.CENTER);
setLabel(JLabel.LEFT, JLabel.BOTTOM);
setLabel(JLabel.CENTER, JLabel.BOTTOM);
setLabel(JLabel.RIGHT, JLabel.BOTTOM);

}

//4.Create and add labels to the Swing applet.
public void setLabel(int horizPos, int vertPos) {
    JLabel label = new JLabel("Java", JLabel.CENTER);
    label.setIcon(icon); // Display Java Duke on the label
    label.setHorizontalTextPosition(horizPos);
    label.setVerticalTextPosition(vertPos);
    // Assign new background and foreground colors
    label.setBackground(Color.white);
    label.setForeground(Color.black);
    label.setOpaque(true); // label should not be transparent
    container.add(label);
}
}
```

### 代码详析

清单 7.2 的开始处是数据定义部分，如声明一个句柄，在 Swing 小程序容器之上命名的容器，开始处还创建了一个图标对象。代码片段 1 显示了在 init() 方法中，对象引用容器是以 getContentPane() 函数所返回的内容窗格对象进行初始化。

在代码片段 2 中，小程序设置为网格布局，3 行 3 列共 9 个单元，程序背景色为灰色。代码片段 3 分 9 次激活 setLabel() 方法来设置程序中的标签。

setLabel() 方法可以接收相应的参数值，以便设置标签文本相对于图标的水平或垂直位置。例如，第 1 行的第一个标签上文本的位置就是由两个参数定义的：JLabel.LEFT 把文本沿水平方向放在图标的左面，JLabel.TOP 则把文本沿垂直方向放在图标的上面。结果是，文本被放在了图标的左上角。

代码片段 4 显示了 setLabel() 方法中包含的各个步骤。该方法中代码的第一行创建了一个 JLabel 对象，其上同时显示了字符串“Java”。它利用 Swing 常数 JLabel.CENTER 把文本字符串放在了标签中央。在下一条语句中，作为程序数据字段创建的图标对象被附加到了标签上。文本字符串相对于图标的水平和垂直位置利用下面的方法加以控制：

```
label.setHorizontalTextPosition(horizPos)
label.setVerticalTextPosition(vertPos)
```

代码片段 4 还给出了将标签的背景色和前景色分别设置为特定的白色和黑色的语句。当参数值为 true 时，SetOpaque() 方法将每个标签对象设置成不透明的。这使得标签能够反映出它被设置的颜色；否则，标签将会显示出其背景色。本程序的输出结果如图 7-5 所示。

**说明:**

缺省地, Swing 标签是透明的。因此, 先激活 `setOpaque()` 方法, 并把 `true` 作为传递参数, 可使标签成为不透明的。

## 7.4 边界

边界其实就是环绕 Swing 组件四周的空白区。AWT 中所支持的镶边 (Inset) 的作用与此类似。但是, Swing 中的边界比 AWT 中的镶边更为复杂且丰富多彩。Swing 库提供的各种不同的边界可使用户界面的外观各具特色。

Swing 边界类的背景仍使用 AWT 中的镶边对象。为给组件赋予某种特定的边界, 可以创建一个边界对象, 并在组件上激活下面的方法把它赋给组件:

```
Public void setBorder(Border border)
```

对于组件对象, 这个方法会覆盖组件上任何已经直接设定了的边界。

为了创建不同类型的边界, Swing 库支持一个叫做 `Border` 的设计级接口 (design level interface) 和一个叫做 `AbstractBorder` 的抽象类。Swing 库中还有几个代表公用边界的具体的类。`javax.swing` 包里的 `BorderFactory` 类包含了一些方便的方法, 它们可以创建边界对象。还可使用这些方法创建边界对象, 而不必由边界类来创建。

### 7.4.1 Border 接口

`Border` 接口是一个设计级的接口, 它可以描述边界对象。因此, 代表边界的类就会实现这个接口。该接口已经存放在 `javax.swing.border` 包中, 并且还包含需要加以实现的以下三个方法:

```
Public void paintBorder(Component c, Graphics g,
    Int x, int y,
    Int width, int height)
```

```
Public Insets getBorderInsets(Component c)
Public boolean isBorderOpaque()
```

`paintBorder` 方法在指定的 `x`, `y` 坐标位置画出某个特定组件 `c` 的边界。参数 `width` 和 `height` 分别代表所画边界的宽度和高度。`getBorderInsets()` 方法返回组件 `c` 的边界的特定镶边。`isBorderOpaque()` 方法返回 `true` 或 `false`, 用来表示边界应该被设置成不透明的还是透明的。如果它返回的状态是 `false`, 那么容器的背景色就是可见的了。

### 7.4.2 AbstractBorder 类

除 `Border` 接口之外, `javax.swing.border` 包中还包含了一个叫做 `AbstractBorder` 的抽象类。这个类实现了 `Border` 接口, 并且它还代表一个大小为零的空边界。这是一个便于使用的基类, 其他的类都可从它派生。`AbstractBorder` 类包含一个被重载了的方法 `getInterior-`

`Rectangle()`，该方法补充了 `Border` 接口中实现的方法。有关此类更详细的内容，请参看附录 A “JFC Swing 快速参考”。

## 7.5 边界的类型

`javax.swing.border` 包中包含一组实用的边界类。可以简单地实例化某个特定的边界类，或者利用相关的工厂方法（factory method）创建边界对象。下面几节给出了可用的 Swing 边界（参看图 7-6）。

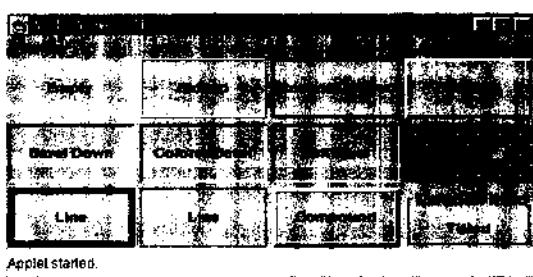


图 7-6 显示带有边界的标签

### 7.5.1 空边界 (Empty Border)

空边界所留出的空白是透明的，它上面没有绘制任何内容（参看图 7-6）。这种类型的边界对象由 `EmptyBorder` 类表示，它们能够完全替代 AWT 中的镶边类。这个类需要指定 `top`、`left`、`bottom` 和 `right` 这样的字段。也可以用 `insets` 对象创建空的边界。下面是空边界构造函数：

```
public EmptyBorder(int top, int left, int bottom, int right)  
public EmptyBorderInsets insets
```

### 7.5.2 蚀刻边界 (Etched Border)

蚀刻边界能够在组件周围产生“凹进 - 蚀刻”或者“凸起 - 蚀刻”的效果（参看图 7-6）。蚀刻边界由 `EtchedBorder` 类表示。可以用 `LOWERED` 或 `RAISED` 常数指定蚀刻边界的类型。蚀刻边界的缺省类型为“凹进 - 蚀刻”类型。

蚀刻边界可以与指定的高亮度色和阴影色相关联。如果没有指定这些属性的颜色，那么边界就会继承组件的背景色。可以使用以下构造函数之一创建蚀刻边界，这些构造函数需要蚀刻边界类型，并以高亮度色和阴影色作为参数值。

```
public EtchedBorder()  
public EtchedBorder(int etchType)  
public EtchedBorder(Color highlight, Color shadow)  
public EtchedBorder(int etchType,  
                    Color highlight, Color shadow)
```

### 7.5.3 斜切边界 (Bevel Border)

斜切边界的表面或者是凸起的，或者是凹陷的（如图 7-6 所示）。斜切边界由 BevelBorder 类表示，可以用 LOWERED 或 RAISED 常数使斜切边界成为凸起的，或者凹陷的。可以指定斜切边界内侧或外侧的高亮度色和阴影色。以下是创建斜切边界的构造函数：

```
public BevelBorder(int bevelType)
public BevelBorder(int bevelType,
                   Color highlight,
                   Color shadow)
public BevelBorder(int bevelType,
                   Color highlightOuter,
                   Color highlightInner,
                   Color shadowOuter,
                   Color shadowInner)
```

### 7.5.4 软斜切边界 (Soft Bevel Border)

软斜切边界是具有圆角的斜切边界。此种类型的边界由 SoftBevelBorder 类表示。SoftBevelBorder 类是 BevelBorder 类的扩展类，因此可以创建凸起的或者凹陷的的软斜切边界。软斜切边界可由下列构造函数之一创建：

```
public SoftBevelBorder(int bevelType)
public SoftBevelBorder(int bevelType,
                      Color highlight,
                      Color shadow)
public SoftBevelBorder(int bevelType,
                      Color highlightOuter,
                      Color highlightInner,
                      Color shadowOuter,
                      Color shadowInner)
```

### 7.5.5 粗糙边界 (Matte Border)

粗糙边界显示一个粗糙模式。粗糙模式由纯色或平铺的图标（参看图 7-6）创建。粗糙边界类为 MatteBorder，它扩展了 EmptyBorder 类。纯色的粗糙边界可以用指定的颜色来创建。类似地，平铺图标的粗糙边界则可利用一个显示粗糙图片的图标来创建。可以用下面的构造函数创建粗糙边界：

```
public MatteBorder(int top, int left,
                   int bottom, int right,
                   Color color)
public MatteBorder(int top, int left,
                   int bottom, int right,
```

```
Icon tileIcon)  
public MatteBorder(Icon tileIcon)
```

### 7.5.6 线条边界 (Line Border)

线条边界是由具有一定粗细和颜色的、环绕组件的简单直线构成的。线条边界对象由 `LineBorder` 类表示，该类是 `AbstractBorder` 类的扩展类。可以用以下的静态方法创建线条边界：

```
static Border createBlackLineBorder()  
static Border createGrayLineBorder()
```

这两个方法可以分别创建白色或灰色的线径为 1 的线条边界。也可以用以下的构造函数创建线条边界：

```
public LineBorder(Color color)  
public LineBorder(Color color, int thickness)
```

需要指定线条的颜色。但线条的粗细是可选的。如果调用构造函数时没有用 `thickness` 参数，那么缺省的线条粗细为 1。

### 7.5.7 复合边界 (Compound Border)

复合边界是由两种边界对象复合而成的。可以将里层边界嵌在外层边界的镶边中。复合边界用 `Border` 类表示，该类是 `AbstractBorder` 类的扩展类。可以用以下的构造函数创建复合边界：

```
public CompoundBorder()  
public CompoundBorder(Border outsideBorder,  
                      Border insideBorder)
```

### 7.5.8 带标题的边界 (Titled Border)

带标题的边界是由标题字符串与前面所讨论的任意边界组合而成的一种边界。可以按照某种对齐方式控制标题的位置，还可以指定标题所用的字体和颜色。标题的这些属性都是 `TitledBorder` 类中的静态字段，它们代表带有标题的边界。附录 A 中给出了这些字段的参考用法。

要创建带标题的边界，需要指定标题和边界。如果没有指定边界，则缺省使用蚀刻边界。下面是创建带标题边界的构造函数：

```
public TitledBorder(String title)  
public TitledBorder(Border border, String title)  
public TitledBorder(Border border, String title,  
                     int titleJustification,  
                     int titlePosition)  
public TitledBorder(Border border, String title,
```

```
    int titleJustification,
    int titlePosition,Font titleFont)
public TitledBorder(Border border,String title,
    int titleJustification,
    int titlePosition,
    Font titleFont,Color titleColor)
```

### 7.5.9 边界代码示例

清单 7.3 给出的示例程序中演示了所讨论过的各种边界类型的用法。程序实现的小程序中显示了一组带有边界的标签。每个标签都带有一个特殊类型的边界。

清单 7.3 带有各种边界类型的小程序 (*TBorder1.java*)

```
/*
 * < Applet code = TBorder1 width = 400 height = 200 >
 * </Applet>
 */

import javax.swing.border.*;
import javax.swing.*;
import java.awt.*;

public class TBorder1 extends JApplet{
    public void init() {
        // 1.Create an object of panel to contain a label
        // and add it to the applet's content pane.
        LabelPanel panel = new LabelPanel();
        getContentPane().add(panel);
    }
}

// 2.Panel that contains labels with borders.
class LabelPanel extends JPanel {
    Border border;
    JLabel label;

    public LabelPanel() {
        // 3.Assign grid layout (with 3 rows,4 columns, and
        // 5 pixel horizontal and vertical spacing) to panel.
        setLayout(new GridLayout(3,4,5,5));

        // 4.Create a label with an empty border.
        label = new JLabel("Empty",JLabel.CENTER);
        label.setOpaque(true);

        // Create an empty border object with the specified insets:
        // top = 1; left = 1; bottom = 1; right = 1.
        border = new EmptyBorder(1,1,1,1);
```

```
// Assign the border to the label.  
label.setBorder(border);  
add(label);  
  
// 5.Create a label with an etched border.  
label = new JLabel("Etched",JLabel.CENTER);  
label.setOpaque(true);  
// Create a raised and etched border.  
border = new EtchedBorder(EtchedBorder.RAISED);  
label.setBorder(border);  
add(label);  
  
// 6.Create a label with an etched border with color.  
label = new JLabel("Etched&Colored",JLabel.CENTER);  
label.setOpaque(true);  
border = new EtchedBorder(EtchedBorder.LOWERED, // Lowered and etched type  
Color.red,Color.blue); // Hightlight and shadow colors  
label.setBorder(border);  
add(label);  
  
// 7.Create a label with a bevel border.  
label = new JLabel("Bevel Up",JLabel.CENTER);  
label.setOpaque(true);  
border = new BevelBorder(BevelBorder.RAISED); // Raised bevel type  
label.setBorder(border);  
add(label);  
  
// 8.Create a label with a lowered bevel border.  
label = new JLabel("Bevel Down",JLabel.CENTER);  
label.setOpaque(true);  
border = new BevelBorder(BevelBorder.LOWERED); // Lowered bevel type  
label.setBorder(border);  
add(label);  
  
// 9.Create a label with a raised bevel border.  
label = new JLabel("ColoredBevel",JLabel.CENTER);  
label.setOpaque(true);  
border = new BevelBorder(BevelBorder.RAISED, // Raised bevel type  
Color.gray,Color.yellow); // Hightlight and shadow colors  
label.setBorder(border);  
add(label);  
  
// 10.Create a label with a soft bevel border.  
label = new JLabel("SoftBevel",JLabel.CENTER);  
label.setOpaque(true);  
// Create a lowered bevel type border object with softened corners  
// without pointings
```

```
border = new SoftBevelBorder(BevelBorder.LOWERED);
label.setBorder(border);
add(label);

// 11.Create a label with a matte border.
label = new JLabel("Matte",JLabel.CENTER);
label.setOpaque(true);
Icon icon = new ImageIcon("cube.gif");
// Create a matte border object with the specified insets of
// top = 20, left = 20, bottom = 20; right' = 20 and matte icon
border = new MatteBorder(20,20,20,20,icon );
label.setBorder(border);
add(label);

// 12.Create a label with a red line border.
label = new JLabel("Line",JLabel.CENTER);
label.setOpaque(true);
border = new LineBorder(Color.red,// Line color = red
5); // line thickness = 5.
label.setBorder(border);
add(label);

// 13.Create a label with a gray line border.
label = new JLabel("Line",JLabel.CENTER);
label.setOpaque(true);
// Create a line border with gray color and thickness = 1.
border = LineBorder.createGrayLineBorder();
label.setBorder(border);
add(label);

// 14.Create a label with a compound border.
label = new JLabel("Compound",JLabel.CENTER);
label.setOpaque(true);
// Create a compound border with a raised bevel border and a raised
// etched border
border = new CompoundBorder(
    new BevelBorder(BevelBorder.RAISED),
    new EtchedBorder(EtchedBorder.RAISED));
label.setBorder(border);
add(label);

// 15.Create a label with a titled border with specified
// border.
label = new JLabel("Titled",JLabel.CENTER);
label.setOpaque(true);
border = new TitledBorder(new LineBorder(Color.red),
```

```

        "Lined&Titled", // Display title.
        TitledBorder.DEFAULT_JUSTIFICATION,
        // Title justification.
        TitledBorder.CENTER, // Title location.
        new Font("Sans", Font.BOLD, 16), // Title font.
        Color.blue); // Title color.

    label.setBorder(border);
    add(label);

}
}

```

### 代码详析

小程序 TBorder1 用到了一个窗格，其中显示了一组 Swing 标签，每个标签都装饰了一种类型的边界。程序里 init() 方法中的代码片段 1 创建了窗格的一个实例。代码片段 2 中声明了一个标签和一个边界作为窗格的数据字段。代码片段 3 将窗格赋予网格布局。

代码片段 4~15 创建了具有相应边界的标签，并把它们加到程序中。注意，这里所创建的各种边界对象是：空边界、蚀刻边界、有色蚀刻边界、凸起斜切边界、凹进斜切边界、软斜切边界、粗糙边界、线条边界、复合边界以及带标题的边界。程序的输出结果如图 7-6 所示。

## 7.6 自定义边界

如果 Swing 库中的边界不能满足要求，可以利用 Border 接口或 AbstractBorder 类创建自定义（或称定制）边界。

要创建定制边界，需生成自定义边界类，这个类定义了 Border 接口内的方法。Border 接口已经在前面的章节中讨论过了。清单 7.4 中使用了自定义边界。其中所创建的定制边界是由一组具有特定颜色的 3D 矩形组成的（参看图 7-7）。

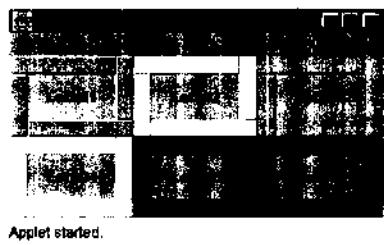


图 7-7 带有定制边界的小程序

### 清单 7.4 定制边界 (TBorder2.java)

```

/*
* <Applet code = TBorder2 width = 300 height = 150 >
* </Applet >

```

```
* /  
import javax.swing.*;  
import java.awt.*;  
import javax.swing.border.*;  
public class TBorder2 extends JApplet {  
    public void init() {  
  
        // 1. Get a handle on the applet's content pane.  
        // and set the grid layout.  
        Container container = this.getContentPane();  
        container.setLayout(new GridLayout(2,3)); // 2 rows and 3 columns.  
  
        // 2. Add labels to the applet with decorating  
        // borders created from the class SimpleBorder.  
  
        // 2(a) Create an array of favorite colors.  
        Color[] colors = {Color.lightGray,Color.yellow,  
                          Color.green,Color.white,  
                          Color.blue,Color.red};  
  
        // 2(b) Create labels and add them to the container.  
        for (int i=0; i<6; i) { // Total number of labels = 6.  
            JLabel label = new JLabel("Label" + (i+1),JLabel.CENTER);  
            label.setOpaque(true);  
            container.add(label);  
            label.setBorder(new SimpleBorder(15,// Top inset.  
                                         15,// Left inset.  
                                         15,// Right inset.  
                                         15,// Bottom inset.  
                                         colors[i])); // border color  
        }  
    }  
  
    // 3. Define the custom border class called SimpleBorder.  
    class SimpleBorder implements Border {  
        int top;  
        int left;  
        int bottom;  
        int right;  
        Color color = null;  
  
        public SimpleBorder(int top, int left, int bottom,  
                           int right,Color color) {  
            this.top = top;  
            this.left = left;  
            this.bottom = bottom;
```

```
        this.right = right;
        this.color = color;
    }

    public void paintBorder(Component c, // Component that will contain the border.
                           Graphics g, // Graphics context.
                           int x, int y, // x and y coordinates of the painted border.
                           int width, int height) { // Border width and height.

        // Create insets around the component to draw the border.
        Insets insets = getBorderInsets(c);

        // Set the border color.
        if (color != null)
            g.setColor(color);

        // Prepare the border by using 3D Rectangles returned by the method.
        // g.fillRect(). This method takes the argument values as following:
        g.fillRect(0, // x - coordinate
                  0, // y - coordinate
                  width - insets.right, // width
                  insets.top, // height.
                  true); // Rectangle appears to be raised.

        // The following methods also work with the arguments as shown in the
        // previous method call.
        g.fillRect(0, insets.top, insets.left,
                  height - insets.top, true);
        g.fillRect(insets.left, height - insets.bottom,
                  width - insets.left, insets.bottom, true);
        g.fillRect(width - insets.right, 0, insets.right,
                  height - insets.bottom, true);
    }

    public Insets getBorderInsets(Component c) {
        return new Insets(top, left, bottom, right);
    }

    public boolean isBorderOpaque() {
        return true;
    }
}
```

### 代码详析

TBorder1 类是一个 Swing 小程序，其中显示了六个带有自定义边界的标签。在程序的 init () 方法中，代码片段 1 取得了小程序内容窗格的句柄，并且将一个二行三列的网格布局赋予该窗格。代码片段 2 中定义了一个颜色数组，并将标签加到了布局上。每个标签都采

用了 SimpleBorder 创建的边界。

代码片段 3 定义了 SimpleBorder 类，其中还实现了 Border 接口。这个类包含了插进参数 (inset argument) top、left、bottom 和 right 作为其数据字段。可以在构造函数中初始化这些字段。

如你所见，在 SimpleBorder 类内定义了 Border 接口。paintBorder () 方法又利用 getBorderInsets () 方法创建了 insets 对象。getBorderInsets () 返回一个 insets 对象，该对象带有由构造函数提供的四个值 top、left、bottom 和 right。

paintBorder () 方法中的图形环境对象随后用于设置需创建的边界颜色。现在，可以根据代码中给定的坐标值用 fill3DRect () 方法创建一个 3D 矩形了。这样就创建了一个环绕标签的矩形。另一个方法 isBorderOpaque () 设置为 true，它使得边界按照所给的颜色显示。程序的输出结果如前一小节的图 7-7 所示。

# 第 8 章 按钮和复选框

Swing 库中提供了简单按钮类、切换按钮类和抽象按钮类，后者封装了按钮类的一些公共功能。Swing 库还支持针对各种不同外观形式的按钮，如 Java/Metallic、Motif 和 Windows。

复选框和单选按钮是在功能上平行于 Swing 按钮的组件，然而，它们的图示外观却完全不同。复选框常用来在用户界面上打开或关闭某个属性。复选框和单选按钮是 Swing 切换按钮类的子类，切换按钮类中封装了响应重要事件的功能。本章将详细讨论上述每一种组件，并通过示例给出相关 API 方法的用法。

## 8.1 AbstractButton 类

一般地，每个 GUI 组件都具有一定的状态和功能。对于常规按钮、复选框和单选按钮来说，它们的公共功能都包含在一个叫做 AbstractButton 的抽象类中。抽象类 AbstractButton 中封装了丰富的方法。AbstractButton 是 JComponent 类的直接子类，它实现了接口 Swing-Constants 和 ItemSelectable。

**提示：**

如果要创建任何类似于按钮的组件，都可以选择将该组件作为 AbstractButton（不是 JComponent）的子类，从而能够利用它的功能。

本章将讨论 Swing 按钮（由 JButton 类和 JToggleButton 类表示），以及 JCheckBox 和 JRadioButton 类型的复选框。AbstractButton 是这些组件的父类，其类层次结构如下面的图 8-1 所示：

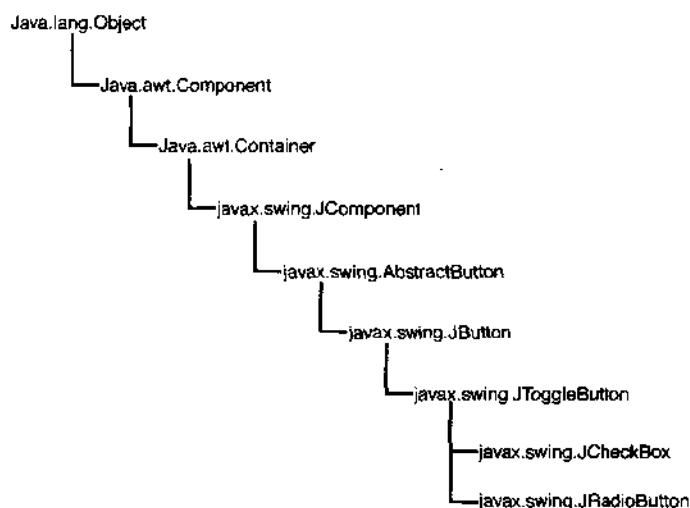


图 8-1 AbstractButton 类及其子类

### 8.1.1 剖析 AbstractButton

`AbstractButton` 类封装了大量的公共方法，这些方法是由其子类激活的，如 `JButton`、`JToggleButton`、`JCheckBox` 以及 `JRadioButton`。此外，因为 `SwingConstants` 接口是由 `AbstractButton` 类实现的，所以能够在继承组件内使用常数。可以找到用来设置或取得各种按钮状态并对它们进行注册的方法。下面列出的是常用的主要方法：

```
Public void addActionListener(ActionListener listener)
Public void addChangeListener(ChangeListener listener)
Public void addItemListener(ItemListener listener)
```

`Swing` 按钮 `JButton` 和 `JToggleButton` 会激活由相关的动作监听类所接收的事件。使用 `addActionListener()` 方法，可以对激活该方法的源组件注册动作监听类。而 `addChangeListener()` 方法则帮助通知相应的监听类去通报组件源的状态变化。`addItemListener()` 方法由 `JCheckbox` 和 `JRadioButton` 共同用于，在选择某一条目或取消选择某一条目时注册该条目的监听类。

除可以显示文本标签外，`Swing` 按钮还支持将图标显示在按钮上。针对不同的按钮环境，比如，选中按钮或被取消选择，被禁用或被禁用的同时还被选择，鼠标指针在按钮上移动或移动的同时选中按钮，此时图标都能够做出相应的反应。以下是提供这种支持的方法：

```
public void setText(String textLabel)
public void setIcon(Icon icon)
public void setPressedIcon(Icon icon)
public void setRolloverIcon(Icon icon)
public void setSelectedIcon(Icon icon)
public void setDisabledIcon(Icon icon)
public void setRolloverSelectedIcon(Icon icon)
public void setDisabledSelectedIcon(Icon icon)
```

以上清单中的每个“set”方法都有与之对应的“get”方法，后者返回组件的相应属性。例如，`setText()` 方法有一个对应的 `getText()` 方法，后者返回一个具有 `String` 类型的文本。

为了把图标或文本按水平方向或垂直方向对齐，可以使用下面的方法，下面的方法取 `Swing` 常数（如 `TOP`、`LEFT`、`BOTTOM`、`RIGHT` 以及 `CENTER`）作为参数确定其位置：

```
setHorizontalAllignment(int swingConstant)
setVerticalAllignment(int swingConstant)
```

下面的方法支持将文本放置在相对于图标沿水平方向或垂直方向的位置上：

```
setHorizontalTextPosition(int swingConstant)
setVerticalTextPosition(int swingConstant)
```

以下更多的“set”方法可用于设置组件的属性。例如，启用或关闭按钮，表示快捷键，画边界或焦点框，等等。

```
public void setEnabled(boolean value)
public void setKeyAccelerator(int key)
public void setBorderPainted(boolean value)
public void setFocusPainted(boolean value)
public void setMarginInsets(insets insets)
public void setActionCommand(String command)
```

以下方法根据按钮状态返回布尔值 true 或 false，比如按钮是否被选择，边界是否已经画出，焦点框否已经画出，以及是否允许当鼠标指针在按钮上移动或移动的同时还可选中按钮（rollover enabled），等等。

```
boolean isSelected()
boolean isBorderPainted()
boolean isFocusPainted()
boolean isRolloverEnabled()
```

附录 A “JFC Swing 快速参考” 给出了这些方法的完整列表，它覆盖了来自不同 Swing 包中的大量的类。本章下一节将介绍如何创建在 GUI 布局中显示的 AbstractButton 类的子类。

## 8.2 按钮

Swing 按钮由 JButton 类的对象表示，并且每个按钮基本上都是一个“按动式”（push-type）按钮的实现。与 AWT 按钮不同，Swing 按钮既能显示文本标签，也能显示图标，还可以利用它所支持的方法，针对按钮的不同状态显示不同的图标（例如，当按钮被按下或被禁用时）。

JButton 类是 AbstractButton 类的直接子类。事实上，JButton 类大部分功能都是从 AbstractButton 继承得到的。可以激活 AbstractButton 类中的大多数方法，来设置诸如缺省图像、文本对齐方式以及事件监听对象等等。

### 8.2.1 JButton 构造函数

Swing 按钮可用以下构造函数之一加以创建：

```
public JButton()
public JButton(String text)
public JButton(Icon icon)
public JButton(String text, Icon icon)
```

第一个构造函数创建一个按钮，该按钮不带任何文本组件对象或图标组件对象，这些文本或图标可以在以后设置为指定的文本或感兴趣的图标。后面两个构造函数所带的 text 和 icon 参数，指定了构造函数所创建按钮上要显示的文本和图标。最后一个构造函数可用于在创建按钮时，需要同时指定文本对象和按钮对象的场合。

### 8.2.2 动作事件和监听对象

按钮对象能够激活 ActionEvent 类型的事件。这些事件可以由相应的监听对象捕获，进而做必要的处理。

动作监听类实现了 ActionListener 接口并提供了 actionPerformed() 方法的代码。这是 ActionListener 接口中唯一的方法。为了能够接收由按钮对象激活的事件，监听对象必须同事件源一起注册。可以通过激活源对象的 AbstractButton 类中的 addActionListener() 方法，将监听对象与按钮源连接起来。addActionListener() 方法取按钮对象作为其参数值。

每当有源按钮操作时，所发生动作就会激活 actionPerformed 方法。因此，发生按钮操作时所执行的代码位于这个方法之内。actionPerformed() 方法取 ActionEvent 类对象作为其参数值。ActionEvent 类中包含在处理 actionPerformed() 方法内的代码时获取有用信息的方法。这些方法如下：

```
public String getActionCommand()  
public int getModifiers()  
public String paramString()
```

当事件源创建一个动作事件时，该事件源可以包括字符串类型的动作命令，这个动作命令中给出了有关该事件源的一些详细信息。getActionCommand() 方法可以获取动作事件的动作命令。下面的按钮示例程序显示了怎样使用 setActionCommand() 和 getActionCommand() 方法。

getModifiers() 方法可以获取创建动作事件时更新键的状态。paramString() 方法生成一个表示动作事件状态的字符串。

### 8.2.3 JButton 代码示例

清单 8.1 演示了按钮的用法。这个程序是一个 Swing 小程序，它给出了 JButton 不同的“可插入外观”的视图，其中使用了在 JButton 对象进行操作的 API（参看图 8-2）。该程序还演示了通过按钮对象激活的事件处理过程。当单击按钮时，小程序的外观就会变成某种特定的样子。

清单 8.1 不同的可插入外观的 JButton (TJButton.java)

```
// Demonstrates the Swing Buttons...  
/*  
<Applet code = TJButton width = 400 height = 65>  
</Applet>  
*/  
import java.awt.*;  
import javax.swing.*;  
import java.awt.event.*;  
  
public class TJButton extends JApplet {  
    //1. Declare the Swing buttons.
```

```
JButton button1,button2,button3;
//2.Icons for the buttons 1,2, and 3.
Icon metalIcon = new ImageIcon("metal.gif");
Icon motifIcon = new ImageIcon("motif.gif");
Icon windowsIcon = new ImageIcon("windows.gif");
//3.String forms of different look - and - feel.
String metallicLF =
"javax.swing.plaf.metal.MetalLookAndFeel";
String motifLF =
"com.sun.java.swing.plaf.motif.MotifLookAndFeel";
String windowsLF =
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
public void init() {
    //4.Creating grid layout and setting it for the applet.
    GridLayout grid = new GridLayout(1,3,5,5);
    this.getContentPane().setLayout(grid);
    this.getContentPane().setBackground(Color.lightGray);
    //5.Creating and adding Swing button1.
    button1 = new JButton("Java L&F",metalIcon);
    button1.setVerticalTextPosition(SwingConstants.BOTTOM);
    button1.setHorizontalTextPosition(SwingConstants.CENTER);
    button1.setMnemonic('J');
    button1.setEnabled(false);
    button1.setActionCommand(metallicLF);
    CustomListener cl = new CustomListener(this);
    button1.addActionListener(cl);
    this.getContentPane().add(button1);
    //6.Creating and adding Swing button2.
    button2 = new JButton("Motif L&F",motifIcon);
    button2.setVerticalTextPosition(SwingConstants.BOTTOM);
    button2.setHorizontalTextPosition(SwingConstants.CENTER);
    button2.setMnemonic('M');
    button2.setActionCommand(motifLF);
    button2.addActionListener(cl);
    this.getContentPane().add(button2);
    //7.Creating and adding Swing button3.
    button3 = new JButton("Windows L&F",windowsIcon);
    button3.setVerticalTextPosition(SwingConstants.BOTTOM);
    button3.setHorizontalTextPosition(SwingConstants.CENTER);
    button3.setMnemonic('W');
    button3.setActionCommand(windowsLF);
```

```
button3.addActionListener(cl);
this.getContentPane().add(button3);
}

//8. Inner class that supports event handling.
class CustomListener implements ActionListener {
    TJButton testApplet = null;
    public CustomListener(TJButton testApplet) {
        this.testApplet = testApplet;
    }
    public void actionPerformed(ActionEvent evt) {
        //9. For pluggable look-and-feel.
        String LF = evt.getActionCommand();
        try {
            UIManager.setLookAndFeel(LF);
            SwingUtilities.updateComponentTreeUI(testApplet);
            JButton button = (JButton) evt.getSource();
            button.setEnabled(false);
            updateStatus(button);
        } catch (Exception e) {
            System.err.println("Look&Feel not set for " + LF + " !");
        }
    }
}

//10. Updates the status of each button (Enabled/Disabled).
public void updateStatus(JButton button) {
    if (button == button1) {
        button2.setEnabled(true);
        button3.setEnabled(true);
    }
    else if (button == button2) {
        button1.setEnabled(true);
        button3.setEnabled(true);
    }
    else if (button == button3) {
        button1.setEnabled(true);
        button2.setEnabled(true);
    }
}
}
```

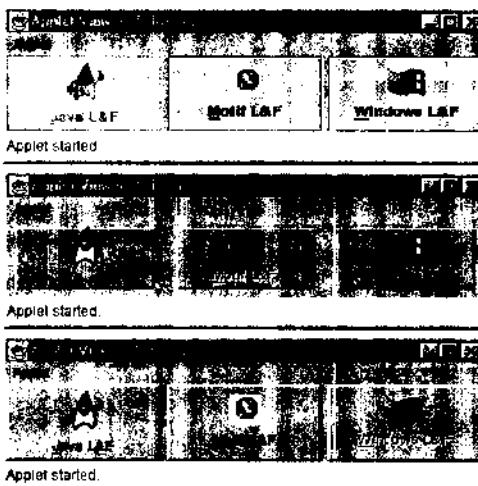


图 8-2 具有不同的可插入外观的 JButton 小程序

### 代码详析

清单 8.1 中包含了 JButton 类型的三个不同的对象，在代码片段 1 中，它们声明为程序的数据成员。代码片段 2 创建了三个 Swing 按钮中要用到的图标。代码片段 3 给出了一列字符串，如 metallicLF，motifLF 以及 windowsLF，这些字符串分别代表了特定的可插入外观。这些信息将在后面创建程序所需的外观时使用。

接下来，程序使用 init() 方法开始初始化过程。代码片段 4 创建了 GridLayout 类型的布局，并把它设置为 Swing 小程序的内容窗格。必须注意，布局被设置成了内容窗格对象，并且它需要由 getContentPane() 方法返回，而不是从 Swing 直接得到。

代码片段 5 创建了 JButton 类型的对象，它上面有一个标签“Java L&F”和一个 metalIcon 图标。metalIcon 已经被创建并作为小程序的数据成员。随后两行程序把文本标签放在了按钮表面。也就是说，利用 setVerticalTextPosition() 方法把文本标签放在了按钮沿垂直方向的底部，利用 setHorizontalTextPosition() 方法把它放在水平方向上的中间位置。

SetMnemonic() 方法通过给特定字符设置下划线来支持加速键（这里，“J”设置为加速键）。当小程序显示出来时，其外观为 Java/Metallic 样式。因此，将 setEnabled() 方法设置成 false 时，相应的按钮 button1 就被关闭了。代码的下一行中，setActionCommand() 方法根据期望的外观样式、用命令字符串形式设置动作命令。正如后面将要讨论的，setActionCommand() 方法可以在任何事件对象上进行操作。

addActionListener() 方法注册代码片段 5 中的按钮对象的监听类。这个按钮监听类叫做 CustomListener，它接收由事件源所激活的事件，button1.this.getContentPane().add(button1) 语句将 button1 加到了 Swing 小程序的容器中。代码片段 6 和代码片段 7 对另外两个按钮 button2 和 button3 的操作方式与 button1 的操作类似。这两个按钮的作用是：在程序执行过程中，把程序外观改成 Motif 类型和 Windows 类型。

代码片段 8 给出的内部类可以处理由按钮对象激活的动作事件。监听类实现了 Action-

Listener 接口，并提供了 actionPerformed() 方法的代码。actionPerformed() 是 ActionListener 接口的成员函数，每当 JButton 类对象之一激活 ActionEvent 对象时就会执行该函数。

代码片段 9 给出 actionPerformed() 方法的执行代码。getActionCommand() 方法返回所期望外观的字符串形式。在 ActionEvent 对象上激活 getActionCommand() 方法。注意，外观的字符串形式已经在前面由 setActionCommand() 设置好了，而 setActionCommand() 方法又是在一个特定的按钮对象上被激活的。UIManager 类控制用户界面的外观。SetLookAndFeel() 方法是把一种外观赋予 JButton 及其组件的句柄。字符串值 LF 用定义的形式显示。必须将 SetLookAndFeel() 方法包含在 try 语句内，以便捕获一般的 Exception 类型的异常，或者如下所示的一系列异常情况：

```
java.lang.ClassNotFoundException  
java.lang.InstantiationException  
java.lang.IllegalAccessException  
java.lang.UnsupportedLookAndFeelException
```

在上述异常情况中，如果没有找到外观类，程序就会抛出第一种异常。如果因为该类是一个接口或者是一个抽象类而不能创建类的实例，程序就会抛出第二种异常。如果该类不是 public（公用的）或者它位于不同的包内，则程序就会抛出第三种异常。如果不支持选择的外观类型，则程序就会抛出最后一种异常。

可以调用 SwingUtilities 类中的 updateComponentTreeUI() 方法把层次式组件中每个组件的外观改换为一种新的形式。这个方法取 JButton 小程序的对象作为其参数值。

当程序运行时，其视图的外观类型为 Java/Metallic。每当单击带有“Motif L&F”和“Windows L&F”按钮时，程序都要变成相应的外观类型（请参看程序的输出结果图）。其中，激活的按钮也同时被关闭了。

#### 说明：

当按钮被关闭后，它的图标和文本也会由其内部变灰过滤函数的执行而自动变成灰色。

## 8.3 切换按钮

切换按钮与前面所讨论的普通按钮类似，但有一点不同。当切换按钮被按下时，它的外观会保持被按下的状态，一直到它被重新按下以后再弹起。

切换按钮由 JToggleButton 类表示，该类实际上只封装了一个双态按钮。JToggleButton 是 AbstractButton 的子类。可以在 JToggleButton 对象上激活 AbstractButton 中的任何可用方法。也可以使用 DefaultButtonModel 和内部类 ToggleButtonModel 所支持的功能。DefaultButtonModel 类包含了一个按钮模型数据的实现。每当进行切换按钮的操作时，它就会象普通按钮那样激活动作事件。

### 8.3.1 JToggleButton 构造函数

切换按钮对象可用本节将要讨论的 8 种构造函数之一来创建：

```
public JToggleButton()
```

```
public JToggleButton(String textlabel)
public JToggleButton(Icon icon)
public JToggleButton(Boolean pressed)
```

第一个构造函数创建一个初始未被选择的切换按钮，该按钮不带任何文本或图标。第二个构造函数的作用与第一个构造函数相似，只不过切换按钮上带有一个文本标签。第三个构造函数创建一个未被选择并且上面带有一个图标的切换按钮。第四个构造函数创建一个具有被按下（true）或弹起（false）状态的切换按钮。

以下的构造函数是上述构造函数中参数不同组合的结果：

```
public JToggleButton(String textlabel, Boolean pressed)
public JToggleButton(Icon icon, Boolean pressed)
public JToggleButton(String textlabel, Icon icon)
```

为了创建具有指定文本、图像并可显示按下状态（true 或者 false）的切换按钮，可以使用下面这个构造函数：

```
public JToggleButton(String textlabel, Icon icon, Boolean pressed)
```

### 8.3.2 切换按钮代码示例

清单 8.2 给出了一个简单的音频播放器。这个小程序包含 Play 和 Stop 两个按钮，上面分别有相应的图标（参看图 8-3）。Play 按钮上的图标使用图像文件 play.gif。而 Stop 按钮则使用上一章曾讨论过、但在这里稍做了修改的 TestIcon 类，并以此类指定图标的颜色。TestIcon 类要求明确地给被启用或被禁用的图标分别传递红色值或者灰色值。每当单击了 Play 和 Stop 按钮时，它们就会保持被按下的状态；如果再单击，它们就会弹起来。



图 8-3 带有 JToggleButton 的音频播放器

#### 清单 8.2 用 JToggleButton 创建一个简单的音频播放器 (JTToggleButton.java)

```
// Demonstrates the Toggle Buttons...
/*
<Applet code = JTToggleButton width = 400 height = 75>
</Applet>
*/
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.applet.*;
```

```
public class JTToggleButton extends JApplet {
    // 1. Declare a couple of toggle button references.
    JTToggleButton playButton, stopButton;
    // 2. Audio support.
    AudioClip audioClip = null;
    // 3. Icons to be used for the Stop button.
    Icon enabledIcon = new TestIcon(Color.red);
    Icon disabledIcon = new TestIcon(Color.gray);
    String LF = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
    public void init() {
        // 4. Sets the Motif look-and-feel.
        try {
            UIManager.setLookAndFeel(LF);
        } catch (Exception e) {
            System.err.println("Look and Feel not Initialized!");
        }
        // 5. Create and set the layout.
        GridLayout grid = new GridLayout(1,2);
        this.getContentPane().setLayout(grid);
        this.getContentPane().setBackground(Color.gray);
        // 6. Create and add a Play button.
        ImageIcon playIcon = new ImageIcon("play.gif");
        playButton = new JTToggleButton("Play",playIcon);
        playButton.setBackground(Color.lightGray);
        playButton.setForeground(Color.blue);
        playButton.setMnemonic('p');
        CustomListener cl = new CustomListener();
        playButton.addActionListener(cl);
        this.getContentPane().add(playButton);
        // 7. Create and add a Stop button.
        stopButton = new JTToggleButton("Stop",enabledIcon);
        stopButton.setVerticalTextPosition(SwingConstants.BOTTOM);
        stopButton.setHorizontalTextPosition(SwingConstants.CENTER);
        stopButton.setBackground(Color.lightGray);
        stopButton.setForeground(Color.blue);
        stopButton.setMnemonic('s');
        stopButton.addActionListener(cl);
        this.getContentPane().add(stopButton);
        // 8. Initialize the audio clip.
        if (audioClip == null) {
            audioClip = this.getAudioClip(getCodeBase(),"music.au");
        }
    }
}
```

```
}

// 9. Listener class for the button objects.

class CustomListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JToggleButton button = (JToggleButton) e.getSource();
        // 10. If the play button is clicked...
        if (button == playButton) {
            playButton.setEnabled(false);
            if (!stopButton.isEnabled())
                stopButton.setEnabled(true);
            stopButton.setIcon(enabledIcon);
            audioClip.loop();
        }
        // 11. If the stop button is clicked...
        else if (button == stopButton) {
            stopButton.setEnabled(false);
            stopButton.setIcon(disabledIcon);
            if (!playButton.isEnabled())
                playButton.setEnabled(true);
            audioClip.stop();
        }
    }
}
/*
 * 12. A class that implements the Icon interface;
 * a slight modification (to display the specified
 * colors) of the TestIcon
 */
class TestIcon implements Icon {
    int width = 20; // default icon - width
    int height = 20; // default icon - height
    Color fillColor; // icon color to be specified

    // 13. Constructor that specifies the icon color.
    public TestIcon(Color fillColor) {
        this.fillColor = fillColor;
    }

    // 14. Interface method to paint the icon.
    // using the methods in graphics context object
    public void paintIcon(Component c,
                         Graphics g,
```

```

        int x,
        int y) {
    g.setColor(fillColor);
    g.fillOval(x,y,width,height);
    g.setColor(Color.black);
    g.drawOval(x,y,width,height);
}

// 15. Interface method to retrieve the icon width.
public int getIconWidth() {
    return width;
}

// 16. Interface method to retrieve the icon height.
public int getIconHeight() {
    return height;
}
}

```

### 代码详析

这个 Swing 小程序首先在代码片段 1 中声明了 Play 和 Stop 两按钮的引用。这两个引用的初始化在 init () 内完成。代码片段 2 声明了一个 AudioClip 类型的对象。代码片段 3 定义了一对叫做 enabledIcon 和 disabledIcon 的图标，这两个图标将被显示在 Stop 按钮上。随后，程序内又定义了 Motif 外观的字符串形式。

在代码片段 4 中，通过激活 UIManager 类的 setLookAndFeel () 方法设置了小程序的 Motif 外观。如果初始化过程失败了，必须有一段程序能够捕获到该初始化过程抛出的异常。代码片段 5 创建了 GridLayout 类型的布局，并把它赋予 Swing 小程序的内容窗格。

代码片段 6 初始化了 playIcon 图像图标，这个图标用于显示 playButton 对象。Play 按钮由 JToggleButton 对象表示。可以用 setBackground () 和 setForeground () 方法把按钮的背景和前景初始化成所期望的颜色。SetMnemonic () 方法提供了一个加速键来启动这个音频播放器。

CustomListener 类型的监听对象 cl 通过 addActionListener () 方法注册到 playButton 对象上。这个程序片段的最后一行把 playButton 加到了小程序的内容窗格上。代码片段 7 创建了 Stop 按钮，但是 stopButton 并不使用图像文件，而是用 TestIcon 类对象作为其启用状态或禁用状态的图标。这两个图标已经定义成小程序的成员数据。要想看 TestIcon 类的视图效果，可参看第 7 章“图标、标签和边界”。

代码片段 8 对音频片段文件 (audio clip) 进行初始化。引用变量 audioClip 由 getAudioClip () 方法的返回值进行初始化。getAudioClip () 方法要求音频文件名及其 URL 作为参数。注意，JApplet 类是 AWT 中 Applet 类的子类，因此，可以从 Applet 类中激活 getAudioClip () 方法。

代码片段 9 用 CustomListener 类创建了 Play 按钮和 Stop 按钮的监听对象。这个类实现了 ActionListener 接口，并在 actionPerformed () 方法内提供了处理动作事件所需的代码。

代码片段 10 中的代码表示，一旦选择了按钮操作就可播放音乐。其中 `audioClip.loop()` 语句的作用是，在一次循环内反复播放由音频片段文件装载的音乐。这段代码还可改变 Play 按钮和 Stop 按钮的状态。代码片段 11 的工作方式与代码片段 10 类似，只是其中最后一条语句 `audio.Clip.stop()` 的作用是，一旦按下了 Stop 按钮就停止播放音乐。

代码片段 12 定义了 `TestIcon` 类。如片段 13 所示，这个类的构造函数要求图标的颜色作参数。代码片段 14 用图形环境对象创建了图标。代码片段 15 和片段 16 取得图标的宽度和高度。

## 8.4 复选框

复选框是一个类似于小方框的图标，它可以用有对钩或没有对钩来表示选择或取消选择程序的某些属性。经常会有与其关联的一小段文本来表示该复选框的目的。除外观不同外，复选框与双态按钮的作用相似。

复选框由 `JCheckBox` 类的对象表示，该类是 `JToggleButton` 的子类。`AbstractButton` 类（它是 `JToggleButton` 类的直接超类）包含了复选框所继承的大多数功能。

### 8.4.1 JCheckBox 构造函数

可以用 `Jcheckbox` 类中提供的任何一个构造函数创建复选框对象。这些构造函数允许开发人员指定复选框的图标、与之关联的文本，或者二者的组合。复选框的初始状态，比如有对钩或没有对钩，也能够利用某些构造函数来指定。下面是 `JCheckBox` 类中构造函数的清单：

```
public JCheckBox()
public JCheckBox(Icon icon)
public JCheckBox(Icon icon, Boolean checked)
public JCheckBox(String text)
public JCheckBox(String text, Boolean checked)
public JCheckBox(String text, Icon icon)
public JCheckBox(String text, Icon icon, Boolean checked)
```

这里的参数 `icon` 是复选框图标，其类型为 `Icon`。参数 `text` 是 `String` 类型的任何文本标签。如果复选框按有对钩的情况显示，则 `Boolean` 类型的参数值为 `true`；如果按没有对钩的情况显示，则参数值为 `false`。

### 8.4.2 条目事件和监听程序

每当选中或取消复选框的选择时，复选框就会激活一个 `ItemEvent` 类型的事件。条目事件是由那些从若干个条目中进行选择的组件对象共同激活的事件。条目事件包含了受影响的条目及其当前的选择状态。以下是来自 `ItemEvent` 类的几个方法：

```
public Object getItem()
public ItemSelectable getItemSelectable()
public int getStateChanged()
```

当条目的选择状态发生变化时, getItem() 方法可以检索出条目可选对象 (item selectable object, 比如复选框) 中的条目。所检索的条目可以是条目标签, 也可以是 Integer 类型的对象。对于复选框和单选按钮来说, 所检索的条目就是该条目的标签。

第二个方法 getItemSelectable() 返回条目可选对象 (例如复选框和单选按钮)。这个方法的目的与 java.util.EventObject 类中的 getSource() 方法一样。getStateChanged() 方法返回 SELECTED 或 UNSELECTED, 它们二者是可选对象中的字段。这两个字段表示是选择了该条目还是取消选择该条目。

如果选择了复选框中的条目以进行某种操作, 则监听类必须捕获该条目事件。通过传递监听对象作为参数, 可以在源对象上激活 addItemListener() 方法。该监听类实现了 java.awt.event.itemListener 接口, 并提供接口中 itemStateChanged() 方法的代码。itemStateChanged() 方法接收 ItemEvent 类型的对象参数, 它还包含了当条目被选中时所要执行的代码。

### 8.4.3 复选框代码示例

复选框已经从 AbstractButton 类中继承了一些方法。清单 8.3 给出了其中的一些方法, 并且说明了如何处理由复选框激活的条目事件。

这个小程序沿对角线方向放了三个复选框, 并在复选框的不同清单位置上放置了文本标签, 如图 8-4 所示。当选择一个复选框时, 该程序会显示出你回答的正确程度。因此, 选择所有三个复选框后, 你的回答就会百分之百正确。

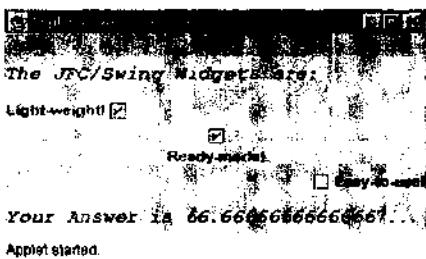


图 8-4 演示 Swing 复选框的小程序

清单 8.3 JCheckBox (TJCheckBox.java)

```
/*
<Applet code = TJCheckBox width = 350 height = 150>
</Applet>
*/
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class TJCheckBox extends JApplet {
    JLabel label2 = null;
```

```
public void init() {  
    //1.Obtain a handle on the JApplet container.  
    Container container = this.getContentPane();  
  
    //2.Set the layout.  
    container.setLayout(new GridLayout(5,1));  
  
    //3.Create the radio buttons.  
    JCheckBox cBox1,cBox2,cBox3;  
    cBox1 = new JCheckBox("Light - weight!");  
    cBox2 = new JCheckBox("Ready - made!");  
    cBox3 = new JCheckBox("Easy - to - use!");  
  
    //4.Add the item listener to the check boxes.  
    CheckBoxListener listener = new CheckBoxListener();  
    cBox1.addItemListener(listener);  
    cBox2.addItemListener(listener);  
    cBox3.addItemListener(listener);  
  
    //5.Set the text position of each button.  
    cBox1.setHorizontalTextPosition(SwingConstants.LEFT);  
    cBox1.setVerticalTextPosition(SwingConstants.BOTTOM);  
    cBox2.setHorizontalTextPosition(SwingConstants.CENTER);  
    cBox2.setVerticalTextPosition(SwingConstants.BOTTOM);  
    cBox3.setHorizontalTextPosition(SwingConstants.RIGHT);  
    cBox3.setVerticalTextPosition(SwingConstants.BOTTOM);  
  
    //6.Set the position of each button in its display area.  
    cBox1.setHorizontalAlignment(SwingConstants.LEFT);  
    cBox2.setHorizontalAlignment(SwingConstants.CENTER);  
    cBox3.setHorizontalAlignment(SwingConstants.RIGHT);  
  
    //7.Create a couple of label to notify what to do and  
    // what happened.  
    Font font = new Font("Monospaced",Font.ITALIC,16);  
    JLabel label1 = new JLabel("The JFC/Swing Widgets are:");  
    label1.setFont(font);  
    label2 = new JLabel("Select the Check Boxes...");  
    label2.setFont(font);  
  
    //8.Finally,add each component to the Swing applet.  
    container.add(label1);  
    container.add(cBox1);  
    container.add(cBox2);  
    container.add(cBox3);  
    container.add(label2);  
}  
//9.Listener class for check boxes.
```

```

class CheckBoxListener implements ItemListener {
    private double n = 0.0;
    public void itemStateChanged(ItemEvent evt) {
        n = n + 100.0/3.0;
        label2.setText("Your Answer is "
            + new Double(n).toString()
            + "% Correct!");
        JCheckBox cBox = (JCheckBox) evt.getSource();
        cBox.setEnabled(false);
    }
}

```

## 代码详析

代码片段 1 创建了一个 Container 类型的引用变量 container，然后把它初始化为内容窗格对象，该对象是由 getContentPane() 方法返回的。变量 container 作为句柄向 Swing 小程序中添加各种组件。代码片段 2 为程序设置了网格布局。

代码片段 3 创建了三个带有相应文本标签的复选框对象。代码片段 4 创建了一个公共的事件监听对象 listener，并把它分别注册到 cBox1、cBox2、cBox3 这三个复选框对象上。

在代码片段 5 内，利用 setHorizontalTextPosition() 和 setVerticalTextPosition() 方法把文本标签放在了确定的位置。这两个方法取 Swing 常数作为其参数。片段 6 中的代码决定复选框在其显示区域内沿水平方向左对齐。缺省地，多个复选框在其显示区域内沿垂直方向居中对齐。

代码片段 7 创建一个表示特定文本字体的字体对象，然后将该字体赋给标签 label1 和 label2。代码片段 8 将两个复选框对象添加到 Swing 小程序的内容窗格上。缺省情况下，小程序具有 Java/Metallic 形式的外观，这是因为程序中没有用其他的代码提供任何特定的外观形式。

复选框对象的监听类是代码片段 9 中所示的一个内部类。监听类 CheckBoxListener 实现了 itemListener 接口，因而也就在 itemStateChanged() 方法的代码内完成了对事件的必要处理。这个方法取 ItemEvent 类型的事件对象作为其参数。

itemStateChanged() 方法内的代码计算用户通过复选框输入的正确的百分比值。代码的下一条语句显示信息 “Your answer is n% correct!”，这里的 n 是计算出的百分比值。代码的其余部分只是简单地禁用复选框。当用户选择第一行和第二行中的复选框时，程序输出结果如图 8-4 所示。当用户选择最后一行的复选框时，用户回答的正确率为百分之百。

图 8-4 中的复选框能够用不同的图标加以装饰，而不是只用其缺省的设置。图标可以由图像文件创建，然后把它传递到相应的复选框构造函数，如下所示：

```

Icon imageIcon = new ImageIcon("iconFile.gif");
JCheckBox checkbox = new JCheckBox("textMessage", imageIcon);

```

每当选中一个复选框时，相关的图标就会变灰，这是因为它已经在事件处理方法中被禁

用了。如果没有把复选框设置为禁用状态，则可以选择用其他的图标来表示选择状态。可以用事件处理函数内的 setIcon() 方法把图标的状态改为被选中状态。图 8-5 给出了使用带图标复选框的同一个小程序。注意，由于已经做了选择，所以其中第一个复选框显示的是指定的已经变灰的图标。

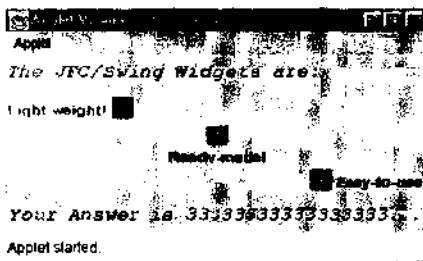


图 8-5 演示带有图像图标的 Swing 复选框的小程序

## 8.5 单选按钮

缺省情况下，单选按钮显示出一个小小的圆形按钮图标，它也可以拥有简单的说明文本。单选按钮一般成组出现，在给定的时刻只能选择其中的一个按钮。也就是说，当一个按钮被选中时，组中的其他按钮自动取消选择。当在一组输入控制中只能操作其中一个时，就可以将该组的选择以单选按钮的方式实现。例如，在具有 Play、Stop、Fast-Forward 以及 Rewind 这样一组按键的媒体播放器上，在一个时刻只能选择一个按键。

单选按钮是 Swing 类 JRadioButton 的对象，后者是 JToggleButton 类的子类。所以，JRadioButton 类的大多数功能都是从 JToggleButton 类的超类 AbstractButton 中继承而来的。

### 8.5.1 JRadioButton 构造函数

可以使用 JRadioButton 类所支持的下列构造函数之一来创建单选按钮对象。构造函数可取参数的组合如下述代码，例如特定的 Icon 类型的图标对象，String 类型的文本对象，以及针对按钮被选中时的初始状态值 true 和未被选中时的状态值 false。

```
public JRadioButton();
public JRadioButton(Icon icon);
public JRadioButton(Icon icon,boolean selected);
public JRadioButton(String text);
public JRadioButton(String text,boolean selected);
public JRadioButton(String text,Icon icon);
public JRadioButton(String text,Icon icon,
                   boolean selected);
```

### 8.5.2 ButtonGroup 类

可以用另外一个叫做 ButtonGroup 的类把若干 JRadioButton 对象组合在一起。Button-

Group 类存放在 javax.swing 包内。要把 JRadioButton 对象添加到一个组里，可以创建一个 ButtonGroup 类型的对象，并激活该对象上的 add() 方法。add() 方法以单选按钮对象作为其参数值。以下是一段典型的代码：

```
...
...
// Create radio button objects
JRadioButton radioButton1, radioButton2, radioButton3;
radioButton1 = new JRadioButton();
radioButton2 = new JRadioButton();
radioButton3 = new JRadioButton();

// Create a button group object
ButtonGroup group = new ButtonGroup();

// Add the button objects to the group
group.add(radioButton1);
group.add(radioButton2);
group.add(radioButton3);

...
...
```

#### 说明:

除了缺省的图标不同外，JRadioButton 类与 JCheckBox 类相似。也可以组合若干个 JCheckBox 类型的对象，起到类似于单选按钮的作用。如此看来，单选按钮好象是多余的。在这里，我们需要回顾一下 Swing 库的主要目标之一——以提供尽可能多的成熟的 GUI 组件来支持用户。

#### 8.5.3 单选按钮代码示例

清单 8.4 演示了单选按钮以及 AbstractButton 类中的一些方法。小程序中给出了两个单选按钮集合，用以选择字体和字体类型（参看图 8-6）。当选择一种字体或字体类型时，所做的选择就要用一个标签对象在程序中显示出来。程序的外观已经设置为 Motif 外观类型。

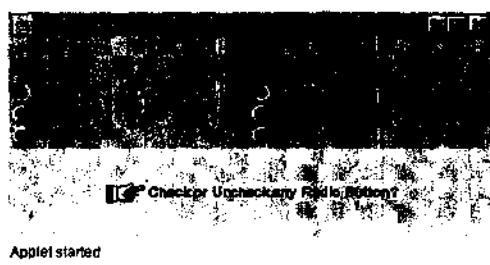


图 8-6 演示 Swing 单选按钮的小程序

**清单 8.4 用来创建两个单选按钮集合的 JRadioButton (TJRadioButton.java)**

```
/*
 * <Applet code = TJRadioButton width = 400 height = 150>
 * </Applet>
 */
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class TJRadioButton extends JApplet {
    // 1. Declare the object references.
    JRadioButton rButton1, rButton2, rButton3;
    JRadioButton rButton4, rButton5, rButton6;

    JLabel label = null;
    String message = null;
    Font font = null;
    Container container;
    String LF = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
    public void init() {
        // 2. Set the look-and-feel of the applet to the Motif type.
        try {
            UIManager.setLookAndFeel(LF);
        } catch (Exception e) {
            System.err.println("Look and Feel not Initialized!");
        }
        // 3. Get a handle on the JApplet container.
        container = this.getContentPane();
        // 4. Initialize the font object.
        font = new Font("Serif", Font.PLAIN, 16);
        // 5. Initialize the message.
        message = "Check or Uncheck any Radio Button?";
        // 6. Create grid layout and set it for the JApplet.
        container.setLayout(new GridLayout(2, 1));
        // 7. Create the JRadioButton objects.
        rButton1 = new JRadioButton("Serif", true);
        CustomListener cl = new CustomListener();
        rButton1.addActionListener(cl);

        rButton2 = new JRadioButton("SansSerif");
        rButton2.addActionListener(cl);

        rButton3 = new JRadioButton("Monospaced");
        rButton3.addActionListener(cl);
    }
}
```

```
// 8. Form a group with these radio buttons.  
ButtonGroup group1 = new ButtonGroup();  
group1.add(rButton1);  
group1.add(rButton2);  
group1.add(rButton3);  
  
// 9. Create the remaining JRadioButton objects.  
rButton4 = new JRadioButton("Plain",true);  
rButton4.addActionListener(cl);  
  
rButton5 = new JRadioButton("Bold");  
rButton5.addActionListener(cl);  
  
rButton6 = new JRadioButton("Italic");  
rButton6.addActionListener(cl);  
  
// 10. Form a group with these radio buttons.  
ButtonGroup group2 = new ButtonGroup();  
group2.add(rButton4);  
group2.add(rButton5);  
group2.add(rButton6);  
  
// 11. Create a JPanel and main label objects.  
JPanel panel = new JPanel();  
panel.setLayout(new GridLayout(4,2));  
Icon picture = new ImageIcon("pointer.gif");  
label = new JLabel(message,picture,SwingConstants.CENTER);  
  
// 12. Add the radio buttons to JPanel.  
panel.add(new JLabel("Font"));  
panel.add(new JLabel("Font Type"));  
panel.add(rButton1); panel.add(rButton4);  
panel.add(rButton2); panel.add(rButton5);  
panel.add(rButton3); panel.add(rButton6);  
  
// 13. Add JPanel and canvas to the JApplet.  
container.add(panel);  
container.add(label);  
}  
  
// 14. The item listener class.  
class CustomListener implements ActionListener {  
    public void actionPerformed(ActionEvent evt) {  
        JRadioButton rButton = (JRadioButton) evt.getSource();  
        if (rButton == rButton1) {  
            font = new Font("Serif",font.getStyle(),16);  
            label.setFont(font);  
            label.setText("You selected the SERIF font!");  
            container.validate();  
        }  
    }  
}
```

```
    }
    else if (rButton == rButton2) {
        font = new Font("SansSerif", font.getStyle(), 16);
        label.setFont(font);
        label.setText("You selected the SANSERIF font!");
        container.validate();
    }
    else if (rButton == rButton3) {
        font = new Font("Monospaced", font.getStyle(), 16);
        label.setFont(font);
        label.setText("You selected the MONOSPACED font!");
        container.validate();
    }
    else if (rButton == rButton4) {
        font = new Font(font.getName(), Font.PLAIN, 16);
        label.setFont(font);
        label.setText("You selected the PLAIN style!");
        container.validate();
    }
    else if (rButton == rButton5) {
        font = new Font(font.getName(), Font.BOLD, 16);
        label.setFont(font);
        label.setText("You selected the BOLD style!");
        label.validate();
    }
    else if (rButton == rButton6) {
        font = new Font(font.getName(), Font.ITALIC, 16);
        label.setFont(font);
        label.setText("You selected the ITALIC style!");
        container.validate();
    }
}
}
```

### 代码详析

代码片段 1 声明了六个对象引用变量，它们将在后面被初始化为 JRadioButton 类型的对象。然后，这个代码段中又声明了一个标签引用变量，一个字符串引用变量，一个字体引用变量和一个容器引用变量。变量 container 被赋予 Swing 小程序的内容窗格。该代码段的最后一条语句给出了 Motif 外观的字符串表示。

在 init() 方法内，代码片段 2 通过激活 UIManager 类中的 setLookAndFeel() 方法，将程序的外观设置为 Motif 类型。每当使用 setLookAndFeel() 方法将程序外观设置为某种特定的类型时，必须捕获 Exception 类型的普通类型异常，或者某种特殊类型的异常。Swing

按钮示例程序中给出了异常情况的列表。

代码片段 3 把容器初始化为程序的内容窗格对象。代码片段 4 创建了一个用以设置特定标签字体的字体对象。代码片段 5 把文本信息赋给了数据成员 message。代码片段 6 设置了程序的网格布局。在代码片段 7 中，前面声明为小程序数据成员的单选按钮引用变量在这里被初始化为单选按钮对象。这个代码段还给出了将监听对象注册到单选按钮对象的代码。

在代码片段 8 中，利用 ButtonGroup 类型的对象 group1，将单选按钮 rButton1、rButton2、rButton3 设置为一个组。在一组单选按钮中，一个时刻只能选择一个按钮。代码片段 9 和 10 初始化了剩下的单选按钮 rButton4、rButton5、rButton6，并将它们组合成了另外一个组 group2。

代码片段 11 创建了一个叫做 panel 的 JPanel 对象和一个叫做 picture 的图像图标。picture 图标是一个小手，它指向当单选按钮被选中时所要显示的一串文字消息。这段代码的最后一行创建了一个 JLabel 类型的标签，用来显示手状图标。

代码片段 12 把所有的单选按钮添加到 panel 对象上。Panel 也包含标签对象，用来指明小程序中的字体和字体类型。代码片段 13 将代码片段 11 中创建的 panel 对象和 label 对象添加到 Swing 小程序中。

代码片段 14 给出了包含事件处理功能的内部类 CustomListener。注意，单选按钮也会激活 ActionEvent 类型的事件，因为 JRadioButton 是 AbstractButton 的子类。这样，该内部类就实现了 ActionListener 接口，而不是 ItemListener 接口。同时，它还提供了实现 actionPerformed() 方法的代码。

actionPerformed() 方法内包含了选择单选按钮时所要执行的代码。例如，当选择单选按钮 rButton1 时，if 语句下面的代码就创建了 Serif 类型的字体。标签上的文本 “You selected the SERIF font!” 就会以 Serif 字体显示出来。这是通过激活 label 对象上的 setFont() 方法和 setText() 方法来完成的。setFont() 方法将 label 对象的字体设置成作为参数传递过来的字体类型。setText() 方法把特定的文本赋给 label 对象。代码片段 14 的其余代码是，当用户选择其他的单选按钮时所要重复执行的动作。图 8-6 给出了当用户选择字体类型为 Serif 时的程序输出结果。请注意小程序中单选按钮的 Motif 外观类型。

# 第9章 列表和组合框

Swing 库提供两类组件来显示一系列项目，即：列表和组合框。Swing 列表就是显示在方框里的项的存档，它允许用户选择一项或几项。本章讨论怎样使用不同类型的数据模型来创建 Swing 列表组件，这些数据模型包括简单的高级化格式、用图标列表表示单元、列表事件和监听程序。本章还展示了怎样通过控制多次鼠标单击来从列表中选择项目。

Swing 组合框是带有一系列项目的小组件，当在其显示区中操作一个按扭时，这个项就会产生一个下拉列表。它每次只允许选择一项。本章讨论有关 Swing 组合框的问题，如：构造函数、数据模型、用图标给项的单元着色。还可以使组合框成为可编辑的，这样能在其下拉式列表中加入项目，关于这一点本章也将讨论。

## 9.1 列表

Swing 列表就是一个象方框一样的组件，它用多行来显示文本项，它在需要显示大量的项时很有用。文本的行被定位在一个单列中的不同单元。列表项通常是可以滚动的，但是它需要单独执行滚动。可以将列表组件放在一个滚动窗格里来达到滚动目的。

Swing 列表组件由类 `JList` 的对象表示。类 `JList` 存储在 `javax.swing` 程序包中，它是 `JComponent` 的直接子类（见图 9-1）。列表组件支持几个接口和类：`ListModel`、`AbstractListModel`、`DefaultListModel`、`ListSelectionModel`、`DefaultListSelectionModel`、`ListCellRenderer` 等等。

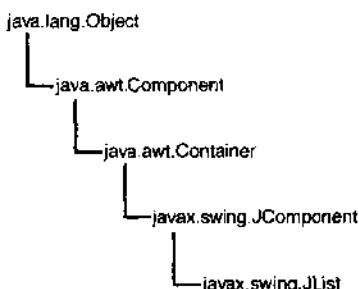


图 9-1 `JList` 类层次结构图

### 9.1.1 `JList` 构造函数

使用 `JList` 支持的任何一个构造函数都可以创建 `JList` 类型的对象。这些构造函数具有不同类型的数据模型。下面列出的是有效构造函数：

```
public void JList()
public void JList(Object[] listData)
```

```
public void JList(Vector listData)
public void JList(ListModel dataModel)
```

第一个构造函数使用了缺省模型。不过，可以通过调用下面列出的 `JList` 对象的任一方法，来设置其他的数据模型：

```
public void setListData(Object[] listData)
public void setListData(Vector listData)
public void setListData(ListModel dataModel)
```

第二个构造函数需要数组形式的数据。可以先创建项目的数组，然后利用此构造函数做出它的列表。例如，下面的代码

```
String[] cities = {"San Jose", "Los Angeles", "San Diego",
"San Francisco", "Sacramento"};
JList citiesList = new JList(cities);
```

创建了显示城市数组的对象。

具有 `Vector` 类型参数的构造函数类似于具有数组类型参数 `Object` 的构造函数。但是，使用向量可以完成各种操作，如使用类 `java.util.Vector` 中提供的功能，可以为列表添加更多的项。例如，下面的代码：

```
Vector months = new Vector();
JList list = new JList(months);
...
months.addElement("January");
...
months.addElement("December");
```

创建了一个具有向量类型数据模型的列表组件。可以在依赖于某种环境和需求的代码的不同位置为此向量添加元素，已经添加的元素可以显示在此列表组件中。

取 `ListModel` 类型值的构造函数用于以综合模型显示可用的列表数据。下一节讨论的接口 `ListModel` 显示了如何使用 `AbstractListModel` 抽象类来创建数据模型，并且提供了一个利用 `ListModel` 类型的数据创建列表对象的示例。

## 9.2 数据模型

为了在列表组件中显示数据的复杂模型，`Swing` 库提供了接口 `ListModel` 以及类 `AbstractListModel` 和 `DefaultListModel`。这些列表模型成员之间的关系显示在图 9-2 中。抽象类 `AbstractListModel` 很容易执行接口 `ListModel` 的某些方法。`DefaultListModel` 类是具有可用向量类型模型的 `AbstractListModel` 的具体子类。

### 9.2.1 接口 `ListModel`

`ListModel` 提供了一个具有一系列方法的设计级接口。这些方法的执行依赖于它的数据模式。因为显示在列表中的数据项是由索引来指明的，所以需要创建一个实现接口 `List-`

Model 的类。下面是接口 ListModel 中方法的声明（可参看附录 A “JFC Swing 快速参考”）。

```
public void addListDataListener(ListDataListener l)
public void removeListDataListener(ListDataListener l)
public int getSize()
public Object getElementAt(int index)
```

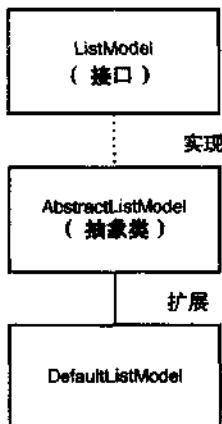


图 9-2 列表模型类之间的关系

方法 getSize () 返回显示在列表组件中的数据大小。方法 getElementAt () 返回列表索引所指示的数据值。其余的方法用来从列表模型类中注册或删除数据监听程序。当数据模式发生改变时，它就会通知其数据监听程序。相关的事件对象具有 ListDataEvent 类型的类。

### 9.2.2 AbstractListModel 类

为了避免实现接口 ListModel 中给出的所有方法，Swing 库提供了一个方便的类，叫做 AbstractListModel。这个类很容易实现方法 addListDataListener () 和 removeListDataListener ()，所以用户只需实现方法 getElementAt () 和 getSize () 即可。

典型的做法是，创建 AbstractListModel 的一个子类，然后执行方法 getElementAt () 和 getSize ()。清单 9.1 演示了如何使用类 ListModel 来创建 JList 对象，它是 AbstractListModel 的子类。图 9-3 显示了此代码的输出结果。

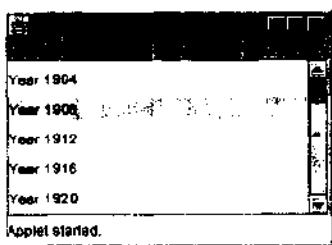


图 9-3 使用闰年列表模式的 JList 组件

**清单 9.1 用来创建闰年列表模式的 JList (TJList1.java)**

```
// Demonstrates how to use the ListModel and JList component

/*
 * <Applet code = TJList1 width = 200 height = 175>
 * </Applet>
 */
import javax.swing.*;
import java.awt.*;

public class TJList1 extends JApplet {
    // 1. Define the data members.
    JList list = null;
    MyListModel model = null;
    JScrollPane scrollPane = null;
    Container container = null;

    public void init() {
        // 2. Get a handle to the applet's content pane and
        // set the grid layout.
        container = this.getContentPane();
        container.setLayout(new GridLayout(1,1));

        // 3. Create objects of type JList, MyListModel,
        // and JScrollPane and add them hierarchically
        // to the applet as shown.
        model = new MyListModel();
        list = new JList(model);
        scrollPane = new JScrollPane(list);
        container.add(scrollPane);
    }
}

// 4. Custom list data model that generates leap years.
class MyListModel extends AbstractListModel {
    int startYear = 1900;
    int endYear = 2000;
    int leapYear;
    // 5. Returns each list element (leap year).
    public Object getElementAt(int index) {
        index = index + 1; // index starts from 0;
                           // so make it 1.
        if ((startYear + index) % 4 == 0)
            return "Year " + (startYear + index);
        else
            return null;
    }
}
```

```
}

// 6.Returns the total count of data; that is,
// total number of years in the present case.
public int getSize() {
    return (endYear - startYear);
}
}
```

### 代码详析

在清单 9.1 中，类 `MyListModel` 表示产生闰年的数据模型。这个类扩展了 `AbstractListModel`，并为 `getElementAt()` 和 `getSize()` 方法提供代码。方法 `getElementAt()` 带有参数 `index`，它计算闰年并返回。方法 `getSize()` 返回年的总数，它提供列表项被计算出的数据点的数值。

`TJList1` 类声明了 `JList`、`MyListModel`、`JScrollPane` 和 `Container` 类型的引用。在 `init()` 方法里，这些引用被其对应的对象初始化。注意，`JList` 构造函数使用 `MyListModel` 类型的对象作为其参数。接着，这些组件中的每一个都被按层次加到另一个之上。将此列表对象加到支持列表项滚动功能的滚动窗格上，然后此滚动窗格又被加到这个 Swing 小程序的容器上。

### 9.2.3 DefaultListModel 类

`DefaultListModel` 类是 `AbstractListModel` 的子类，而且是 `java.util.Vector` 类的当前列表模型的实现形式。因为 `JList` 构造函数允许向量类型的数据模型，所以这个类看起来似乎是多余的。事实上，在使用向量类型数据模型时，这个类为开发者提供了另一种选项。

## 9.3 绘制列表单元

列表单元就是在其中显示列表项的一个很窄的水平空间。除了指示列表项的文本字符串以外，还可以绘制成小图形或图形图标，以表示列表单元。为了方便地绘制列表单元，Swing 库提供了接口 `ListCellRenderer`。此接口只包含一个方法：

```
public interface ListCellRenderer {
    public Component getListCellRendererComponent (
        JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus);
}
```

方法 `getListCellRendererComponent()` 返回的组件将作为列表单元绘制。带有标识符列表的参数就是正在绘制单元的 `JList` 对象。下一个参数 `value`，就是由方法 `getElementAt()`

以指定索引后得到的值。参数 index 就是每个列表值的索引。逻辑参数 isSelected 和 cellHasFocus 指明了指定的单元是否被选定或是否可作为焦点。

因为 JLabel 对象可以拥有文本和图标，所以由 Swing 标签的一个扩展类来实现这个接口。下面显示的是绘制列表单元的典型组件类。

```
class MyCellRenderer extends JLabel implements ListCellRenderer {  
    Icon [] icons; // array of icons to be used for the list items  
  
    public MyCellRenderer (Icon[] icons) {  
        this.icons = icons;  
        setOpaque(true);  
    }  
  
    // Only method to be implemented from ListCellRenderer.  
    public Component getListCellRendererComponent (  
            JList list,  
            Object value,  
            int index,  
            boolean isSelected,  
            boolean cellHasFocus) {  
  
        setText(value.toString());  
        // Set the icon for each selection using the index.  
        if (index < icons.length)  
            setIcon(icons[index]);  
  
        // What happens when a selection is made.  
        if (isSelected) {  
            setBackground(list.getSelectionBackground());  
            setForeground(list.getSelectionForeground());  
        }  
        else {  
            setBackground(list.getBackground());  
            setForeground(list.getForeground());  
        }  
        return this;  
    }  
}
```

在这段代码中，叫做 MyCellRenderer 的自定义组件扩展了类 JLabel 并实现接口 ListCellRenderer。这个类的数据通过构造函数初始化图标数组。方法 getListCellRendererComponent () 返回 MyCellRenderer 类型的组件。在这个方法里，以索引号为基础，使用 JLabel 中的方法 setText () 和 setIcon () 来设置文本标签和图标。

下一步提供的代码显示，当选择某项时将要发生的事。方法 getSelectionBackground () 和 getSelectionForeground () 返回为 JList 对象设置的背景色和前景色。还需要将列表单元的背景色和前景色重新设置为前面代码中所给出的颜色。

### 9.3.1 用图标绘制列表单元

清单 9.2 使用类 MyCellRenderer 来显示 Swing 列表中的图标。该程序利用 Swing 列表创建了一个显示一系列计算机组件的小程序（见图 9-4）。

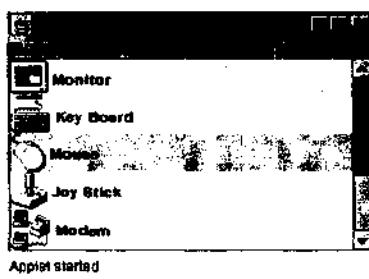


图 9-4 用图标着色的 Swing 列表单元

清单 9.2 JList 创建带有图标列表单元 (TJList.java)

```

/*
 * <Applet code = TJList2 width = 200 height = 150>
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;

public class TJList2 extends JApplet {
    // 1. Labels for the list items.
    String[] ComputerComps = {"Monitor", "Key Board", "Mouse",
        "Joy Stick", "Modem", "CD ROM",
        "RAM Chip", "Diskette"};

    // 2. Create icons for selections and make an array.
    Icon icon1 = new ImageIcon("monitor.gif");
    Icon icon2 = new ImageIcon("keyboard.gif");
    Icon icon3 = new ImageIcon("mouse.gif");
    Icon icon4 = new ImageIcon("joystick.gif");
    Icon icon5 = new ImageIcon("modem.gif");
    Icon icon6 = new ImageIcon("cdrom.gif");
    Icon icon7 = new ImageIcon("ramchip.gif");
    Icon icon8 = new ImageIcon("diskett.gif");
    Icon[] icons = {icon1, icon2, icon3, icon4,
        icon5, icon6, icon7, icon8};

    public void init() {
        // 3. Create a list object to display the computer
        // components.
    }
}

```

```
JList list = new JList(ComputerComps);

// 4.Render the list cells with icons.
list.setCellRenderer(new MyCellRenderer(Icons));

// 5.Add the list to the applet's content pane
// through a scrollpane.
getContentPane().add(new JScrollPane(list));

}

}

// 6.Class that renders the list cells with icons.
class MyCellRenderer extends JLabel implements ListCellRenderer {
    Icon[] icons;

    public MyCellRenderer (Icon[] icons) {
        this.icons = icons;
        setOpaque(true);
    }

    // 7.Method that must be implemented from the interface
    // ListCellRenderer.
    public Component getListCellRendererComponent (
        JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus) {

        // 8.Set the text for the label.
        if (value != null) {
            String text = value.toString();
            setText(text);
        }

        // 9.Set the icon for each selection cell
        // indicated by the index.
        setIcon(Icons[index]);

        // 10.What happens when a selection is made.
        if (isSelected) {
            setBackground(list.getSelectionBackground());
            setForeground(list.getSelectionForeground());
        }

        // NOTE: This step is very important; otherwise, the
        // selection colors don't render as expected!
        else {
            setBackground(list.getBackground());
        }
    }
}
```

```
    setForeground(list.getForeground());
}

// 11.Finally,return this rubber stamp component.
return this;
}
}
```

### 代码详析

代码片段 1 和片段 2 显示了此小程序的数据成员。片段 1 定义了文本字符串数组，每个字符串都是一个计算机组件的名字。片段 2 创建了一串计算机组件的图标。这些图标存储在名称为 icons 的数组中。

在方法 init() 里，片段 3 创建了一系列对象，它们具有作为参数传送到其构造函数的计算机组件数组。片段 4 利用 setCellRenderer() 方法，将 MyCellRenderer 类型的对象赋值给已经创建的列表对象。片段 5 通过滚动窗格对象将此列表组件加到小程序的内容窗格上。在有太多的项需要经过小程序的窗口显示时，这个滚动窗格提供必要的滚动功能。

片段 6 显示了绘制类 MyCellRenderer。注意，绘制组件就象橡皮图章一样绘出一列带有图标和文本的单元。该组件以方法 getListCellRendererComponent() 的参数索引来识别每个单元。当执行此小程序并将鼠标焦点放在某一项时，小程序的显示如图 9-4 所示。

## 9.4 列表选择和事件

在列表组件中，可选择某一项或某一组项来执行某些更进一步的操作。与选择列表项有关，Swing 库支持接口 ListSelectionModel 和类 DefaultListSelectionModel。还有一个表示选择事件的类 ListSelectionEvent 和接收事件的接口 ListSelectionListener。

ListSelectionModel 接口表示选择的当前状态，DefaultListSelectionModel 类为列表选择提供缺省的数据模型。尽管存在这些模型，但是很少用到它们，因为由类 JList 支持的选择方法足以满足一般的需求。

### 9.4.1 选择模式

有三种不同的模式进行项的选择。下面是一系列在接口 ListSelectionModel 中被压缩为数据字段的选择模式：

```
static int SINGLE_SELECTION
static int SINGLE_INTERVAL_SELECTION
static int MULTIPLE_INTERVAL_SELECTION
```

通过设置 SINGLE\_SELECTION 模式，每次只能选择一个列表项。而模式 SINGLE\_INTERVAL\_SELECTION 用于每次选择一块相邻的区域。可以通过保持按下 Shift 键并选择项来执行这种选择类型。最后一个模式叫做 MULTIPLE\_INTERVAL\_SELECTION，它允许每次选择一块或几块相邻的区域。这个类型的选择通过保持按下 Ctrl 键来实现。选择

模式可以通过调用方法 `setSelectionMode()` 赋值给 `JList` 组件。

### 9.4.2 选择方法

`JList` 类支持一系列涉及列表组件各种选择属性的 `set` 和 `get` 方法。下面是经常使用的一些 `set` 方法的简单描述：

```
void setSelectedIndex(int index)
void setSelectedIndices(int[] indices)
void setSelectedValue(Object object, boolean shouldScroll)
```

上述方法带有 `index` 或 `indices` 参数，它在选择状态下设置一个或几个指定的列表项。参数 `indices` 的整数数组在相邻的区域内选定具体单项。方法 `setSelectionValue()` 需要一个象数组元素这样的值，例如 `cities[i]`，来进行选择。可以将 `boolean` 参数设为 `true`，从而允许列表通过视窗上下滚动来显示选定的项。

`JList` 接口和 `DefaultListModel` 类支持处理多重列表选择的方法。为了做出多重列表选择，可以使用选择项的锁定和引线索引特性，并使用下面的方法：

```
void setAnchorSelectionIndex(int index)
void setLeadSelectionIndex(int index)
```

为了检查所有列表项或已指定索引的项的选择状态（被选定或没有选定），可以使用下面列出的 `JList` 类的各种有效方法：

```
boolean isSelectedEmpty()
boolean isSelectedIndex(int index)
```

`JList` 类还提供能改变选择项背景色和前景色的方法。这些方法如下：

```
void setSelectionBackground(Color c)
void setSelectionForeground(Color c)
```

### 9.4.3 选择事件和监听程序代码示例

当作出选择或者选择发生改变时，列表组件的选择模型就会激活 `ListSelectionEvent` 类型的事件。这些事件被实现 `ListSelectionListener` 接口的监听程序类捕获。可以使用方法：

```
Void addListSelectionListener(ListSelectionListener listener)
```

向列表组件注册一个选择监听程序，此方法由 `ListSelectionModel` 接口、`DefaultListModel` 类提供，或者使用类 `JList` 提供的简便方法。

`ListSelectionListener` 接口只包含一个 `valueChanged()` 方法，它需要在执行的类中定义。这个方法将 `ListSelectionEvent` 类型的事件作为其参数。清单 9.3 中给出的程序演示了这些概念。此程序的输出结果显示在图 9-5 中。

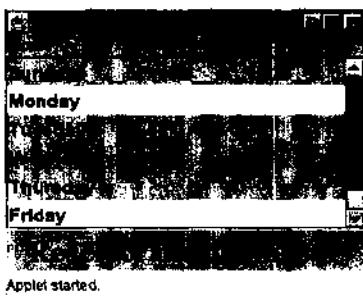


图 9-5 演示 Swing 列表项目选择的小程序

## 清单 9.3 列表项选择 (TJList3.java)

```
// Demonstrates the item-selection from a list widget
/*
 * < Applet code = TJList3 width = 300 height = 200 >
 * </Applet>
 */
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;

public class TJList3 extends JApplet {
    // 1. Declare the required references.
    JList list = null;
    JLabel label = null;
    Container c = null;

    // 2. Create the arrays for the list selections.
    String[] weekdays = {
        "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday" };
    int[] holidays = { 0, 6 }; // Indices for sunday&saturday

    public void init() {
        // 3. Get the handle on the applet's content pane.
        c = this.getContentPane();

        // 4. Create a list object.
        list = new JList(weekdays);
        Font f1 = new Font("SansSerif", Font.BOLD, 16);
        list.setFont(f1);
        list.setSelectionMode(
            ListSelectionMode.MULTIPLE_INTERVAL_SELECTION);
        list.setSelectedIndices(holidays);
        list.addSelectionInterval(2, 4);
    }
}
```

```

list.setSelectionBackground(Color.blue);
list.setSelectionForeground(Color.white);

// 5.Registering a listener class.
ListSelectionListener l = new MyListener();
list.addListSelectionListener(l);

// 6.Add the list to a scroll pane and finally to the applet.
JScrollPane sp = new JScrollPane(list);
c.add(sp);

// 7.Create a label object.
label = new JLabel("Reselect using mouse?",JLabel.CENTER);
Font f2 = new Font("Serif",Font.BOLD,26);
label.setFont(f2);
label.setOpaque(true);
label.setBackground(Color.lightGray);
label.setForeground(Color.black);
c.add(label,BorderLayout.SOUTH);

}

// 8.JList selection event listener class.
class MyListener implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent evt) {
        label.setText((String) list.getSelectedValue());
    }
}

```

代码详析

代码片段 1 声明了 `JList`、`JLabel` 和 `Container` 类型的引用。代码片段 2 定义了周日的字符串数组和假日的整数数组。

在 init() 方法里，片段 3 得到小程序内容窗格的句柄。片段 4 创建了具有周日数据模型的 JList 对象。然后，可以看到一些 JList 方法在运转。setSelectionMode() 方法使用常量 MULTIPLE\_INTERVAL\_SELECTION 启动选择模式。当小程序已经显示时，setSelectionIndices() 将名为 holidays、Sunday 和 Saturday 的数组设置为选定。还有用来控制选择项背景色和前景色的 setSelectionBackground() 和 setSelectionForeground() 方法。

片段 5 创建了一个 ListSelectionListener 类型的对象，并且使用 addListSelectionListener()方法将其注册为列表对象。由于 JList 组件没有预置的滚动功能，所以片段 6 把该组件定位在滚动窗格里，然后再放到小程序中。片段 7 显示了也被加到此小程序中的一个标签。

片段 8 是列表选择监听程序类，它实现 ListSelectionListener 接口并为 valueChanged () 方法提供代码。一旦选定了某个列表项，该组件就会激活 ListSelectionEvent 类型的事件。由 valueChanged () 方法处理该事件，以显示代表选择项的文本。getSelectedValue () 方法检

索已经选定的列表项。小程序显示了在清单 9.3 中设置的多重选择技术。不过，通过操作鼠标和键盘（Ctrl 键），可以选择任何感兴趣的一天或几天。

## 9.5 多重鼠标单击

要从列表组件中选择一项，只需简单地在该项上双击鼠标即可。在列表类中没有预置的可用方法来处理多重鼠标单击。不过，利用事件 MouseEvent 类中的 getClickCount() 方法很容易处理鼠标单击。一旦用户单击鼠标，getClickCount() 方法会立即返回鼠标单击的次数。可以将要执行的代码包装为控制语句的主体，检查指定的鼠标单击次数。

清单 9.4 演示了处理鼠标双击的典型实现方式。该程序用了两个列表组件，其中一个组件显示 Java 产品的列表，另一个代表一张购物卡。当你在要购买的 Java 产品上双击鼠标时，该项产品便会被加到购物卡上，如图 9-6 所示。

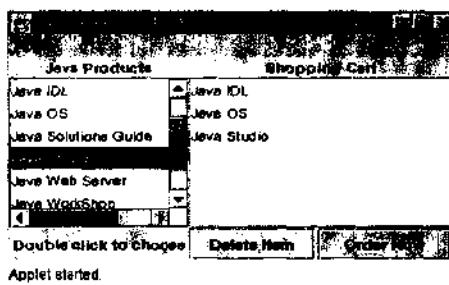


图 9-6 在线购物小程序演示了利用双击鼠标来选择列表组件的过程

当要购买的项都已加到购物卡上时，可单击 Order Now 按钮提交订单。利用 Delete Item 按钮还可以使用户从购物卡上删除指定的项。为此，先将焦点放在购物卡的某一项上并单击，然后再单击 Delete Item 按钮即可。利用这个小程序，可以在 Web 主页上进行在线购物。

清单 9.4 利用双击鼠标操作来选择列表组件 (TJList4.java)

```
// Demonstrates the List component and double mouse clicks
/*
 * <Applet code = TJList4 width = 400 height = 200>
 * </Applet>
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;

public class TJList4 extends JApplet {
    // 1. Data members.
    JList prodList = null;
    JList cartList = null;
```

```
String[] products = {  
    "Internet Access PlusPack2.0",  
    "Java Developer's Companion CD",  
    "Java IDL", "Java OS",  
    "Java Solutions Guide", "Java Studio",  
    "Java Web Server", "Java WorkShop",  
    "JavaScope", "JavaSTAR" };  
  
Container container = null;  
GridBagLayout gridbag = null;  
GridBagConstraints c = null;  
Vector vector = new Vector();  
JScrollPane scrollPane2 = null;  
int index;  
JButton button1, button2;  
  
public void init() {  
    // 2. Prepare the JApplet layout.  
    container = this.getContentPane();  
    gridbag = new GridBagLayout();  
    c = new GridBagConstraints();  
    container.setLayout(gridbag);  
  
    // 3. Necessary labels.  
    c.fill = GridBagConstraints.BOTH;  
    c.gridx = 0; c.gridy = 0;  
    c.gridwidth = 2; c.gridheight = 1;  
    c.weightx = 0; c.weighty = 0;  
    setLabel("Java Products");  
  
    c.gridx = 2;  
    setLabel("Shopping Cart");  
  
    // 4. Create a list of Java products.  
    c.gridx = 0; c.gridy = 1;  
    c.weightx = 1; c.weighty = 1;  
    prodList = new JList(products);  
    MyMouseListener mListener = new MyMouseListener();  
    prodList.addMouseListener(mListener);  
    JScrollPane scrollPanel = new JScrollPane(prodList,  
        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,  
        JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);  
    gridbag.setConstraints(scrollPanel, c);  
    container.add(scrollPanel);  
  
    // 5. Create another list object called cartList.  
    c.gridx = 2;  
    cartList = new JList();  
    scrollPane2 = new JScrollPane(cartList,
```

```
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
gridbag.setConstraints(scrollPane2,c);
container.add(scrollPane2);

// 6.Another label indicating what to do.
c.gridx = 0; c.gridy = 2;
c.weightx = 0; c.weighty = 0;
setLabel("Double click to choose");

// 7.Add a button to delete items.
c.gridx = 2; c.gridwidth = 1;
c.fill = GridBagConstraints.NONE;
button1 = new JButton("Delete Item");
ButtonListener bl = new ButtonListener();
button1.addActionListener(bl);
gridbag.setConstraints(button1,c);
container.add(button1);

// 8.A Button to order items.
c.gridx = 3;
c.ipadx = 10;
button2 = new JButton("Order Now");
gridbag.setConstraints(button2,c);
container.add(button2);

container.validate();
}

// 9.Supporting method for labels.
public void setLabel(String text) {
    JLabel label = new JLabel(text,SwingConstants.CENTER);
    gridbag.setConstraints(label,c);
    container.add(label);
}

// 10.Listener class that recognizes multiple mouse clicks.
class MyMouseListener extends MouseAdapter {
    public void mouseClicked(MouseEvent me) {
        if (me.getClickCount() == 2) {
            index = prodList.locationToIndex(me.getPoint());
            prodList.setSelectedIndex(index);
            prodList.setSelectionBackground(Color.blue);
            prodList.setSelectionForeground(Color.white);
            vector.addElement(products[index]);
            cartList.setListData(vector);
            cartList.repaint();
            scrollPane2.repaint();
        }
    }
}
```

```

        }

    }

    // 11. Button listener to delete items from the shopping cart.
    class ButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent ae) {
            Object srcButton = (JButton) ae.getSource();
            if (srcButton == button1 && !cartList.isSelectionEmpty()) {
                index = cartList.getSelectedIndex();
                vector.remove(index);
                cartList.repaint();
            }
        }
    }
}

```

## 代码详析

代码片段 1 声明了所需要的数据成员。prodList 和 cartList 是两个对象参数，用于 Java 产品的列表和代表购物卡的列表。其中产品列表指的是叫做 products 的字符串数组。该片段的其余数据成员创建了布局参数、向量类型的参数、滚动窗格和一对按钮。其中，向量类型是用于购物卡的数据模型句柄。

在 init() 方法里，片段 2 中显示的代码使得初始布局被做成无序的网格布局。片段 3 将表示 Java 产品和购物卡的标签附在小程序上。

片段 4 使用带有字符串数组的构造函数将产品列表引用初始化。这样得到的列表对象被加到滚动窗格上，然后再将此滚动窗格加到小程序的内容窗格上。片段 5 创建了一个代表购物卡的列表实例。这个列表以类似的方式创建，除了使用没有任何参数的构造函数之外，它也被加到小程序上。

片段 6 将一个标签加到小程序上，该标签可说明产品列表的用途。片段 7 和片段 8 创建了推压式按钮。这些按钮用来从购物卡中删除某项，或者在完成购物清单时提交定单。片段 9 按照无序网格布局生成标签。

在片段 10 中，内部类 MyMouseListener 包装的代码指出，当鼠标在产品列表的一项上单击时，将会发生什么。鼠标监听程序对象被注册为片段 4 中给出的产品列表对象。此监听程序类执行方法 mouseClicked() 来处理事件。在这个方法里，利用 getClickCount() 方法，可捕获鼠标的多重单击事件。

其中的 if 语句根据 getClickCount() 方法的返回值，检查是否单击了两个鼠标。如果单击次数等于 2，则调用类 JList 的一批方法对产品列表和卡片列表对象执行各种操作。

LocationToIndex() 方法返回产品列表中一项的索引，用其在列表对象中的坐标表示。此坐标被应用于传递 Point 类型对象的方法，这些对象由类 MouseEvent 中的方法 getPoint() 返回。一旦获得某个列表项的索引，就可以利用 setSelectedIndex() 方法选定这一项，此方法将这一项的索引作为它的参数值。其余的两个方法，setSelectionBackground() 和 setSelectionForeground() 为选择项设置特殊颜色，以表示特殊的功能。在产品列表中具有

类似的索引项，可使用 `addElement()` 方法将其加到向量对象上。这个向量对象用来初始化卡片列表对象的数据。

片段 11 执行了一个按钮监听程序，它控制当单击删除项目的按钮时，该监听程序控制具体操作。在 `actionPerformed()` 处理器的代码中，`if` 语句检查此按钮是否要删除一项，或选定一项。然后，使用该项的索引，从向量对象中删除这一项，再依次从购物卡中删掉这一项。`JList` 类中的方法 `getSelectedIndex()` 返回选定项的索引。代码的最后一个语句重新显示了修改数据后的卡片列表对象。当执行这个程序时，它就会显示一个能用于 Web 主页的小程序。

## 9.6 组合框

组合框就是一个带条状的显示区，它具有一个下拉式的列表。在显示区的右端有一个小按钮，单击该按钮，可以打开相关的列表。可以选择列表中的项目，并使之显示在显示区中。依赖于出现在列表中的项目数和每次能看到的项目数，下拉式列表可加入滚动条。组合框经常出现在工具条中。

### 提示：

因为组合框是一种带条状组件，所以当放置组件的有效空间很有限时，可以用它来创建 Swing `JList` 组件。不过要注意，组合框每次只支持一项选择。

组合框是轻量级的，而且对 AWT 中有效的“choice”组件也是更好的选择。除了支持项目的选择以外，组合框还可以做成可编辑的。对可编辑的组合框，用户可以在其显示区中输入一项，这一项将被存储在下拉式列表中。

Swing 组合框由 `JComboBox` 类的对象表示。`JComboBox` 类是 `JComponent` 类的直接子类（见图 9-7），它存储在程序包 `javax.swing` 中。组合框的特征已经被分配给不同的支持类，如 `ComboBoxModel`、`ComboBoxEditor`、`ListCellRenderer` 等，我们将在下一节的演示例子中讨论它们。

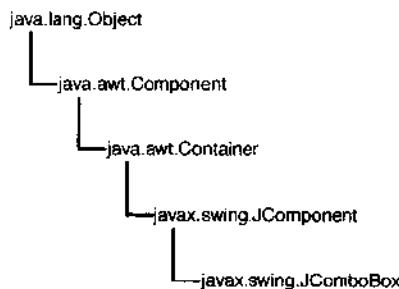


图 9-7 `ComboBox` 的类层次结构图

### 9.6.1 `JComboBox` 构造函数

Swing 组合框是使用缺省数据模型建立，或者用下列构造函数指定的模型来创建的：

```
public JComboBox()  
public JComboBox(ComboBoxModel dataModel)
```

其中没有任何参数的构造函数使用了缺省的数据模型，随后也可以利用类 `JComboBox` 的方法 `setModel()` 来为其设置指定的模型。另一个构造函数由设计接口 `ComboBoxModel` 表示。

除了前面讨论的数据模型以外，还可以使用数组结构中有效数据，或采用项目向量。下列构造函数支持用这些数据结构来创建组合框对象：

```
public JComboBox(Object[] arrayData)  
public JComboBox(Vector vectorItems)
```

参数 `arrayData` 表示数据成员数组，例如，

```
String[] fonts = {"Times Roman", "Arial",  
"Courier New", "Book Antiqua"};
```

另一个构造函数的参数 `vectorItems` 是 `java.util.Vector` 类型的对象。

### 9.6.2 事件和监听程序

组合框对象可以激活两种类型的事件，它们是具有 `ItemEvent` 和 `ActionEvent` 类型的类。一旦在下拉式列表中的选择状态发生改变，它就激活 `ItemEvent` 类型的事件。例如，如果你选择了一项或者在列表中加入一项，那么选择状态就发生了变化，并且激活了一个事件。这个项的事件通常由实现接口 `ItemSelectable` 的组件激活。

被激活项目的事件由 `ItemListener` 类型的对象接收。该监听程序类必须为 `itemStateChanged()` 接口方法提供代码。这样，一旦选择了组合框，要执行的代码就被装进这个方法里。

除了激活项目事件以外，当列表项的状态发生改变时，组合框还能激活 `ActionEvents`。动作事件由实现 `ActionListener` 接口的动作监听程序接收，并为 `actionPerformed()` 方法提供代码。关于对项目事件和动作事件的处理，将在本章的最后部分“按钮和复选框”中详细讨论。

## 9.7 JComboBox 模型

与 `JComboBox()` 的构造函数相似，`ComboBoxModel` 接口允许使用复杂的数据模型创建组合框对象。`ComboBoxModel` 接口代表一般的数据模型，它扩展了 `ListModel`。这样，必须实现这两个接口中的方法来创建一定的数据模型。下面是要求从接口 `ComboBoxModel` 中实现的另外两种方法：

```
public void setSelectedItem(Object anItem)  
public Object getSelectedItem()
```

`setSelectedItem()` 方法带有 `Object` 类型的一般参数，它可以被赋给一个执行类的选择变量。`getSelectedItem()` 方法通过它的执行返回选择值。

为了创建定制的数据模型，必须创建一个实现 ComboBoxModel 接口的类，这个类代表希望使用的数据模型。该类的对象可以作为参数值传递给组合框构造函数，它带有 ComboBoxModel 类型的对象。

在创建实现 ComboBoxModel 接口的类时，可以避免实现 ListModel 接口的所有抽象方法。为了做到这一点，创建一个类，如 MyComboBoxModel，它扩展了 AbstractListModel 类并且实现 ComboBoxModel 接口的方法。它们之间关系的层次结构如图 9-8 所示。

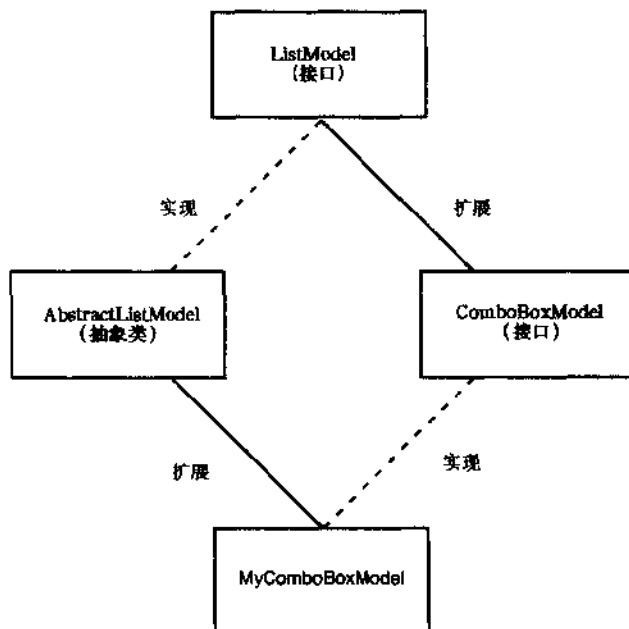


图 9-8 定制模型的层次结构图

MyComboBoxModel 类只要求实现 AbstractListModel 抽象类中两个附加的方法。这些方法不同于 ComboBoxModel 接口里的两个方法。

### 9.7.1 JComboBox 代码示例

清单 9.5 演示了如何使用定制模型来创建组合框对象。该程序的输出结果显示在图 9-9 中。

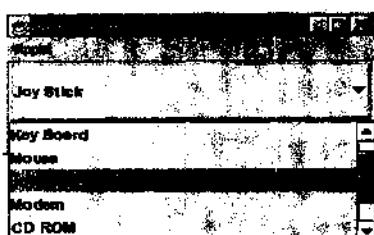


图 9-9 使用定制模型的 JComboBox

**清单 9.5 使用定制模型的 JComboBox (TJComboBox1.java)**

```
/*
 * < Applet code = TJComboBox1 width = 400 height = 50 >
 * </Applet>
 */

import javax.swing.*;
import javax.swing.event.*;

public class TJComboBox1 extends JApplet {
    public void init() {
        // 1. Create a combo box with the custom model.
        JComboBox cbox = new JComboBox(new MyComboBoxModel());
        // 2. Set the number of rows to be visible.
        cbox.setMaximumRowCount(5);
        // 3. Add the combo box object to the applet's content pane.
        getContentPane().add(cbox);
    }
}

// 4. Custom data model for the combo box.
class MyComboBoxModel extends AbstractListModel
    implements ComboBoxModel {
    String[] ComputerComps = {"Monitor", "Key Board", "Mouse",
        "Joy Stick", "Modem", "CD ROM",
        "RAM Chip", "Diskette"};
    String selection = null;

    // Methods implemented from the class AbstractListModel
    public Object getElementAt(int index) {
        return ComputerComps[index];
    }

    public int getSize() {
        return ComputerComps.length;
    }

    public void setSelectedItem(Object anItem) {
        selection = (String) anItem; // to select and register an
    } // item from the pull-down list

    // Methods implemented from the interface ComboBoxModel
    public Object getSelectedItem() {
        return selection; // to add the selection to the combo box
    }
}
```

## 代码详析

Swing 小程序 TJComboBox1 简单地创建并显示了一个组合框，它使用名为 MyComboBoxModel 的定制数据模型。上述代码片段 1 创建了一个具有定制模型的组合框对象。片段 2 描述了在组合框下拉式列表中必须为可见的项的数目。在 init() 方法里的最后一句代码将此组合框对象加到小程序上。

片段 4 显示了 MyComboBoxModel 类，该类定义了定制数据模型。可以看到，为了保存代码，这个类扩展了 AbstractListModel，并实现 ComboBoxModel 接口。该类包含两个数据成员。ComputerComps 数据字段定义了显示在组合框下拉式列表中的字符串标签数组。另一个字段选择中间变量，它存储了下拉式列表中的选定项。

接下来，实现 AbstractListModel 类中的 getElementAt() 方法和 getSize() 方法。并用 ComboBoxModel 接口实现 getSelectedItem() 和 setSelectedItem() 方法。

正如已经看到的列表组件一样，实现 getElementAt() 方法将以给定的索引返回该项标签。getSize() 方法返回显示在下拉式列表中的项的数目。注意，这个信息用来准备组合框的下拉式列表。

当组件的下拉式列表准备好以后，该模型支持用户在下拉式列表中选择一项，该列表将显示在组合框的文本字段里。实现 ComboBoxModel 接口的另外两个方法可以得到这个功能。必须实现 setSelectedItem() 方法来存储利用中间变量（如 selection）选定的项。必须实现的另一个 getSelectedItem() 方法返回已经选定的项。然后，该项将显示在组合框里。试试在下拉式列表中选择一项并按期望的样子显示它。

## 9.8 绘制项目单元

我们已经看到怎样使用图标绘制列表单元。列表对象基本上是由叫做 ListCellRenderer 的设计级接口来实现的。同样的接口可以用来绘制组合框下拉式列表的项目单元。

ListCellRenderer 接口希望执行它的唯一方法 getListCellRendererComponent() 的代码。此方法返回一个组件，它绘制类似图章的下拉式列表单元。在 getListCellRendererComponent() 方法中，需要处理有关文本标签、图标以及列表单元上鼠标焦点等属性的设置。

### 9.8.1 带有标签和图标的 JComboBox 代码示例

清单 9.6 演示了怎样绘制带有标签和图标的组合框（如图 9-10 所示）。

清单 9.6 绘制用图标和标签表示的 JComboBox 项目 (TJComboBox2.java)

```
/*
 * <Applet code = TJComboBox2 width = 400 height = 50>
 * </Applet>
 */
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
```

```
public class TJComboBox2 extends JApplet {
    // 1. Labels for the combo box pull-down list.
    String[] ComputerComps = {"Monitor", "Key Board", "Mouse",
        "Joy Stick", "Modem", "CD ROM",
        "RAM Chip", "Diskette"};
    // 2. Create icons for selections and make an array.
    Icon icon1 = new ImageIcon("monitor.gif");
    Icon icon2 = new ImageIcon("keyboard.gif");
    Icon icon3 = new ImageIcon("mouse.gif");
    Icon icon4 = new ImageIcon("joystick.gif");
    Icon icon5 = new ImageIcon("modem.gif");
    Icon icon6 = new ImageIcon("cdrom.gif");
    Icon icon7 = new ImageIcon("ramchip.gif");
    Icon icon8 = new ImageIcon("diskett.gif");
    Icon[] icons = {icon1, icon2, icon3, icon4,
        icon5, icon6, icon7, icon8};
    public void init() {
        // 3. Create a combo box with the custom model.
        JComboBox cbx = new JComboBox(ComputerComps);
        cbx.setRenderer(new MyCellRenderer(icons));
        // 4. Set the number of rows to be visible.
        cbx.setMaximumRowCount(5);
        // 5. Add the combo box object to the applet's content pane.
        getContentPane().add(cbx);
    }
}

// 6. Class that renders the list cells with icons.
class MyCellRenderer extends JLabel implements ListCellRenderer {
    Icon[] icons;
    public MyCellRenderer (Icon[] icons) {
        this.icons = icons;
        setOpaque(true);
    }
    // 7. Method that must be implemented from the interface.
    public Component getListCellRendererComponent (
        JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus) {
        // 8. Set the text for the label.
        setText((String) value);
        if (isSelected)
            setForeground(Color.red);
        else
            setForeground(Color.black);
        return this;
    }
}
```

```
if (value != null) {
    String text = value.toString();
    setText(text);
}

// 9. Set the icon for each selection cell indicated
// by the index.
if ((index != -1) && (index < icons.length)) {
    setIcon(icons[index]);
}

// 10.What happens when a selection is made.
if (isSelected) {
    setBackground(list.getSelectionBackground());
    setForeground(list.getSelectionForeground());
}

// NOTE: This step is very important; otherwise, the
// selection colors don't render as expected!
else {
    setBackground(list.getBackground());
    setForeground(list.getForeground());
}

// 11.Finally, return this rubber stamp.
return this;
}
}
```

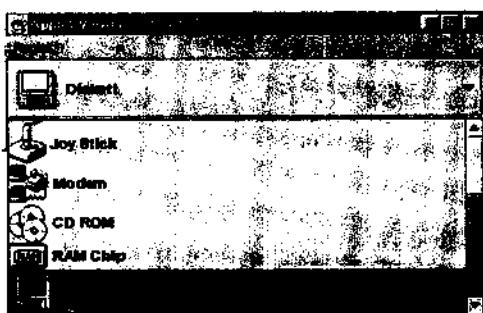


图 9-10 绘制用图标和标签表示的 JComboBox 项目

### 代码详析

清单 9.6 是 Swing 小程序，它创建了一个组合框并将它附加在小程序上。代码片段 1 显示了该小程序的数据字段，它是一个字符串标签数组。这些标签将用来绘制组合框的列表单元。片段 2 创建了一批图标对象并以 Icon 类型的数组形式存储它们。除标签以外，这些图

标还用来绘制列表单元。

在 init() 方法里, 使用构造函数创建了一个组合框对象, 该构造函数接收一个字符串标签数组作为它的参数值。接下来, 使用方法 setRenderer() 设置组合框对象, 该方法具有由 MyCellRenderer 类创建的一个着色对象。MyCellRenderer 类扩展了 JLabel 并实现了 ListCellRenderer 接口。因此, 这个类包括一个标签和一个图标。

片段 4 设置组合框的下拉式列表, 以显示指定数目的项。可以使用 setMaximumRowCount() 方法来实现这一点。片段 5 将此组合框对象加到小程序的内容窗格上。

片段 6 显示了 MyCellRenderer 类。在这个类中, 必须实现 getListCellRendererComponent() 方法, 它包含一些重要的因素作为它的参数。每个这样的参数都带有一些关于如何绘制组合框列表单元的重要信息。List 类型的参数是指组合框的下拉式列表, value 是指要绘制的标签, integer 类型的索引是指定组合框列表单元的指针。逻辑参数 isSelected 提供有关指定的列表单元是否已经被选定的信息。类似地, 逻辑字段 cellHasFocus 说明指定的单元是否已经接受了焦点属性。

片段 8 使用方法 setText() 为 MyCellRenderer 组件设置文本。注意, 这个组件是 Swing 类型的标签对象, 因为它扩展了 JLabel。用类似的方式, 片段 9 根据 index 参数设置图标。这个片段使用了其父类中的 setIcon() 方法。

作好选择后, 片段 10 就为指定的列表单元设置背景色和前景色。一旦绘制了选定的颜色, 就必须明确地给那些未选中的单元重新设置背景色和前景色。在控制语句的 else 子句中处理了这个情况。

最后, 在片段 11 中, 需返回由 JLabel 属性构成的橡皮图章组件。组合框利用该组件用指定的标签和图标绘制其列表单元。

## 9.9 可编辑的组合框

组合框的一个重要特征就是, 它可以成为可编辑的。即用户可以在组合框的显示域中输入任何文本, 然后按 Enter 键, 该文本就会被加到下拉式列表中。

在缺省情形下, Swing 组合框是不可编辑的, 但是通过在组合框对象上调用下面的方法:

```
public void setEditable(boolean trueOrFalse)
```

可以使该组合框变为可编辑的。

为了在代码执行的任何阶段确定指定的组合框是否为可编辑的, 可以在该对象上调用下面的方法:

```
public boolean isEditable()
```

如果该组合框是可编辑的, 方法 isEditable() 就返回 true; 否则它就返回 false。

为了在组合框的编辑器里绘制或编辑项目, 该组件使用了 ComboBoxEditor 类型的缺省编辑器。ComboBoxEditor 是一个设计接口, 它允许创建任何指定的感兴趣的编辑器。下面是在实现此接口的类中定义的方法:

```
public Component getEditorComponent()
public void setItem(Object anObject)
public Object getItem()
public void selectAll()
public void addActionListener(ActionListener l)
public void removeActionListener(ActionListener l)
```

其中 `getEditorComponent` 方法检索用来编辑项目的组件。该组件给出编辑器的句柄来改变它的属性，如背景色和前景色、字体等等。`setItem()` 方法向编辑到组合框列表中的项赋值；补充的 `getItem()` 方法只是简单地获取被编辑的项目；另一个 `selectAll()` 方法给出从编辑器中选定的所有项目。

`addActionListener()` 方法向编辑器注册了一个 `ActionListener` 类型的对象。当一个被编辑的项发生改变时，就会产生一个动作事件。另一个 `removeActionListener()` 方法删除已经注册过的那个监听程序对象。在创建了自定义的编辑器以后，就可以调用下面的方法把它放到组合框里：

```
public void setEditor(ComboBoxEditor customEditor)
```

组合框的当前编辑器可以使用下面的方法来获取：

```
public ComboBoxEditor getEditor()
```

通过调用下面组合框实例中的方法，已经设置了编辑器的组合框可以适当配置来显示缺省项：

```
public void ConfigureEditor(ComboBoxEditor editor, Object anItem)
```

缺省的组合框编辑器使用 Swing `JTextField` 作为它的编辑组件，所以确实没必要涉及接口 `ComboBoxEditor` 的实现。不过，如果有特殊需求的话，则需使用实现接口 `ComboBoxEditor` 的编辑器类。

### 9.9.1 可编辑的 JComboBox 代码示例

清单 9.7 演示了怎样去创建一个可编辑的组合框。已经编辑的文本在按回车键后被加到下拉式列表的顶端（见图 9-11）。此代码还能够过滤出组合框列表中的重复项。

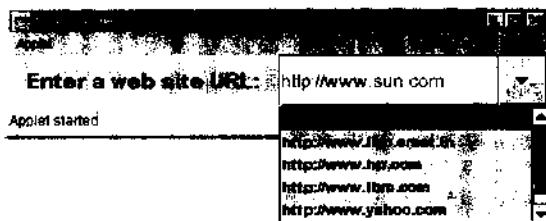


图 9-11 显示可编辑的组合框

**清单9.7 可编辑的 JComboBox (TJComboBox3.java)**

```
// Demonstrates editable combo boxes
/*
 * < Applet code = TJComboBox3 width = 450 height = 50 >
 * </Applet >
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TJComboBox3 extends JApplet {
    // 1. Declare the references.
    JComboBox cbox = null;
    Container c = null;

    // 2. Define an array of sites to be shown in the
    // combo box pull-down list, by default.
    String[] defaultSites = {
        "http://www.sun.com", "http://www.iisc.ernet.in",
        "http://www.hp.com", "http://www.ibm.com",
        "http://www.yahoo!.com", "http://www.news.com"};
    String defaultEdit = "http://www.javasoft.com";

    public void init() {
        // 3. Get a handle on the applet's content pane.
        // and prepare the container with grid layout
        c = this.getContentPane();
        c.setLayout(new GridLayout(1, 2));

        // 4. Create the combo box with the list of default sites.
        cbox = new JComboBox(defaultSites);
        cbox.setOpaque(true);

        // 5. Make the combo box editable.
        cbox.setEditable(true);

        // 6. Configure the combo box editor.
        cbox.configureEditor(cbox.getEditor(), defaultEdit);

        // 7. Rows to be visible without scrollbars.
        cbox.setMaximumRowCount(5);

        // 8. Set the combo box editor colors and font.
        ComboBoxEditor cboxEditor = cbox.getEditor();
        Component editorComp = cboxEditor.getEditorComponent();
        editorComp.setBackground(Color.white);
        editorComp.setForeground(Color.blue);
    }
}
```

```
Font f1 = new Font("Dialog", Font.PLAIN, 16);
editorComp.setFont(f1);

// 9. Font for the combo box popup list.
cbox.setFont(f1);

// 10. Register the action listener.
cbox.addActionListener(new CboxListener());

// 11. A label to indicate what to do.
JLabel label = new JLabel("Enter a web site URL:",
                           JLabel.CENTER);
label.setFont(new Font("SansSerif", Font.BOLD, 18));

// 12. Finally...
c.add(label);
c.add(cbox);
}

class CboxListener implements ActionListener {
    public void actionPerformed(ActionEvent evt) {
        // Helpful flag
        boolean isItemPresent = false;

        // 13. Check if the item already exists in the pop-up list.
        for (int i = 0; i < cbox.getItemCount(); i++) {
            if (cbox.getItemAt(i).equals(cbox.getSelectedItem())) {
                isItemPresent = true;
                break;
            }
        }

        // 14. If the item is not present in the pop-up list...
        if (!isItemPresent) {
            cbox.insertItemAt(cbox.getSelectedItem(), 0);
            isItemPresent = false;
        }
    }
}
```

### 代码详析

清单 9.7 首先声明了一个组合框参数并作为它的数据成员。另一个 Container 类型的引用为小程序的内容窗格提供了句柄。片段 2 定义了一些缺省地点表；当该程序启动时，这些地点就会出现在组合框的下拉列表中。另一个数据变量 defaultEdit 将显示在组合框的可编辑区内。

在 init() 方法里，片段 3 得到小程序内容窗格的句柄，并且给小程序准备了一行两列

的格状布局。片段 4 和片段 5 创建了组合框对象，调用 JComboBox 方法，使其 setEditable () 参数为 true，使组合框成为可编辑的。

片段 6 实现 JComboBox 的另一个叫做 configureEditor () 的方法。此方法用来将组合框编辑器设置为缺省文本。注意，必须应用 ComboBoxEditor 和 Object 类型的参数值。

片段 7 中的 setMaximumRowCount () 方法为组合框准备了显示指定个数的项目下拉列表。通过在组合框对象上调用 getEditor () 方法，片段 8 得到缺省组合框编辑器的句柄。利用该句柄，再使用 getEditorComponent () 方法创建一个 Component 类型的组件。在获得这个组件以后，再使用类 Component 支持的 API 来修改其属性。这样，编辑器的背景色和前景色分别被设置为白色和兰色。此片段还将编辑器的字体设置为给定的新字体。

片段 9 为下拉列表指定新字体。注意这样一些对象，必须调用其上的 setFont () 方法将这种字体赋给组合框编辑器和下拉列表。片段 10 将动作监听程序注册到组合框对象上。片段 11 创建了一个具有指定字体的标签对象。最后，片段 12 将这些对象加到小程序的内容窗格上。

在组合框的监听程序类里面，片段 13 检查组合框的下拉列表中是否有已经编辑过的项。 getItemCount () 方法返回出现在下拉列表中的项的个数； getItemAt () 方法返回指定索引值的项目，这个项目将与输入到组合框中的项目相比较。按回车键以后，getSelectedItem () 方法就返回在组合框编辑器中输入的项目。然后，如果该项目已经在下拉列表中，则标志 isItemPresent 就会被设置为 true。

片段 14 在组合框下拉列表的指定索引位置插入一项目。方法 insertItemAt () 有两个参数，分别为待插入项和索引位置。

# 第 10 章 表 格

表格可以以行和列的格式在用户接口上显示数据，而且允许随意地对数据进行编辑。

Swing 表格模型功能强大、灵活并易于执行，它通常是缺省模型的更好选择。有关表格的类和方法的详情，请参看附录 A：“JFC Swing 快速参考”。

## 10.1 简单表格

Swing 表格由 JComponent 的扩展类 JTable（见图 10-1）表示，它存储在 javax.swing 程序包中。本节演示如何利用两个表格构造函数创建简单表格。

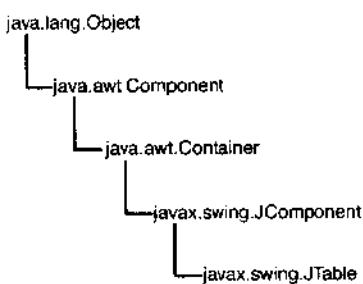


图 10-1 JTable 类层次结构图

### 10.1.1 JTable 构造函数

JTable 支持的构造函数有七个，它们分别采用不同的参数来生成表格。可以随意指定表格的行数和列数、表格模型、表格列模型、选择模型等等。下面是这些构造函数说明：

```
public JTable()
```

创建了初始化后的 JTable 对象，它具有缺省的数据模型、缺省的列模型和缺省的选择模型。

```
public JTable(int numRows, int numColumns)
```

创建了具有指定行数和列（具有空单元）数的 JTable 对象。此表格对象使用了缺省的表格模型。

```
public JTable(Object[][] rowData, Object[] columnNames)
```

创建了 JTable 对象，它显示由 rowData 指定的二维数组的值。其中列名由叫做 columnNames 的数组指定。

```
public JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm)
```

创建了初始化了的 JTable 对象，它具有指定的数据模型 dm、列模型 cm 和选择模型 sm。可以使用此表格创建更复杂的表格。

```
public JTable(TableModel dm, TableColumnModel cm)
```

创建了初始化了的 JTable 对象，它的数据模型为 dm、列模型为 cm。而它的选择模型是缺省模型。

```
public JTable(TableModel dm)
```

创建了初始化了的 JTable 对象，它具有指定的数据模型 dm。其列模型和选择模型使用了缺省模型。

```
public JTable(Vector rowData, Vector columnNames)
```

创建了 JTable 对象，它显示了存储在向量里的数据。此表格行的数据结构是向量的向量，而它的列名就是简单的向量。当在程序中使用向量来存储数据时（可以用表格的形式显示它们），这个构造函数就很有用。

为了能快速实现一个基本表格，可以选择前面介绍的只使用行和列的构造函数，或者使用需要数据数组的构造函数。注意，使用这些构造函数创建的表格具有一定的内在缺陷。基本上，这些表格只允许 String 类型的数据。而且，它们不能变为可编辑的。为了克服这些缺陷，需要使用下一节介绍的定制模型来实现表格。

### 10.1.2 表头

表头就是表格中每一列的顶部，在这里显示该列的标题。在 Swing 中，列头由 JTableHeader 类表示。可以调用 getTableHeader() 方法来检索表格对象的表头。如果没有赋给其他的表头，还有另一种叫做 getDefaultTableHeader() 的方法可以返回缺省的表头。使用 setTableHeader() 类可以指定表格的表头。

### 10.1.3 调整列的尺寸

在缺省情况下，Swing 中创建的表格允许调整列的尺寸。在表格的列分隔线上拖动鼠标，可以交互形式调整列的宽度。在调整期间，列的状态可以使用下面的这些常量来控制：

```
static int AUTO_RESIZE_OFF
```

不允许改变任何列宽。

```
static int AUTO_RESIZE_ALL_COLUMNS
```

以成比例的方式控制所有列尺寸的调整。这样能保证在调整以后所有列的空间仍然一样。

```
static int AUTO_RESIZE_NEXT_COLUMN
```

以这样一种方式来控制：当调整其中一列时，在对应方向上的下一列也得到调整。

```
static int AUTO_RESIZE_SUBSEQUENT_COLUMNS
```

通过改变下一列的宽度来保持总宽度不变。

```
static int AUTO_RESIZE_LAST_COLUMN
```

只调整最后一列，其余的列保持不变。

#### 10.1.4 简单表格的代码示例

清单 10.1 创建了一个非常简单的表格并将它加到一个小程序上。此程序使用的表格构造函数要求指定表格的行数和列数。注意，此表格缺省为可编辑的，它的输出结果显示在图 10-2 中。

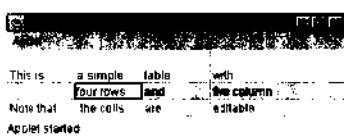


图 10-2 有四行和五列的简单表格

##### 清单 10.1 简单表格 (TJTable1.java)

```
// Demonstrates how to create a simple table with the
// specified number of rows and columns.

/*
 * <applet code = TJTable1 width = 350 height = 150>
 * </applet>
 */

import javax.swing.*;
public class TJTable1 extends JApplet {
    public void init() {
        // Create a table using the number of rows and
        // columns and add it to the content pane.
        JTable table = new JTable(4,5); // 4 rows & 5 columns
        getContentPane().add(table);
    }
}
```

#### 代码详析

清单 10.1 是一个小程序，它显示了一个有四行五列的简单表格。在 init() 方法里只有两个语句。在片段 1 中，第一个语句使用构造函数 JTable (int rows, int columns) 创建了一个表格。第二个语句将这个表格加到小程序的内容窗格上。

#### 10.1.5 由数组创建表格的代码示例

清单 10.2 显示了另一个程序，它使用数组中可用的数据创建了一个 Swing 表格。其列

数组就是显示在每个列的简单标题。其行由一列数组表示，因为每一行本身就是要显示的一组数据项。此表格已经置于一个小程序中。输出结果见图 10-3。

#### 说明：

表头必须明确地加在容器的顶部，简单地将表格加到它的容器上就不会出现表头。

Name	Size (Bytes)	Date	Directory
AUTOEXEC.BAT	149	09-11-98	false
REAL	DIR	12-11-97	true
WINDOWS	DIR	03-24-97	true
COMMAND.COM	92879	07-11-97	false

图 10-3 使用数据数组创建的表格

#### 清单 10.2 使用数据数组创建的表格 (TJTable2.java)

```
// Demonstrates how to create a simple table with the
// specified row data and column data.

/*
* <applet code = TJTable2 width = 350 height = 100>
* </applet>
*/

import javax.swing.*;
import java.awt.*;
import java.util.*;

public class TJTable2 extends JApplet {
    // Create an array of names to be displayed in the table
    // header.
    String[] columnNames = {"Name", "Size (Bytes)", "Date", "Directory"};

    // Create an array of row data (each row is an array) to be
    // displayed in the rows of the table.
    Object[][] rowData = {
        {"AUTOEXEC.BAT", "149", "09 - 11 - 98", new Boolean(false)},
        {"REAL", "DIR", "12 - 11 - 97", new Boolean(true)},
        {"WINDOWS", "DIR", "03 - 24 - 97", new Boolean(true)},
        {"COMMAND.COM", "92879", "07 - 11 - 97", new Boolean(false)}};

    public void init() {
        // 1.Create a table by using the row and column data.
        JTable table = new JTable(rowData, columnNames);

        // 2.Add the table column header and the
        // table to the content pane of the applet.
    }
}
```

```
getContentPane().add(table.getTableHeader(), BorderLayout.NORTH);
getContentPane().add(table);
}
}
```

### 代码详析

在清单 10.2 中，显示在表格里的数据已经以数组的形式存储。它的列数据是叫做 `columnName` 的字符串数组。行数据则是多个数组，这里每个数组都包含一个特殊行的数据。

在 `init()` 方法里，片段 1 使用数组中定义的数据创建了一个表格对象。在片段 2 中，此表格被加到小程序的内容窗格上。包含标题的列头也被加到小程序的顶部从而完全地显示该表格。

#### 说明：

如果不使用滚动窗格来显示表格，就必须明确地将此表格加在内容窗格的顶部。

## 10.2 表格模型

前面讨论了怎样利用在数组中可用的数据来创建树。但事实上，这些数据可以以更复杂的形式出现。为了使用复杂的数据格式来创建表格，必须依赖于表格模型。实际上，开发者必须意识到，只有广泛地使用表格模型，才能实现 Swing 表格的全部潜能。幸运的是，创建 Swing 的定制表格模型比你想象的要容易。

程序包 `javax.swing.table` 支持叫做 `TableModel` 的 Swing 表格的设计级接口。`TableModel` 接口包括下列被执行的方法：

```
public void addTableModelListener(TableModelListener l)
```

将监听程序加到列表上，每当数据模型发生改变时，此列表就会发出通知。

```
public Class getColumnClass(int columnIndex)
```

返回该列中的最小公分母类。

```
public int getColumnCount()
```

返回由数据源对象管理的列数。

```
public String getColumnName(int columnIndex)
```

返回在 `columnIndex` 位置的列名。

```
public int getRowCount()
```

返回由数据源对象管理的记录数。

```
public Object getValueAt(int rowIndex, int columnIndex)
```

返回在 `columnIndex` 和 `rowIndex` 位置的单元的属性值。

```
public boolean isCellEditable(int rowIndex, int columnIndex)
```

如果在 rowIndex 和 columnIndex 位置的单元是可编辑的，返回 true。

```
public void removeTableModelListener(TableModelListener l)
```

从列表中删除监听程序，每当数据模型发生改变时，此列表就会发出通知。

```
public void setValueAt(Object aValue, int rowIndex, int columnIndex)
```

给位于 columnIndex 和 rowIndex 位置的单元的记录赋以属性值。

### 10.2.1 缺省的表格模型

在缺省状态下，表格对象使用 DefaultTableModel 类的数据模型。这个类实现 TableModel 接口，并扩展类 AbstractTableModel。当没有其他模型赋以表格时，可以通过调用 getModel() 方法检索缺省的表格模型对象。

**提示：**

通常不必使用缺省模型，因为定制表格模型很容易创建，而且它足够灵活并且功能强大，可以代表复杂的数据。

### 10.2.2 抽象表格模型

Swing 库也支持叫做 AbstractTableModel 的类，使开发者能够很容易地创建定制的表格模型。定制表格模型类可以扩展 AbstractTableModel，并且只要求实现下面的方法：

```
public int getRowCount();
public int getColumnCount();
public Object getValueAt(int row, int column);
```

### 10.2.3 表格模型事件和监听程序

当表格数据模型发生改变时，就会激活 TableModelEvent 类型的事件。这些事件由实现接口 TableModelListener 的对象监听。该接口包含 tableChanged() 方法，该方法包括数据模型经历某些变化时将要执行的代码。

另外，还可以精确地知道在何处的数据发生了改变。要达到这个目的，可以调用 TableModelEvent 事件类支持的方法。有关这些方法的列表请参阅附录 A。

### 10.2.4 定制表格模型代码示例

清单 10.3 显示的程序使用定制的表格模型从指定目录中得到文件和子目录。此应用程序利用 Swing 表格显示计算机主目录的内容。通过在框架顶端的文本字段里输入路径，还可以看到另一个目录下的内容。程序的输出结果见图 10-4。

Name	Size (bytes)	Last modified
INF		
COMMAND		
SYSTEM		
HELP		
NETDET.INI	7885	
FORMS		
SMARTDRV.EXE	45145	
REGEDIT.EXE	120320	
ACCSTAT.EXE	24676	

图 10-4 使用定制模型产生的表格

**清单 10.3 定制表格模型 (TTableModel.java)**

```
// Demonstrates a custom table model.

import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;

public class TTableModel extends JFrame {
    Container container;
    JTable table;
    JScrollPane scrollPane;
    JLabel label;
    JTextField textField;

    public TTableModel() {
        // 1. Assign a title to the frame and get
        // the handle on the content pane.
        super("TTableModel");
        container = this.getContentPane();

        // 2. Create a label and text field and add
        // them to a panel.
        label = new JLabel(
            "Enter A Valid Directory Name and Press Return",
            JLabel.CENTER);
        textField = new JTextField();
        textField.addActionListener(new TextFieldListener());
        JPanel panel = new JPanel(new GridLayout(2,1));
        panel.add(label);
        panel.add(textField);

        // 3. Get the root/system home. Use this home directory
        // as the default value for the text field.
        String homeDir = System.getProperty("user.home");
        textField.setText(homeDir);
    }
}
```

```
// to create a file object that is used by the directory
// or file system model construct the table model. Also
// display the home directory in the text field.
String home = System.getProperty("user.home");
table = new JTable(new DirectoryModel(new File(home)));
table.createDefaultColumnsFromModel();
textField.setText(home);

// 4. Add the panel and table to the container.
container.add(BorderLayout.NORTH, panel);
container.add(new JScrollPane(table));

// 5. Frame settings.
// Add the window closing listener.
addWindowListener(new WindowEventHandler());
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
setBackground(Color.white);
setSize(350,300); // Frame width = 350, height = 300
show(); // Display the frame
}

// 6. Window event handler.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}

// 7. The main method.
public static void main(String[] args) {
    TTableModel frame = new TTableModel();
}

// 8. Text field listener.
class TextFieldListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // 9. Get the next directory name entered in
        // the text field, prepare the model, and display
        // the table by assigning the new data.
        DirectoryModel model = new DirectoryModel(
            new File(textField.getText()));
        table.setModel(model);
    }
}

// 10. The "Directory" or "FileSystem" model.
```

```
class DirectoryModel extends AbstractTableModel {  
    File directory;  
    String[] members;  
    int rowCount;  
  
    // 11. Model constructor.  
    public DirectoryModel(File dir) {  
        directory = dir; // Hold the directory  
        members = dir.list(); // Get the list of files  
                           // and subdirectories  
        if (members != null)  
            // Table rows = No.of entities inside the directory  
            rowCount = members.length;  
        else {  
            // If the member list is null, row count should be zero.  
            rowCount = 0;  
            // This can happen if an invalid directory is entered  
            // in the text field.  
            System.out.println("Not a valid directory!");  
        }  
    }  
  
    // 12. Retrieve the number of rows for the table to be prepared.  
    public int getRowCount() {  
        return members != null? rowCount:0;  
    }  
  
    // 13. Similarly, retrieve the column count.  
    public int getColumnCount() {  
        return members != null? 3:0;  
    }  
  
    // 14. Retrieve each of the table values at the specified  
    // row and column.  
    public Object getValueAt(int row,int column) {  
        if (directory == null || members == null) {  
            return null;  
        }  
  
        File fileSysEntity = new File(directory,members[row]);  
        switch(column) {  
        case 0:  
            return fileSysEntity.getName();  
        case 1:  
            if (fileSysEntity.isDirectory()) {  
                return "...";  
            }  
        }  
    }  
}
```

```

        }

        else {
            return new Long(fileSysEntity.length());
        }

        case 2:
            return fileSysEntity.isDirectory() ? new Boolean(true):
                new Boolean(false);

        default:
            return "";
    }
}

// 15. Retrieve the column names to be used in the table header.

public String getColumnName(int column) {
    switch(column) {
        case 0:
            return "Name";
        case 1:
            return "Bytes";
        case 2:
            return "Directory";
        default:
            return "";
    }
}

// 16. Retrieve the class types of the entries in
// each of the table columns.

public Class getColumnClass(int column) {
    Class returnClass = String.class;
    if (column == 2)
        returnClass = Boolean.class;
    return returnClass;
}
}

```

## 代码详析

在 TTableModel 框架的构造函数中，片段 1 为框架赋以一个标题并初始化了对其内容窗格的引用。片段 2 创建了一个标签和文本字段并将它们加到窗格上。此文本字段可以使用户通过输入一个有效目录名来查看它的内容。

片段 3 检索作为字符串的主目录，并将它赋给作为文件对象且称作 DirectoryModel 的定制模型。此定制模型由已经创建的树对象使用。createDefaultColumnsFromModel () 方法清除所有已有的列，并且基于由此表格模型的方法 getColumnCount () 和 getColumnClass ()

返回的值来创建新的列。

片段 4 显示的代码将窗格（在片段 2 中创建的）和表格加到框架上。注意，此表格是通过滚动窗格加到框架上的。片段 5 显示了与该框架对象有关的配置。片段 6 是控制关闭框架代码的事件。片段 7 是主要的方法，在这里创建了这个框架实例。片段 8 是文本字段监听程序类。当输入项生效时，此文本字段就会激活“动作”事件。在 actionPerformed() 方法里，该程序取得作为字符串的目录路径，并使用新的目录路径创建目录模型。然后，此表格被赋以重新取得的数据模型，从而显示新目录的内容。

片段 10 显示了叫做 DirectoryModel 的定制模型，它是 AbstractTableModel 的扩展类。如果从 AbstractTableModel 扩展了一个类，不必实现 TableModel 接口的所有方法。片段 11 显示了此定制模型类的构造函数。使用这个构造函数，可以将目录的拷贝存储在该类中。该类还使用叫做 members 的数组存储目录内容和该目录下的行数 (rowCount)。注意，在此片段的 else 子句中，当用户输入一个不存在的目录名时，必须将行数设为零。

片段 12 显示的方法获取了用于该表格的行数。类似地，片段 13 获取列数。如果一个目录下有某些成员，则有标题分别为 Name、Bytes 和 Directory 的三列。假如没有成员的话，其列数将为零。

片段 14 显示了一个至关紧要的 getValueAt() 方法，它返回在指定行和列中表格单元的数据值。此方法最初检查目录或它的成员是否存在。假如它们不存在，该方法就返回 null。如果它们存在，该方法就使用目录名和它在指定行的成员创建一个 File 对象。然后，它在第一列返回该目录或文件名。类似地，对重新命名的那些列，该方法视其为目录或文件而分别返回诸如文件大小和 boolean 对象这样的信息。

片段 15 获取显示在表头里的列标题。片段 16 在每一个表格列的数据项中获取类的类型。这个方法控制了显示输入项的方式。例如，如果指定数据象此代码中显示的那样具有 Boolean 类型的类，其输出结果就会伴随一个具有相应选择状态的复选框（见图 10-4）。将这个结果与图 10-3 中的结果作比较，在图 10-3 中，当程序使用缺省表格模型时，其 Boolean 值作为字符串显示。

### 10.3 表格选择

Swing 表格不仅可以用来显示数据，而且还允许使用鼠标和键盘进行数据的选择。用户本来可以选择一行、一列或一个独立单元来编辑。这些选择模型可以通过使用相应的 boolean flag 打开或关闭它们来对其进行控制。这种表格也允许执行检查或删除选择的操作。

使用下列方法可以关闭选项的背景色和前景色：

```
public void setSelectionBackground(Color selectionBackground)  
public void setSelectionForeground(Color selectionForeground)
```

以下方法控制行或列或独立单元的选择；

```
public void setRowSelectionAllowed(boolean flag)  
public void setColumnSelectionAllowed(boolean flag)  
public void setCellSelectionEnabled(boolean flag)
```

给出由一定的索引（如 `index0` 和 `index1`）指定的闭区间，使用下面的方法可以做出在区间中被选定的行或列：

```
public void setRowSelectionInterval(int index0, int index1)  
public void setColumnSelectionInterval(int index0, int index1)
```

为了从有指定索引值（index0 和 index1）的闭区间里删除一项，可以使用下面的方法：

```
public void removeRowSelectionInterval (int index0, int index1)  
public void removeColumnSelectionInterval (int index0, int index1)
```

有关支持检查选择、获得选择的索引等 API 方法，请看附录 A。

### 10.3.1 表格选择模型

为了控制选择过程，表格依赖于由接口 `ListSelectionModel` 代表的模型。这个模型在第 9 章“列表和组合框”中已经讨论过。为了得到对行选择模型的引用，可以对此表格对象调用 `getSelectionModel()` 方法。为了得到对列选择模型的引用，需要对此表格的列模型调用 `getSelectionModel()` 方法。而通过在表格对象上使用 `getColumnModel()` 方法，可以检索列模型。

由于表格依赖于 ListSelectionModel，你可以认为象 single selection、single contiguous interval selection、multiple interval selection 这样的选择模型分别由常量 SINGLE\_SELECTION、SINGLE\_INTERVAL\_SELECTION、MULTIPLE\_INTERVAL\_SELECTION 表示。为了使用单个相邻区间或多个相邻区间，需要使用 Shift 键或 Alt 键。调用 setSelectionMode() 方法，可以将其中之一赋给该选择模型，它把这些常量作为参数值。

### 10.3.2 表格选择事件和监听程序

一旦确定了一项选择，就会激活 `ListSelectionEvent` 类型的事件。这些事件需要由实现接口 `ListSelectionListener` 的对象来维护。上述实现的类为 `valueChanged()` 方法提供代码。而当作出新的选择时，就会激活这个方法。

### 10.3.3 表格选择代码示例

清单 10.4 演示了表格单元的选择。此程序创建了一个显示表格单元索引的简单表格。当用户在一个单元内单击鼠标时，在框架的底部显示选定单元的索引，可以指示选择过程。此代码的输出结果见图 10-5。

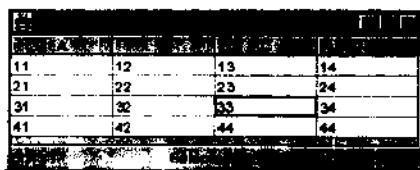


图 10-5 演示表格选择的程序输出结果

**清单 10.3 表格选择 (TTableSelection.java)**

```
// Demonstrates the implementation of table selections.  
import javax.swing.*;  
import javax.swing.table.*;  
import javax.swing.event.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class TTableSelection extends JFrame  
    implements ListSelectionListener {  
  
    Container container;  
    JTable table;  
    JScrollPane scrollPane;  
    JLabel label;  
  
    String[] columnTitles = {"A", "B", "C", "D"};  
    Object[][] rowData = {  
        {"11", "12", "13", "14"},  
        {"21", "22", "23", "24"},  
        {"31", "32", "33", "34"},  
        {"41", "42", "43", "44"}};  
  
    public TTableSelection() {  
        // 1. Assign a title to the frame and get  
        // the handle on its content pane.  
        super("TTableModel");  
        container = this.getContentPane();  
  
        // 2. Instantiate the label and table objects.  
        label = new JLabel("", JLabel.CENTER);  
        table = new JTable(rowData, columnTitles);  
  
        // 3. Registering for cell selection.  
        table.setCellSelectionEnabled(true); // selection enabled  
        // Obtain a reference to the table selection model  
        ListSelectionModel cellSelectionModel =  
            table.getSelectionModel();  
        // Assign the selection mode.  
        cellSelectionModel.setSelectionMode(  
            ListSelectionModel.SINGLE_SELECTION);  
        // Add the listener to the table selection model.  
        cellSelectionModel.addListSelectionListener(this);  
  
        // 4. Add the panel and table to the container.  
        container.add(BorderLayout.SOUTH, label);  
        container.add(new JScrollPane(table));  
    }  
  
    public void valueChanged(ListSelectionEvent e) {  
        if (!e.getValueIsAdjusting()) {  
            int row = table.getSelectedRow();  
            int col = table.getSelectedColumn();  
            label.setText("Row: " + row + ", Column: " + col);  
        }  
    }  
}
```

```
// 5. Frame settings.  
// Add the window closing listener.  
addWindowListener(new WindowEventHandler());  
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
setBackground(Color.white);  
setSize(350,150); // Frame width = 350, height = 150  
show(); // Display the frame  
}  
  
// 6. Window event handler.  
class WindowEventHandler extends WindowAdapter {  
    public void windowClosing(WindowEvent evt) {  
        System.exit(0);  
    }  
}  
  
// 7. List selection listenter method to be implemented.  
public void valueChanged(ListSelectionEvent e) {  
    int rows,columns;  
    String selectedData = null;  
    int[] selectedRow = table.getSelectedRows();  
    int[] selectedColumns = table.getSelectedColumns();  
    for (int i = 0; i < selectedRow.length; i++) {  
        for (int j = 0; j < selectedColumns.length; j++) {  
            selectedData = (String) table.getValueAt(  
                selectedRow[i],selectedColumns[j]);  
        }  
    }  
    label.setText("Selected: " + selectedData);  
}  
  
// 8. The main method.  
public static void main(String[] args) {  
    TTableSelection frame = new TTableSelection();  
}
```

}

## 代码详析

在此框架的构造函数里，片段 1 赋值一个标题并获得框架内容窗格的一个句柄。片段 2 例示了标签和表格对象。片段 3 显示了启动选择过程的代码，并且注册了一个监听程序来接受此选择事件。为了注册此选择监听程序，需要获得对表格选择模型的引用。片段 4 向框架加入了标签和表格。

片段 5 显示了构造该框架的代码；片段 6 是一个处理关闭窗口的窗口事件；片段 7 是已

经在列表选择模型中实现的方法；当做出表格选择时，需要执行此代码。片段 8 是创建主框架的主方法。

## 10.4 编辑表格单元

除了显示数据以外，Swing 表格还支持对数据的编辑。这个特性对将数据存储在数据库中极其有用。Swing 表格更加高级化，以便能够附加上其他的数据编辑组件，如组合框、列表等等。也可以使用复选框来编辑属性的 boolean 状态（true 或 false）。在缺省状态下，Swing 表格使用文本字段作为它的编辑器。

为了编辑表格，需要安排 TableModel 来相应地执行 isCellEditable() 方法。这个方法对给定的可编辑单元一定返回 true 值。另外，还需要执行 setValueAt() 方法，它在一个单元值已经改变并且需要确认时调用。最后，当编辑了新的数据项时，该表格模型就会通知它的监听程序去修改此表格的视图。

### 10.4.1 可编辑表格的代码示例

清单 10.5 是一个演示可编辑表格的应用程序。该程序基本上创建了一个包含列头的表格，在它的列头里显示了一些人的姓名、体重、血型和年龄段。在缺省状态下，此表格出现时就具有程序中介绍的这些数据。图 10-6 显示了由这个程序产生的可编辑表格。

在这个表格单元里，可以对每个人的数据进行编辑。注意，可以选择血型，并且利用组合框将其编辑到表格中，此组合框与血型栏中的每一个单元相匹配。表格的最后一列显示了用来编辑相应布尔值的复选框。

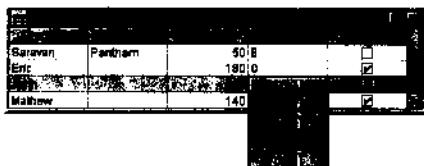


图 10-6 演示表格选择的程序输出结果

清单 10.3 可编辑的表格 (*TEditableTable.java*)

```
// Demonstrates the editable property of Swing tables.

import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

public class TEditableTable extends JFrame {
    Container container;
    JTable table;
    JScrollPane scrollPane;
```

```
JLabel label;
JTextField textField;

public TEdditableTable() {
    // 1. Assign a title to the frame and get
    // the handle on the content pane.
    super("TEdditableTable");
    container = this.getContentPane();

    // 2. Define the data to be displayed in the table.
    String[] columnTitles = {
        "First Name", "Last Name", "Weight (lb)", "Blood Group", "Age > 20yrs"};
    Object[][] dataEntries = {
        {"Saravan", "Panthal", new Integer(50), "B", new Boolean(false)},
        {"Eric", "", new Integer(180), "O", new Boolean(true)},
        {"John", "", new Integer(120), "AB", new Boolean(false)},
        {"Mathew", "", new Integer(140), "A", new Boolean(true)},
    };

    // 3. Assign the table model object to a table object.
    TableModel model = new EditableTableModel(columnTitles, dataEntries);
    table = new JTable(model);
    table.createDefaultColumnsFromModel();

    // 4. Create a combo box and use it as an editor for the
    // blood group column.
    String[] bloodGroups = {"A", "B", "AB", "O"};
    JComboBox comboBox = new JComboBox(bloodGroups);
    table.getColumnModel().getColumn(3).setCellEditor(
        new DefaultCellEditor(comboBox));

    // 5. Add the above panel and table to the container.
    container.add(new JScrollPane(table));

    // 6. Frame settings.
    // Add the window closing listener.
    addWindowListener(new WindowEventHandler());
    setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    setBackground(Color.white);
    setSize(450, 150); // Frame width = 450, height = 150
    show(); // Display the frame
}

// 7. Window event handler.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}
```

```
}

// 8. The main method.
public static void main(String[] args) {
    TEditableTable frame = new TEditableTable();
}

}

// 

class EditableTableModel extends AbstractTableModel {
    String[] columnTitles;
    Object[][] dataEntries;
    int rowCount;

    // 9. Model constructor.
    public EditableTableModel(String[] columnTitles, Object[][] dataEntries) {
        this.columnTitles = columnTitles;
        this.dataEntries = dataEntries;
    }

    // 10. Retrieve the number of rows for the table to be prepared.
    public int getRowCount() {
        return dataEntries.length;
    }

    // 11. Similarly, retrieve the column count.
    public int getColumnCount() {
        return columnTitles.length;
    }

    // 12. Retrieve each of the table values at the specified
    // row and column.
    public Object getValueAt(int row, int column) {
        return dataEntries[row][column];
    }

    // 13. Retrieve the column names to be used in the table header.
    public String getColumnName(int column) {
        return columnTitles[column];
    }

    // 14. Retrieve the class types of the entries in
    // each of the table columns.
    public Class getColumnClass(int column) {
        return getValueAt(0, column).getClass();
    }

    // 15. Allow all the table cells to be editable.
    public boolean isCellEditable(int row, int column) {
```

```
    return true;
}

// 16. Assign the data entered to the data model.
public void setValueAt(Object value, int row, int column) {
    dataEntries[row][column] = value;
}
}
```

## 代码详析

在 TEditableTable 框架的构造函数里，片段 1 赋值一个标题并获得框架内容窗格的句柄；片段 2 定义了显示在表格单元中的缺省数据；片段 3 将此表格模型赋给已经创建的表格对象；此表格模型执行的方法使表格成为可编辑的；片段 5 将此表格加到主框架的内容窗格上；片段 6 构造了主框架，片段 7 是控制关闭窗口的事件；随后的片段 8 显示了主方法。

# 第 11 章 文本小配件

随着 Swing 的出现，文本组件的地位超过了文本字段和文本区域组件。Swing 库引入了一个叫做口令字段的更简单的组件，还引入了两个处理样式文本的高级组件编辑器窗格和文本窗格。尽管处理样式文本是复杂的，但是 Swing 库通过将复杂功能打包的办法使得一般任务更易于完成。

Swing 库提供了一种理解性的框架，可以通过图形用户接口（GUI）来处理几乎所有的实际文本操作。现在你可以很容易地处理单行纯文本、多行纯文本和多行样式文本。文本字段支持单行纯文本，而口令字段则意味着输入的秘密信息不能在文本字段中直接看到。文本区域组件支持多行纯文本，同时编辑器窗格和文本窗格可支持多行样式文本的编辑处理。除了这些组件以外，Swing 库还有大量支持类和接口的组件，用来处理信息模型、色彩、样式、事件等等。

## 11.1 JTextFieldComponent 类

JTextComponent 类是 Swing 中文本组件的父类。因此，它给文本组件提供了大量的 API 方法组件，特别是关于组件文档模型和组件内容的方法，以及关于显示和控制的方法。JTextComponent 类还提供 API 调用用来处理恢复或取消恢复操作、键盘映射、绑定以及光标的变化。有关这些 API 调用的理解性列表请见附录 A：“JFC Swing 快速参考”。

## 11.2 文档接口

文档是 Swing 文本组件的内容模型，即文本组件利用文档的对象来存储数据。文档是由名为 Document 的设计级接口来描述的。这个接口保存在名为 javax.swing.text 的程序包里。

组件的信息内容存储在文档中作为它的元素。这些元素实现了接口元素（Element），它们保存在程序包 javax.swing.text 中。每个这样的元素都拥有诸如字体、大小、颜色等描述它们风格的属性。

## 11.3 文本字段

Swing 文本字段可以用来显示或编辑一个单行纯文本。该组件类似于 AWT 文本字段；不过相对来说 Swing 文本字段是一个轻量级的组件。一个文本字段对象通过使用 JTextComponent 的一个子类——JTextField 类来创建的。因此，JTextField 的功能就扩展到 JTextComponent 类和 JComponent 类。JTextField 能够激活动作和由监听程序捕获的鼠标事件。

### 11.3.1 JTextField 构造函数

可以使用 JTextField 支持的具有不同参数的构造函数，来创建一个 JTextField 类型的对象。整型参数（int type）意味着指定该文本字段的行数或字段宽度。字符串类型（String type）的参数初始化带有指定文本的文本字段，并且给出显示缺省文本字符串的方法。文档类型的参数（Document type）为文本存储器指定了内容模型。下面是 JTextField 类所支持的构造函数列表：

```
public JTextField();
public JTextField(String text);
public JTextField(int fieldwidth);
public JTextField(String text, int fieldwidth);
public JTextField(Document docModel, String text, int fieldwidth);
```

### 11.3.2 JTextField 代码示例

清单 11.1 列出的是一个示例程序 TJTextField.java，它演示了一个 Swing 文本字段在程序中的执行过程。你可以在上面的文本字段里编辑任何文本，然后按回车键，输入的文本就会显示在下面的文本字段里。如果单击 Clear 按钮，则会清空文本字段。该程序的输出结果如图 11-1 所示。

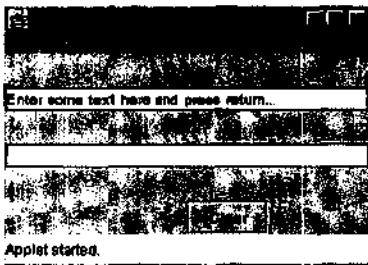


图 11-1 显示 Swing 文本字段的小程序

清单 11.1 JTextField 组件和一个清除按钮 (TJTextField.java)

```
// Demonstrates Swing text fields.

/*
 * <Applet code = TJTextField width = 300 height = 150>
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class TTTextField extends JApplet {
    Container container;
    JTextField textField1, textField2;
    public void init() {
        // 1. Get the handle on the applet's content pane.
        container = this.getContentPane();
        // 2. Prepare the applet with vertical box layout.
        container.setLayout(new BoxLayout(container, BoxLayout.Y_AXIS));
        // 3. Create two text - fields and a button.
        textField1 = new JTextField(
            "Enter some text here and press return", // Initial string
            20); // 20 columns width
        // Add an action listener to this text field.
        textField1.addActionListener(new TextFieldListener());
        // Assign a line border with black color.
        textField1.setBorder(BorderFactory.createLineBorder(Color.black));
        textField2 = new JTextField(20); // 20 columns width.
        // Assign a line border with blue color.
        textField2.setBorder(BorderFactory.createLineBorder(Color.blue));
        JButton button = new JButton("Clear");
        // Add an action listener to this button.
        button.addActionListener(new ButtonListener());
        // 4. Add text - fields and button to the applet's content pane.
        container.add(Box.createVerticalGlue()); // Add a glue component.
        container.add(textField1); // Add the textField1.
        container.add(Box.createVerticalGlue()); // Add another glue component.
        container.add(textField2); // add the textField2
        container.add(Box.createVerticalGlue()); // Add another glue component.
        container.add(button); // Add the button.
    }
    // 5. The text field listener class.
    class TextFieldListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            // Retrieve the text entered in textField1
            // and assign it the other textField.
            textField2.setText(e.getActionCommand());
        }
    }
    // 6. The button listener class.
    class ButtonListener implements ActionListener {
```

```
public void actionPerformed(ActionEvent e) {  
    // Clear both the text fields.  
    textField1.setText("");  
    textField1.requestFocus(); // Get the focus back  
    textField2.setText("");  
}  
}  
}
```

代码详析

在 init() 方法内，代码片段 - 1 为小程序的内容窗格得到一个句柄。片段 - 2 为小程序设计了一个垂直的框式布局。创建两个文本字段和一个按钮的代码在片段 - 3 中。注意，textfield1 是通过名为 TextFieldListener 的动作监听程序注册的。Clear 按钮也是通过一个名为 ButtonListener 的动作监听程序注册的。而片段 - 4 把这些组件加到小程序的内容窗格里。

片段 - 5 显示了监听程序 `TextFieldListener` 类，它实现了 `ActionListener` 接口。当焦点位于一个文本字段上面时按回车键，该文本字段就会激活一个动作事件。这个事件可以象片段 - 5 中所显示的那样去处理。

在 actionPerformed() 方法内，来自 textfield1 的文本被设计为 textfield2。而方法 getActionCommand() 返回该文本字段的输入项，因为该文本字段没有通过方法 setActionCommand() 被设计任何动作命令。方法 setText() 为 textfield2 设计了找回的文本。片段 - 6 显示的是按钮监听程序的代码。这段监听程序显示了单击 Clear 按钮时清除文本字段的代码。

#### 11.4 日令字段

口令字段组件类似于文本字段，我们可以在其中不显示原有特征的情况下编辑单行文本；但是对文本字段内每一个键入的字符都会显示一个缺省的或指定的返回字符。顾名思义，可以为键入对话框中的任何加密口令加上这个组件。

口令字段对象是通过程序包 javax.swing 中提供的 JPasswordField 类来创建的。JPasswordField 是 JTextField 的直接子类。JPasswordField 内部的构造函数非常接近于 JTextField 的构造函数。下面是 JPasswordField 内的构造函数列表：

```
public JPasswordField();
public JPasswordField(String text);
public JPasswordField(int fieldWidth);
public JPasswordField(String text, int fieldwidth);
public JPasswordField(Document docModel, String text,
                     int fieldwidth);
```

正如前面看到的 JTextField 构造函数，参数 fieldWidth 表示为该口令字段所设计的行数。事实上，这个参数的值为 6。字符串类型的参数是为缺省使用情况设计的秘密文本。而文档

类型参数 (Document) 是用于该口令字段的文本文档的模型。

#### 11.4.1 JPasswordField 代码示例

清单 11.2 给出了一个小程序的示例，它包含一个输入加密文本的口令字段和一个有关用户名的文本字段。当用户输入一个口令时，在口令字段中仅显示“\*”。这个字符可选择使用缺省字符“\*”。该程序的输出结果显示在图 11-2 中。

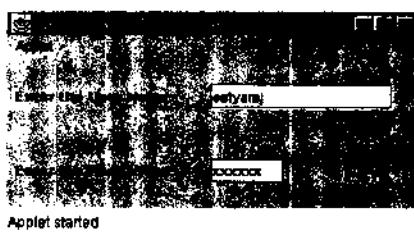


图 11-2 一个显示文本字段和口令字段的 Swing 小程序

清单 11.2 JPasswordField 和 JTextField 组件 (TJPasswordField.java)

```
// Demonstrates Swing password fields.

/*
 * <Applet code = TJPasswordField width = 350 height = 125 >
 * </Applet >
 */

import javax.swing.*;
import java.awt.*;

public class TJPasswordField extends JApplet {
    Container container;

    public void init() {
        // 1. Get the handle on the applet's content pane.
        container = this.getContentPane();

        // 2. Set the applet background color and layout.
        container.setBackground(Color.lightGray);
        GridBagLayout gridbag = new GridBagLayout();
        container.setLayout(gridbag);

        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(5,5,5,5); // top, left, bottom, right
        c.weightx = 1; c.weighty = 1;

        // 3. Create the component objects
        // and add them to the applet using grid bag layout.
        c.fill = c.BOTH;
        c.gridx = 0; c.gridy = 0;
```

```
c.gridx = 1; c.gridy = 1;
JLabel label1 = new JLabel("Enter the Username: *");
gridbag.setConstraints(label1,c);
container.add(label1);

c.gridx = 0; c.gridy = 1;
JLabel label2 = new JLabel("Enter the Pass - Word: ");
gridbag.setConstraints(label2,c);
container.add(label2);

c.fill = c.NONE;
c.anchor = c.WEST;
c.gridx = 1; c.gridy = 0;
JTextField textField = new JTextField("satyaraj",15);
gridbag.setConstraints(textField,c);
container.add(textField);

c.gridx = 1; c.gridy = 1;
JPasswordField passwordField = new JPasswordField(6);
passwordField.setBorder(BorderFactory.createLineBorder(Color.red));
gridbag.setConstraints(passwordField,c);
container.add(passwordField);

// 4. Set an echo character for the password field.
// Note: The default character is "*".
passwordField.setEchoChar('x');
}

}
```

## 代码详析

在清单 11.2 中，代码片段 - 1 得到该小程序的内容窗格的引用。片段 - 2 将小程序的布局设计为无序网格状，并且创建了一个名为 textField 的 JTextField 对象和一个名为 passwordField 的口令字段对象。名为 label1 和 label2 的两个 JLabel 对象指明了文本字段和口令字段组件的目的。这些组件在小程序中的定位则显示在片段 - 3 中。

文本字段显示缺省的文本 satyaraj，它是缺省情况下显示的用户名。如果必要，用户名可以改变。口令字段的宽度是 6 个字符，并且最初是空的。注意，在口令字段内每键入一个字符，在其显示域内都会返回一个 x 字符。这可以通过在对象 passwordField 中调用 setEchoChar (char c) 来实现。如果没有特殊选择返回字符，那么缺省字符 \* 就会被返回。片段 - 4 显示了改变返回字符的代码。

## 11.5 文本区域

相对 AWT 文本区域组件来说，Swing 文本区域是一个简单选择方式。使用文本区域可以显示或编辑多行纯文本。文本区域也可以限制用户的读写权限。假如一个内容模型（它是

用来实现 Document 接口的) 已经是有效的, 还可以将这个模型设计为文本区域对象。

任何一个 Swing 文本区域组件都是一个 JTextArea 类型的对象。JTextArea 类是 JTextComponent 的一个子类, 因而从它的父类那里继承了一个重要功能。在使用文本区域组件的过程中, 可能经常要用到 JTextComponent 和 JComponent 类中的这个方法。

### 11.5.1 JTextArea 构造函数

为实例化一个 JTextArea 对象, 你可以使用下列任何一个带可选参数的构造函数:

```
public JTextArea();
public JTextArea(String text);
public JTextArea(int rows, int columns);
public JTextArea(String text, int rows, int columns);
public JTextArea(Document doc);
public JTextArea(Document doc, String text,
                int rows, int columns);
```

String 类型的参数 text 意味着最初的文本信息显示在文本区域内。构造函数里的参数 rows 指的是文本行数; 而参数 columns 指的是文本列数。

参数 Document 提供了一种为文本区域指定文档模型的方法。Document 是设计级接口, 需要用依赖于选定的文档模型的类来实现。如果没有指定文档模型, 则组件会自动选择缺省模型。在程序后面任何地方, 都可以使用 JComponent 中的 setDocument (Document docModel) 方法来设计文档模型。

JTextArea 包括一系列 “set” 方法, 如 setColumns (int columns)、setFont (Font font)、setRow (int rows)、setTabSize (int tabSize) 和与之对应的 “get” 方法。这些 “get” 方法将返回具有不同属性的值。

方法 append (String text) 将参数字符串文本追加到文本区域中当前文本的末端, insert (String text, int position) 是把文本插入到指定的列。还可以用方法 replaceRange (String text, int startPosition, int endPosition) 来替换文本区域内已经存在的任何文本。除了刚刚讨论过的这些方法外, 为了达到编辑的目的, JTextArea 还使用了 JTextComponent 类中的大量方法。

### 11.5.2 光标事件和监听程序

光标一旦改变位置或选中某段文本, 那么文本组件就会激活一个光标事件。CaretEvent 类支持 getDot () 方法和 getMark () 方法, 它们分别找到当前位置和选中文本的末端位置。对这个事件的所有监听程序对象都必须执行 CaretListener 接口监听程序。该接口需要由 caretUpdate (CaretEvent e) 来实现。光标位置一改变此方法就会被调用。

### 11.5.3 JTextArea 代码示例

清单 11.3 是一个演示 JTextArea 类中 API 方法的示例程序。文本区域的内容可以通过使用文本字段和显示的按钮来生成。在文本字段中可以键入任何文本然后按 Insert 按钮将其插入光标位置; 也可以选中一段文本, 再单击 Delete 按钮去删除它。其余按钮用于剪切、复

制和粘贴等操作。比如通过在文本上拖动鼠标选中一段文本，然后单击 Cut 按钮来剪切选中的文本，这个文本会被放在系统剪贴板上。单击 Copy 按钮也能达到相似的目的，只是文本不会被剪切掉。当单击 Paste 按钮时，剪贴板上的文本会被粘贴到光标位置。该程序的输出结果显示在图 11-3 中。

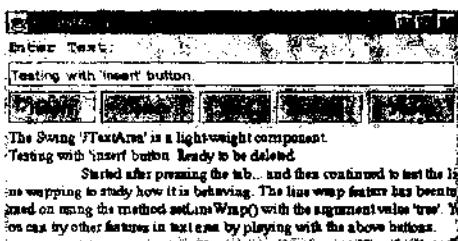


图 11-3 Swing 文本区域

### 清单 11.3 JTextArea 和编辑按钮 (TJTextArea.java)

```
// Demonstrates the Swing text area.

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TJTextArea extends JFrame {
    // 1. Declare the references for the following objects.
    Container container;
    JLabel label = null;
    JTextField textField = null;
    JTextArea textArea = null;
    JButton insertButton = null;
    JButton deleteButton = null;
    JButton cutButton = null;
    JButton copyButton = null;
    JButton pasteButton = null;

    public TJTextArea() {
        // 2. Assign a name to the frame and obtain a handle
        // on the frame's content pane.
        super("TJTextArea");
        container = this.getContentPane();
        // 3. Create the fonts for label and text area.
        Font labelFont = new Font("SanSerif", Font.BOLD, 14);
        Font textFont = new Font("Dialog", Font.PLAIN, 12);
        // 4. Use the gridbag layout for the applet.
        GridBagLayout gridbag = new GridBagLayout();
        container.setLayout(gridbag);
```

```
GridBagConstraints c = new GridBagConstraints();

// 5. Add a label.
c.fill = c.BOTH;
c.insets = new Insets(2,2,2,2);
c.gridx = 0; c.gridy = 0;
c.gridwidth = 5; c.gridheight = 1;
c.anchor = c.WEST;
c.weightx = 1.0; c.weighty = 1.0;
label = new JLabel("Enter Text:");
label.setFont(labelFont);
gridbag.setConstraints(label,c);
container.add(label); // add the label

// 6. Add a text field.
c.anchor = c.CENTER;
c.gridx = 0; c.gridy = 1;
textField = new JTextField();
gridbag.setConstraints(textField,c);
container.add(textField); // add the text - field

// 7. Add the Insert button.
c.gridx = 0; c.gridy = 2;
c.gridwidth = 1; c.gridheight = 1;
c.fill = c.BOTH;
insertButton = new JButton("Insert");
insertButton.setBackground(Color.lightGray);
gridbag.setConstraints(insertButton,c);
container.add(insertButton); // add insert button
ButtonListener inButtonListener = new ButtonListener();
insertButton.addActionListener(inButtonListener);

// 8. Add the Delete button.
c.gridx = 1; c.gridy = 2;
deleteButton = new JButton("Delete");
deleteButton.setBackground(Color.lightGray);
gridbag.setConstraints(deleteButton,c);
container.add(deleteButton); // add the delete button
ButtonListener dlButtonListener = new ButtonListener();
deleteButton.addActionListener(dlButtonListener);

// 9. Add the Cut button.
c.gridx = 2; c.gridy = 2;
cutButton = new JButton("Cut");
cutButton.setBackground(Color.lightGray);
gridbag.setConstraints(cutButton,c);
```

```
container.add(cutButton); // add the cut button
ButtonListener ctButtonListener = new ButtonListener();
cutButton.addActionListener(ctButtonListener);

// 10. Add the Copy button.
c.gridx = 3; c.gridy = 2;
copyButton = new JButton("Copy");
copyButton.setBackground(Color.lightGray);
gridbag.setConstraints(copyButton,c);
container.add(copyButton); // add the copy button
ButtonListener cpButtonListener = new ButtonListener();
copyButton.addActionListener(cpButtonListener);

// 11. Add the Paste button.
c.gridx = 4; c.gridy = 2;
pasteButton = new JButton("Paste");
pasteButton.setBackground(Color.lightGray);
gridbag.setConstraints(pasteButton,c);
container.add(pasteButton); // add the paste button
ButtonListener psButtonListener = new ButtonListener();
pasteButton.addActionListener(psButtonListener);

// 12. Add the text area.
c.gridx = 0; c.gridy = 3;
c.gridwidth = 5; c.gridheight = 1;
c.weightx = 1.0; c.weighty = 1.0;
c.anchor = c.CENTER;
c.fill = c.BOTH;
String someText = "The Swing 'JTextArea' is a "
    "light-weight component.";
textArea = new JTextArea(someText, // To be displayed initially
6 // Number of rows,
30); // Number of columns.
textArea.setFont(textFont);
textArea.setBackground(Color.white);
textArea.setSelectionColor(Color.yellow);
textArea.setTabSize(5); // The tab size.
textArea.setLineWrap(true); // Wrap the line at the container end.
gridbag.setConstraints(textArea,c);
container.add(textArea); // Add the text area.

// 13. Add the window listener to close the frame
// and display it with the specified size.
addWindowListener(new WindowEventHandler());
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
setSize(400,200); // width = 400, height = 200
```

```
show(); // Display the frame.  
}  
  
// 14. Button listener class.  
class ButtonListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        JButton b = (JButton) e.getSource();  
        if (b == insertButton) {  
            try {  
                // Insert text at the caret position. The text is  
                // is retrieved from the text field by using getText().  
                textArea.insert(textField.getText(),  
                                textArea.getCaretPosition());  
            }  
            catch (IllegalArgumentException excep) {  
            }  
        }  
        else if (b == deleteButton) {  
            textArea.replaceSelection(""); // Selected text is deleted.  
        }  
        else if (b == cutButton) {  
            textArea.cut(); // Cut operation.  
        }  
        else if (b == copyButton) {  
            textArea.copy(); // Copy operation.  
        }  
        else if (b == pasteButton) {  
            textArea.paste(); // Paste operation.  
        }  
    }  
}  
  
// 15. The listener class to handle closing of the frame.  
class WindowEventHandler extends WindowAdapter {  
    public void windowClosing(WindowEvent evt) {  
        System.exit(0);  
    }  
}  
  
// 16. The main method.  
public static void main(String args) {  
    new TUTextArea();  
}
```

## 代码详析

代码片段 - 1 显示了用于框架类的一系列数据成员的声明语句。在 JTextArea 框架的构造函数里，片段 - 2 为框架指定了一个标题并得到它的内容窗格一个引用。片段 - 3 定义了标签上和文本区域内使用的文本字体。

片段 - 4 创建了无序网格布局的对象，并通过它的内容窗格将其指定为框架，同时它还创建了一个强迫的无序网格布局的对象。片段 - 5 把文本字段加到框架的顶端。片段 - 6 把文本字段加到标签的下面。片段 7 - 11 则分别加上了用于插入、删除、剪切复制和粘贴操作的按钮。每个这样的按钮都是通过一个动作监听程序来注册的。

片段 - 12 显示了将文本区域加到框架底部的代码。文本区域对象用显示最初文本的构造函数来创建，其中文本的行数和列数作为它的参数值。该代码中其他一些有趣的方法是 setTabSize () 和 setLineWrap ()。在按下 Tab 键并且直线包住容器的末端时，这些方法能控制光标的移动。片段 - 13 显示了配置框架的代码。

片段 - 14 是一个动作监听程序，当操作附加在框架上的某个按钮时，它就会激活其功能。片段 15 和片段 16 为关闭框架和创建框架的主要方法的事件处理程序。

## 11.6 编辑器工具箱

文本组件使用编辑器工具箱去处理和组织诸如剪切、复制和粘贴等动作。这样就可以依赖于工具箱而无须为各基本操作而编程。编辑器工具箱还具有识别文档格式（如纯文本类型和 HTML 类型）的能力。Swing 文本支持提供诸如的 DefaultEditorKit、StyledEditorKit 和 HTMLEditorKit 编辑器工具箱。DefaultEditorKit 是 StyledEditorKit 的父类并且管理纯文本的编辑动作。StyledEditorKit 管理样式文本的编辑动作。HTMLEditorKit 为 HTML 语言编辑组织动作。这个类是由 StyledEditorKit 生成的。

## 11.7 编辑器窗格

Swing 编辑器窗格用来编辑或显示诸如 HTML、RTF 等数据内容。编辑器窗格利用 EditorKit 类型的编辑器工具箱来得到编辑资源。编辑器工具箱也决定了在编辑器窗格中要编辑内容的类型。因此，处理特定类型信息的编辑器窗格就需要一个匹配的编辑器工具箱。

Swing 编辑器窗格对象由 JEditorPane 类代表，它扩展了 JTextComponent。这样就能够使用 JTextComponent 所支持的核心功能。JEditorPane 和 JTextComponent 类分别存储在 javax.swing 和 javax.swing.text 程序包里。

### 11.7.1 JEditorPane 构造函数

JEditorPane 类支持三个构造函数，以创建其对象。使用 URL 的字符串形式（String 类型）或直接指定 URL 对象自身，可指定被显示的信息内容。

如果使用的构造函数在创建窗格或编辑窗格中无任何内容，则会利用 setPage () 方法来指定信息。setPage () 已存在两个可重载框架，它可用于 URL 的字符串框架或 URL 对

象本身。

下面是 JEditorPane 类的构造函数：

```
public JEditorPane()
```

创建了一个 JEditorPane 对象，它的文档模型置为空。

```
public JEditorPane(URL initialPage)
    throws IOException
```

创建了一个 JEditorPane 对象，该对象基于内容的 URL 对象。如果构造函数是空的或不可存取的，构造函数就返回一个异常事件。

```
public JEditorPane(String url)
    throws IOException
```

创建了一个 JEditorPane 对象，该对象基于输入字符串所指定的 URL。正如前面的构造函数一样，需要捕获一个输入/输出异常事件。

### 11.7.2 JEditorPane 代码示例

程序清单 11.4 是一个示例程序，它实现了 JEditorPane 方法，以显示由给定 URL 指定的 HTML 内容。当你可以在文本字段内键入 URL 字符串后，该程序可取到相应的信息内容并把它显示在编辑器窗格内。所以这是一个简化版的浏览器程序。该程序的输出结果显示在图 11-4 中。

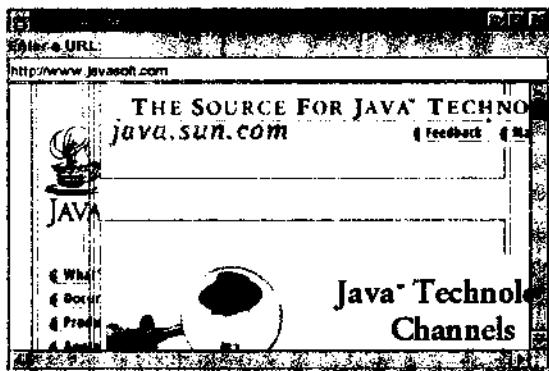


图 11-4 一个 Swing 编辑器窗格的显示实例

清单 11.4 用 JEditorPane 显示一个 Web 主页 (JEditorPane.java)

```
// Demonstrates the Swing editor pane.

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
```

```
public class TJEditorPane extends JFrame {  
    JTextField textField;  
    JEditorPane editorPane;  
  
    public TJEditorPane () {  
        // 1. Assign a title to the frame and obtain  
        // a reference to the frame's content pane.  
        super("TJEditorPane");  
        Container container = this.getContentPane();  
  
        // 2. Create a label and attach it to a panel.  
        JLabel label = new JLabel("Enter a URL: ");  
        JPanel panel = new JPanel(new GridLayout(2,1));  
        panel.add(label);  
  
        // 3. Create a text field and add it to the panel.  
        textField = new JTextField();  
        textField.addActionListener(new TextFieldListener());  
        panel.add(textField);  
  
        // 4. Add the panel to the content pane.  
        container.add(panel, BorderLayout.NORTH);  
  
        // 5. Create an editor pane and add it to the content pane  
        // through a scroll pane.  
        editorPane = new JEditorPane();  
        JScrollPane scrollPane = new JScrollPane(editorPane);  
        container.add(scrollPane);  
  
        // 6. Add the window listener to close the frame  
        // and display it with the specified size.  
        addWindowListener(new WindowEventHandler());  
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
        setSize(350, 200); // width = 350, height = 200  
        setBackground(Color.lightGray);  
        setForeground(Color.black);  
        show(); // Display the frame  
  
        // 7. Get the focus to textField.  
        textField.requestFocus();  
    }  
  
    // 8. The text field listener class.  
    class TextFieldListener implements ActionListener {  
        public void actionPerformed(ActionEvent ae) {  
            URL pageURL = null;  
  
            // 9. Create the URL object for the text that has been  
            // entered in the text field.  
        }  
    }  
}
```

```
try {
    pageURL = new URL(textField.getText());
} catch (MalformedURLException me) {
    System.out.println("MalformedURLException");
}

// 10. Assign the page to the editor pane so that the
// page is displayed.
try {
    editorPane.setPage(pageURL);
} catch (java.io.IOException ioe) {
    System.out.println("IOException while loading page");
}
}

// 11. The main method.
public static void main(String[] args) {
    new TJEditorPane();
}

// 12. The listener class to handle closing of the frame.
class WindowEventHandler extends WindowAdapter {
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
}
}
```

## 代码详析

在 JEditorPane 框架的构造函数里，片段 - 1 为框架指定了一个标题并得到内容窗格框架的引用。片段 - 2 创建了一个标签并把它附加到窗格对象上。片段 - 3 创建了一个文本字段对象并把它附加到该窗格上。片段 - 4 把该窗格附加到内容窗格上。

片段 - 5 创建了一个编辑器窗格对象并通过滚动窗格把它连到内容窗格上。片段 - 6 显示了配置框架并显示它的代码。片段 - 7 把焦点移到在主框架中已初始化过的文本字段上。

片段 - 8 显示了文本字段监听程序类。在方法 actionPerformed () 里，片段 - 9 从原位置指定的字符串创建了一个 URL 对象，指定字符串来自于文本字段。片段 - 10 显示了为编辑器窗格设计 URL 对象的代码，该窗格用来下载或显示信息。

## 11.8 文本窗格

文本窗格是支持样式文本的轻量级组件。用它可以显示各种样式风格和段落对齐方式的多行文本，还可以显示图标及其他用户接口组件。文本窗格可以将图像嵌入到文本中。这

样，该组件就能建立包含大量显示文本和图形特性的高级字处理程序。

为了显示各种格式的文本，需要将（字符或段落的）样式连到属性集，并在其后设置文本窗格的属性。也可以使用 `setCharacterAttributes (AttributeSet asset, boolean b)` 和 `setParagraphAttributes (AttributeSet asset, boolean b)` 方法为文本窗格设置字符和段落属性。其中 `boolean` 的值决定当前存在的属性是否完全被替换，可以为 `true` 或 `false`。

文本窗格由 `JTextPane` 类代表，它是 `JEditorPane` 的子类。这个类存储在程序包 `javax.swing` 内。

### 11.8.1 `JTextPane` 构造函数

为了创建 `JTextPane` 的实例，可以使用两种构造函数中的任何一个。其一是不带参数的构造函数，它缺省地指向 `StyleEditorKit`。在此例中，不存在可直接快速使用的文档模型，需要在程序中设计一个 `Document` 类型的文档模型。另一个构造函数允许指定文档模型。下面是 `JTextPane` 支持的两个构造函数：

```
public JTextPane()  
public JTextPane() (StyleDocument doc)
```

### 11.8.2 撤消和恢复

为了提供撤消和恢复特性，需要实例化 Swing 的 `UndoManager` 类。这个对象可控制发生在文本组件中文本的编辑操作。该文档激活 `UndoableEditEvent` 类型的事件，此事件被已注册的监听程序所监视。

监听程序类必须执行 `UndoableEditListener` 接口并为 `UndoableEditHappened ()` 方法提供代码。在这个方法里，可以调用 `addEdit ()` 方法为 Undo Manager 增加一个编辑功能。也可以通过调用已激活的编辑事件上的 `getEdit ()` 方法得到“恢复”操作的功能。

### 11.8.3 `JTextPane` 代码示例

表 11.5 是用 `JTextPane` 显示一个样式文本的示例程序。其中带格式的文本包括一个鸭子 (Java Duke) 图标和一个带文本标语。此程序的输出结果显示在图 11-5 中。

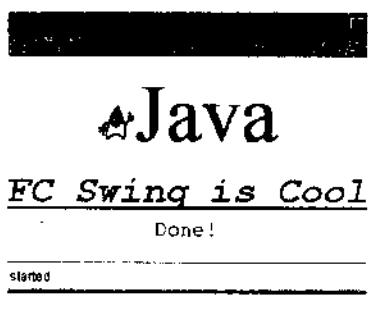


图 11-5 一个 Swing 文本窗格的演示

**清单 11.5 用 JTextPane 显示样式文本 (JTextPane.java)**

```
// TJTextPane.java/* * <Applet code = TJTextPane.class width = 400 height = 300>
* </Applet>
*/
import javax.swing.*;
import javax.swing.text.*;
import java.awt.*;

public class TJTextPane extends JApplet {
    JTextPane textPane = null;
    MutableAttributeSet centerAlign;
    MutableAttributeSet charStyle1,charStyle2,charStyle3;

    public void init() {
        // 1. Create a text pane object.
        textPane = new JTextPane();

        // 2. Prepare the paragraph attribute for center alignment.
        centerAlign = new SimpleAttributeSet();
        StyleConstants.setAlignment(centerAlign,
            StyleConstants.ALIGN_CENTER);

        // 3. Prepare the character attribute set called style1.
        charStyle1 = new SimpleAttributeSet();
        StyleConstants.setFontFamily(charStyle1,"Serif");
        StyleConstants.setFontSize(charStyle1,75);

        // 4. Prepare another character attribute set called style2.
        charStyle2 = new SimpleAttributeSet();
        StyleConstants.setFontSize(charStyle2,35);
        StyleConstants.setUnderline(charStyle2,true);
        StyleConstants.setForeground(charStyle2,color.red);
        StyleConstants.setBold(charStyle2,true);
        StyleConstants.setItalic(charStyle2,true);

        // 5. Prepare one more character attribute set called style3.
        charStyle3 = new SimpleAttributeSet();
        StyleConstants.setFontSize(charStyle3,20);

        // 6. Get a reference to the content model of the text pane.
        Document contentModel = textPane.getDocument();

        try {
            // 7. Assign a position to the caret.
            textPane.setCaretPosition(contentModel.getLength());
            // Insert the Java Duke icon.
            textPane.insertIcon(new ImageIcon("javaIcon1.gif"));
        }
    }
}
```

```

// 8. Assign character attributes to the text pane.
textPane.setCharacterAttributes(charStyle1, false);
// Assign paragraph attributes to the text pane
textPane.setParagraphAttributes(centerAlign, true);

String javaText = "Java \n";
// 9. Insert the string in the content of text pane.
contentModel.insertString(contentModel.getLength(),
                           javaText, charStyle1);

// 10. Assign new character attributes to the text pane.
textPane.setCharacterAttributes(charStyle2, true);

String coolText = "JFC Swing is Cool! \n";
// 11. Insert the string in the content of text pane.
contentModel.insertString(contentModel.getLength(),
                           coolText, charStyle2);

// 12. Assign new character attributes to the text pane.
textPane.setCharacterAttributes(charStyle3, true);

String doneText = "Done!";
// 13. Insert the string in the content of text pane.
contentModel.insertString(contentModel.getLength(),
                           doneText, charStyle3);

} catch(BadLocationException blexcep) {
    System.err.println("Exception while inserting the string.");
    blexcep.printStackTrace();
}

// 14. Add the text pane to a scroll pane.
JScrollPane scrollPane = new JScrollPane(textPane);
// 15. Finally, add the scroll pane to the content pane.
getContentPane().add(scrollPane);
}
}

```

## 代码详析

在 TJTextPane 框架的构造函数里，片段 - 1 创建了一个文本窗格对象。片段 - 2 将段落属性设置为，信息在文本窗格中居中显示。片段 - 3 准备将字符属性置为字号为 75 的 Serif 字体。类似地，片段 - 4 和片段 5 准备为内容里的字符指定不同的属性集。

片段 - 6 得到文本窗格中内容模型的引用。片段 - 7 显示了定位光标和插入 Java Duke 图标的语句。片段 - 8 为该文本窗格指定了字符和段落属性集。片段 - 9 显示了将文本字符串插入到文本窗格的内容模型里的语句。片段 10 - 13 显示了类似于片段 8 和片段 9 中的步骤。片段 14 和片段 15 显示了利用滚动窗格将文本窗格加到框架里的代码。

# 第 12 章 滚动条和滑动条

消息、图像和草图常常是在小程序或应用程序中的框架容器内显示的。如果一个对象（也称子组件）的尺寸比其容器大，那就需要有某种机制能够支持将对象一部分一部分地显示出来。滚动条就是这样一种机制，它允许用户沿水平方向或垂直方向滚动显示一个子组件。

子组件的滚动是其某一部分在一个可见区域中不断重画的过程，重画过程依赖于滚动条的运动方向。本章将要讨论滚动条的内部运行机制，并给出一个例子用于说明滚动条的调整是如何影响滚动条参数的。

本章还将讨论另外一个被称为滚动窗格的实用滚动处理机制。滚动窗格是一个现成的窗格，它上面已经有了所需的滚动条。因此，用它实现滚动条更为简单。

从外观及内部属性来看，滑动条与滚动条是很相近的，但它们的用途却有所不同。滑动条用于给某个参数在一个连续区间范围内设定值。本章通过一个模仿媒体播放窗格的示例来讨论有关滑动条的内容。

## 12.1 滚动条

滚动条包括一个滚动块，滚动块沿着一个滑动条滑动，滑动条的两端各有一个端点滑动块。滚动块可以帮助用户按照他期望显示的数量，平滑地滚动子组件，并且端点滑动块还支持按指定方向逐行滚动。在滚动条的空区域内单击鼠标可以控制可见区域内子组件一块一块地滚动。

### 提示：

也许你用滚动窗格而非滚动条来滚动某个给定的子组件。比较而言，滚动条的实现要比滚动窗格的实现需要更多的编程过程。

Swing 滚动条是 `JScrollBar` 类的一个轻量级对象。`JScrollBar` 类存放在 `javax.swing` 包中，它是 `JComponent` 的一个直接子类。滚动条的 `orientation` 字段指定滚动条的方向应该为水平方向或者垂直方向，它的取值为常数 `JScrollBar.HORIZONTAL` 或 `JScrollBar.VERTICAL`。另外两个字段 `unitIncrement` 和 `blockIncrement` 的取值用来控制显示区域或视口内的子组件逐行或逐块地进行调整。调整的值由滚动条的最大值和最小值界定。

### 提示：

`JScrollBar` 类的内部所支持的功能通常已经足以实现滚动条。

滚动条对象的其他参数有最大值、最小值、当前值以及可视区域的范围。注意，这里所说的范围是指实际显示区域可见部分的数量。这些参数可以通过 `JScrollBar` 带参数的构造函数进行赋值，或者利用相关的方法进行赋值。

### 12.1.1 JScrollBar 构造函数

JScrollBar 类包括以下几种构造函数，用以创建滚动条对象：

```
public JScrollBar();
public JScrollBar(int orientation);
public JScrollBar(int orientation,
                  int value,
                  int extent,
                  int minimum,
                  int maximum);
```

构造函数中的 `orientation` 参数指定滚动条为水平的还是垂直的。`value` 参数是滚动条的初始值或当前值；`extent` 是可视范围或显示区域的大小；`minimum` 和 `maximum` 分别是滚动条的最小值和最大值。

### 12.1.2 调整事件和监听程序

当调查滚动条时，它会引发一些调整事件，这些事件是 `AdjustmentEvent` 事件类的对象。实现 `AdjustmentListener` 接口的类会接受这些事件。通过激活滚动条对象的 `AdjustmentListener()` 方法将事件监听类以源滚动条（source scrollbar）进行注册。`AdjustmentListener()` 方法以监听对象作为其参数值。

监听类必须从接口中实现 `AdjustmentValueChanged()` 方法，该方法以 `AdjustmentEvent` 对象作为其参数值。滚动条被调整时所要执行的代码就包含在这个方法中。

`AdjustmentEvent` 类包含了以下方法，它们可用来存取一些有用的参数：

```
public Adjustable getAdjustable()
public int getAdjustmentType()
public int getValue()
```

`getAdjustable()` 方法返回引发事件的那个可调整对象。后一个方法 `getAdjustmentType()` 则取得一个如下的调整类型：

```
UNIT_DECREMENT
UNIT_INCREMENT
BLOCK_DECREMENT
BLOCK_INCREMENT
TRACK
```

`getValue()` 方法取得调整事件的当前值。

### 12.1.3 JScrollBar 示例

清单 12.1 是一个 Swing 小程序，它演示了 `JScrollBar` 构造函数的用法，揭示了当滚动条被调整时，它的一些重要参数是如何变化的（参看图 12-1）。这个小程序还演示了相关的滚动条 API 是如何工作的。

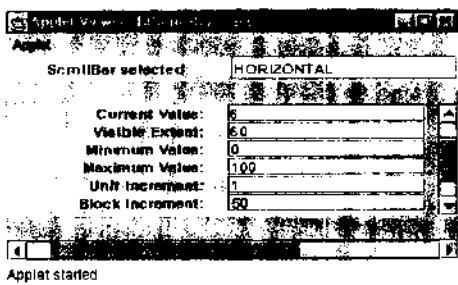


图 12-1 显示 JScrollBar 属性的 Swing 小程序

## 清单 12.1 JScrollBar (TJScrollBar.java)

```

/*
 * <Applet code = TJScrollBar.class width = 400 height = 200>
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TJScrollBar extends JApplet {
    // 1.Create references to two scrollbars.
    JScrollBar horizSBar = null;
    JScrollBar vertiSBar = null;

    // 2.Create the other necessary text fields and a panel.
    JPanel panel1 = null; JPanel panel2 = null;
    JTextField tf = null;
    JTextField tf1 = null; JTextField tf2 = null;
    JTextField tf3 = null; JTextField tf4 = null;
    JTextField tf5 = null; JTextField tf6 = null;

    public void init() {
        // 3.Get a handle to the applet's container.
        Container c = this.getContentPane();
        c.setBackground(Color.lightGray);

        // 4.Create horizontal and vertical scrollbar objects.
        horizSBar = new JScrollBar(JScrollBar.HORIZONTAL,
                                   20, 60, 0, 100);
        horizSBar.setBlockIncrement(50);
        ScrollBarListener hsbListener = new ScrollBarListener();
        horizSBar.addAdjustmentListener(hsbListener);

        vertiSBar = new JScrollBar();
        vertiSBar.setOrientation(JScrollBar.VERTICAL);
    }
}

```

```
vertiSBar.setValue(10);
vertiSBar.setVisibleAmount(30);
vertiSBar.setMinimum(0);
vertiSBar.setMaximum(50);
ScrollBarListener vslListener = new ScrollBarListener();
vertiSBar.addAdjustmentListener(vslListener);

// 5. Create a panel object panel1.
JPanel panel1 = new JPanel();
panel1.setLayout(new GridLayout(1,2));
// 6. Create a label and text field objects.
// and fix them to panel1
JLabel label = new JLabel("ScrollBar selected:",
    JLabel.CENTER);
tf = new JTextField();
panel1.add(label); panel1.add(tf);

// 7. Create a panel object panel2.
JPanel panel2 = new JPanel();
GridLayout gridLayout = new GridLayout(6,2);
gridLayout.setHgap(20);
panel2.setLayout(gridLayout);

// 8. Create the following labels and text fields, and
// fix them to the panel2.
JLabel label1 = new JLabel("Current Value:",JLabel.RIGHT);
JLabel label2 = new JLabel("Visible Extent:",JLabel.RIGHT);
JLabel label3 = new JLabel("Minimum Value:",JLabel.RIGHT);
JLabel label4 = new JLabel("Maximum Value:",JLabel.RIGHT);
JLabel label5 = new JLabel("Unit Increment:",JLabel.RIGHT);
JLabel label6 = new JLabel("Block Increment:",JLabel.RIGHT);
tf1 = new JTextField();
tf2 = new JTextField();
tf3 = new JTextField();
tf4 = new JTextField();
tf5 = new JTextField();
tf6 = new JTextField();
panel2.add(label1); panel2.add(tf1);
panel2.add(label2); panel2.add(tf2);
panel2.add(label3); panel2.add(tf3);
panel2.add(label4); panel2.add(tf4);
panel2.add(label5); panel2.add(tf5);
panel2.add(label6); panel2.add(tf6);

// 9. Set the border layout for the applet's content pane
// and add the panels and scrollbars as give next.
```

```
BorderLayout borderLayout = new BorderLayout();
borderLayout.setHgap(10);
borderLayout.setVgap(20);
c.setLayout(borderLayout);
c.add("North",panel1);
c.add("Center",panel2);
c.add("South",horizSBar);
c.add("East",vertiSBar);
}

// 10. A listener class that handle the scrollbar adjustment events
class ScrollBarListener implements AdjustmentListener {
    public void adjustmentValueChanged(AdjustmentEvent evt) {
        JScrollBar sBar = (JScrollBar) evt.getSource();

        if (sBar.getOrientation() == 0) {
            tf.setText("HORIZONTAL");
            tf1.setText(Integer.toString(sBar.getValue()));
            tf2.setText(Integer.toString(sBar.getVisibleAmount()));
            tf3.setText(Integer.toString(sBar.getMinimum()));
            tf4.setText(Integer.toString(sBar.getMaximum()));
            tf5.setText(Integer.toString(sBar.getUnitIncrement()));
            tf6.setText(Integer.toString(sBar.getBlockIncrement()));
        }
        else if (sBar.getOrientation() == 1) {
            tf.setText("VERTICAL");
            tf1.setText(Integer.toString(sBar.getValue()));
            tf2.setText(Integer.toString(sBar.getVisibleAmount()));
            tf3.setText(Integer.toString(sBar.getMinimum()));
            tf4.setText(Integer.toString(sBar.getMaximum()));
            tf5.setText(Integer.toString(sBar.getUnitIncrement()));
            tf6.setText(Integer.toString(sBar.getBlockIncrement()));
        }
    }
}
}
```

## 代码详析

程序片段 1 声明了两个 JScrollBar 类型对象的引用，horizSBar 和 vertiSBar。片段 2 创建了几个必需的 JPanel 类型和 JTextField 类型对象的引用。片段 3 取得了小程序内容窗格的句柄，它还利用内容窗格的句柄设置浅灰色为新的背景色。

片段 4 将先前创建的水平滚动条和垂直滚动条引用进行初始化。水平滚动条 horizSBar 由 JScrollBar 类的构造函数用 5 个参数来创建。这些参数指定了滚动条的方向、初始值、可

视范围以及滚动条的最大值和最小值。用于控制子组件向上或向下滚动数量的块增量值则在下一条语句中利用 `setBlockIncrement()` 方法来设定。水平滚动条的注册是由调整监听对象 `hsbListener` 利用 `addAdjustmentListener()` 方法完成的。

接下来，使用不带任何参数的构造函数创建了带有引用 `vertiSBar` 的垂直滚动条。然后滚动条属性通过使用 API 方法 `setOrientation()`、`setValue()`、`setVisibleAmount()`、`setMinimum()` 和 `setMaximum()` 被设为指定的值。代码片段中的最后一个语句注册了滚动条监听程序 `vsbListener`。

片段 - 5 创建了一个带有网格状布局的窗格对象 `panel1`。片段 - 6 给该窗格创建和加上一个标签和文本字段。片段 - 7 创建了另一个带有网格状布局的窗格对象 `panel2`。片段 - 8 创建了一个标签和文本字段集并将它们加到 `panel2` 上。片段 - 9 将 Swing 小程序设置为带边框的布局，而且两个窗格一个被放在中间，另一个放在布局的北边。水平和垂直滚动条分别放置在小程序的南边和东边。

片段 - 10 把监听程序类设置为处理调整事件。监听程序类 `ScrollBarListener` 执行 `AdjustmentListener`，并且为 `adjustmentValueChanged()` 接口方法提供代码。这个方法将 `AdjustmentEvent` 类型的对象作为其参数值。`adjustmentValueChanged()` 方法的主体提供了显示水平或垂直滚动条的代码，然后再显示选定的滚动条的属性值。

`getOrientation()` 方法返回滚动条的方位常数 0 或 1，分别指示水平或垂直方向。代码中的其余 `get` 方法简单地返回了当前值、可见的宽度、最小和最大值以及显示在小程序输出结果中的滚动条的单位和显示块的增量。

## 12.2 滚动窗格

滚动窗格是用来显示带内建滚动功能的子组件的。当子组件的尺寸大于有效视窗时，就使用与滚动窗格联合的滚动条来执行水平或垂直方向的滚动。滚动窗格很容易实现，因为调整事件是由已被滚动窗格关照过的滚动条激活的。只要空间允许，滚动窗格可以显示足够的子组件。

Swing 滚动窗格是由类 `JComponent` 扩展得到的 `JScrollPane` 类型的对象。在程序中实现一个滚动窗格时，可以使用另一种叫做 `JViewport` 的帮助类，`JViewport` 代表一个矩形显示窗口或者具有固定位置和边界的视窗。视窗可以被加上合适的边界（参看附录 A 中的 `JScrollPane` API “JFC Swing 快速参考”）。`JViewport` 就象照相机的取景器一样工作。当你滑动滚动窗格的滚动条时，视窗就随之移动以使看到目标对象的特定位置。

有时需要双缓冲器来避免图像（子组件）的闪烁。这种情况下，一般使用支持双缓冲器的 `JPanel` 然后将子组件加到这个 `JPanel` 对象上。

Swing 滚动窗格也包含核心组件和行、列标题。核心组件可以被加到滚动窗格的每一个有效核心空间。每一个行、列标题都是一个需要指定的 `JViewport` 对象。列标题视窗左右滚动，跟踪主视窗的左右滚动，而行标题在对应的方向滚动。

### 12.2.1 滚动窗格常数

公共界面 `ScrollPaneConstants` 提供大量的滚动窗格常数。为了设置关联的水平和垂直滚

动条, JScrollPane 类实现这个接口。下面给出的是常用的滚动窗格常数。当滚动条总是显示或从未显示时, 用来显示滚动条的这些常数都是自解释性的。作为必要的基础, 当一个子组件大于视窗时就会显示滚动条。

```
HORIZONTAL_SCROLLBAR_ALWAYS  
HORIZONTAL_SCROLLBAR_AS_NEEDED  
HORIZONTAL_SCROLLBAR_NEVER
```

```
VERTICAL_SCROLLBAR_ALWAYS  
VERTICAL_SCROLLBAR_AS_NEEDED  
VERTICAL_SCROLLBAR_NEVER
```

### 12.2.2 JScrollPane 构造函数

下面列出的是由 JScrollPane 类提供的构造函数, 可以使用其中任何一个函数来创建滚动窗格对象:

```
JScrollPane();  
JScrollPane(Component childComponent);  
JScrollPane(int vertSBarPolicy, int horizSBarPolicy);  
JScrollPane(Component childComponent,  
           int vertSBarPolicy,  
           int horizSBarPolicy);
```

在这些构造函数里, 参数 childComponent 是要显示的子组件对象, 另两个参数 horizSBarPolicy 和 vertSBarPolicy 描述了在滚动窗格里显示滚动条的模型。这些参数接受上一小节列出的 ScrollPaneConstants 接口提供的常数。

### 12.2.3 滚动窗格举例

清单 12.2 显示了一个简单的图像和滚动窗格 (见图 12-2)。注意, 已经用来实现滚动窗格的代码的简单性是很重要的。无须个别地处理调整事件。

清单 12.2 JScrollPane (TJScrollPane.java)

```
/*  
 * <Applet code = TJScrollPane.class width = 300 height = 200>  
 * </Applet>  
 */  
  
import javax.swing.*;  
import java.awt.*;  
  
public class TJScrollPane extends JApplet {  
    // 1. Create a scroll pane object and the other  
    // necessary objects.  
    JScrollPane scrollPane = null;
```

```
JLabel label = null; // Not a canvas for JScrollPane!
JPanel panel = null; // supports double buffering
Icon icon = null;

public void init() {
    // 2. Get a handle on the JApplet's container.
    Container container = getContentPane();
    container.setLayout(new GridLayout(1,1));

    // 3. Create a Swing label and a panel for double buffering.
    icon = new ImageIcon("saravan.gif");
    label = new JLabel(icon);
    panel = new JPanel();
    panel.add(label);

    // 4. Create a scroll pane and add the panel to it.
    scrollPane = new JScrollPane(panel,
        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
        JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

    // 5. Add the scroll pane to the contentpane of JApplet.
    container.add(scrollPane);
}

}
```

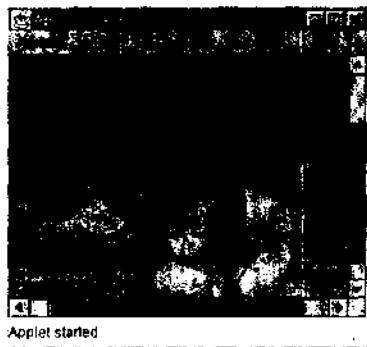


图 12-2 用 JScrollPane 显示的图像

## 代码详析

片段 - 1 声明了 JScrollPane 类型的对象引用，以及 JLabel、Icon 和 JPanel 类型的有帮助的对象引用。片段 - 2 提供了为该 Swing 小程序的内容窗格创建一个句柄的代码。另外小程序还被设置为 GridLayout 类型的布局。

片段 - 3 首先从给定图像文件中创建了一个图标，然后将它加到一个标签对象上。标签对象准备好以后就可以加到片段 - 4 所创建的 JScrollPane 对象上。但是在这个程序里，标签对象被加到 JPanel 类型的窗格上， JPanel 类型的对象支持控制图像闪烁的双缓冲器。片段 - 4 将窗格对象通过其构造函数加到滚动窗格对象上。

**说明：**

Swing 组件也可以起容器的作用，因为从类的继承关系看，它们是从 AWT 类 Container 继承的子类。这就允许 JPanel 对象通过了 JScrollPane 的构造函数。

构造函数也接受前面列出的总是显示垂直滚动条的 Swing 窗格常数。而水平滚动条只在图像宽度大于视窗宽度时才显示。最后，片段 - 5 将滚动窗格加到 Swing 小程序的内容窗格上。

## 12.3 滑动条

滑动条是带有指针滑动块的控制器，该指针滑动块可以在一个连续的值域上为已使用小配件的参数选择指定的值。经常在媒体播放器中见到的滑动条用来调节自动音量、频道频率、图像的对比度或亮度等等。

**说明：**

在 AWT 中没有等价的重量级的滑动条组件。AWT 滚动条加上某些额外的特性就转化为滑动条。

Swing 类 JSlider 是 JComponent 的直接子类并且代表滑动条小配件。类 JSlider 也实现 SwingConstants 接口。也可以通过指定类 JSlider 中 majorTickSpacing 和 minorTickSpacing 字段的值将滑动条组件画上对勾记号。也可以通过将方法 setPaintLabels () 设为真 (true) 来为对勾记号指定标签。

### 12.3.1 JSlider 构造函数

可以使用下面任何一个构造函数来创建滑动条：

```
public JSlider();
public JSlider(int orientation,
               int minimum, int maximum,
               int value);
```

这些不带参数的构造函数创建了一个水平的滑动条，它的数值从 0 到 100，初始值为 50。参数方向决定滑动条是水平的或垂直的并接收 JSlider.HORIZONTAL 或 JSlider.VERTICAL 值。其余参数指定了滑动条参数的最小、最大值，以及它在给定队列中的初始值。

### 12.3.2 滑动条事件和监听程序

滑动条的滑块一移动，组件就激活一个由监听程序类识别的 ChangeEvent 类型的事件。监听程序类实现 ChangeListener 接口。任何实现 ChangeListener 接口的类都必须为公共方法 stateChanged () 提供代码。这样，当滑动条滑块一移动时，要执行的代码便放在这个方法之中。

stateChanged () 方法接收 ChangeEvent 类型的事件。除了从 java.util.EventObject 和

java.lang.Object 继承的方法以外, ChangeEvent 不包含它自己的任何方法。

### 12.3.3 JSlider 代码举例

显示在图 12-3 中的 Swing 小程序演示了 JSlider 类的 API 方法。该小程序仿效了一个音频播放器的控制窗格。该窗格包括两个垂直的滑动条, 它们分别代表左声道和右声道的音量控制。单个频道的即时值显示在音量小配件上方的文本字段里。

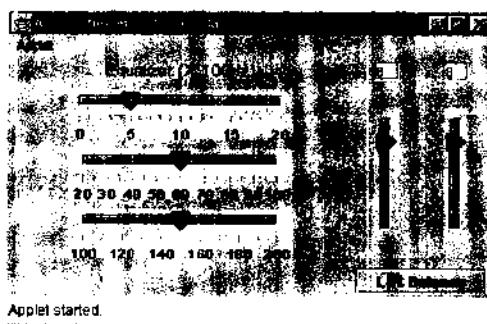


图 12-3 用 JSlider 显示一个音频系统控制窗格的 Swing 小程序

显示在音量控制条底部的切换按钮用来平衡左右声道的音量水平。当按钮被按下时, 右声道的值被自动调节以适合左声道的值。在平衡按钮被按下的条件下, 一旦调节其中一个音量滑块, 另一个就会自动被调节为相同的水平。另一组三个水平滑动条代表为音频谱显示对勾记号的三条均衡器。此小程序的代码在清单 12.3 中给出。

清单 12.3 JSlider 音频控制窗格 (TJSlider.java)

```
// Demonstrates the Swing slider widget...
/*
 * <Applet code = TJSlider.class width = 400 height = 200>
 * </Applet>
 */
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

public class TJSlider extends JApplet {
    // 1. Declare the required JSlider references.
    JSlider slider = null;
    JSlider slider1 = null;
    JSlider slider2 = null;
    JSlider slider3 = null;
    JSlider slider4 = null;
    JSlider slider5 = null;
    // 2. Other necessary fields
```

```
JTextField textField = null;
JTextField textField1 = null;
JTextField textField2 = null;
GridBagLayout gridbag = null;
GridBagConstraints c = null;
Container container = null;
boolean buttonPressed = false;

public void init() {
    // 3. Get a handle on the container of JApplet
    // and assign the color and layout model.
    container = this.getContentPane();
    container.setBackground(Color.lightGray);
    gridbag = new GridBagLayout();
    container.setLayout(gridbag);

    // 4. Constraints for the layout
    c = new GridBagConstraints();
    c.weightx = 1.0; c.weighty = 1.0;
    c.fill = GridBagConstraints.NONE;
    c.insets = new Insets(4, 4, 4, 4);

    // 5. Label showing E qualizer (x 100Hz)
    c.gridx = 0; c.gridy = 0;
    c.gridwidth = 1; c.gridheight = 1;
    c.ipadx = 150;
    Font font = new Font("Helvetica", Font.BOLD, 14);
    JLabel label = new JLabel("Equalizer (X 100Hz)",
        JLabel.CENTER);
    label.setFont(font);
    gridbag.setConstraints(label, c);
    container.add(label);

    // 6. Create horizontal slider1.
    c.gridy = 1;
    setSlider(JSlider.HORIZONTAL, true,
        0, 20, 5, 5, 1);
    slider3 = slider;
    slider3.setLabelTable(slider3.createStandardLabels(5));
    slider3.setPaintLabels(true);

    // 7. Create horizontal slider2.
    c.gridy = 2;
    setSlider(JSlider.HORIZONTAL, true,
        20, 100, 60, 10, 5);
    slider4 = slider;
```

```
slider4.setLabelTable(slider4.createStandardLabels(10));
slider4.setPaintLabels(true);

// 8.Create horizontal slider3.
c.gridx = 3;
setSlider(JSlider.HORIZONTAL,true,
           100,200,150,20,10);
slider5 = slider;
slider5.setLabelTable(slider5.createStandardLabels(20));
slider5.setPaintLabels(true);

// 9.Create the toggle button for the volume balance.
c.ipadx = 0;
c.gridx = 1; c.gridy = 4;
c.gridwidth = 2; c.gridheight = 1;
setButton("L|R Balance");

// 10.Create volume slider1.
c.ipady = 75;
c.gridy = 1;
c.gridwidth = 1; c.gridheight = 3;
setSlider(JSlider.VERTICAL,false,0,10,8);
slider1 = slider;

// 11.Create volume slider2.
c.gridx = 2; c.gridy = 1;
setSlider(JSlider.VERTICAL,false,0,10,8);
slider2 = slider;

// 12.Create textfield1 for the volume slider1.
c.ipadx = 0; c.ipady = 0;
c.gridx = 1; c.gridy = 0;
c.gridwidth = 1; c.gridheight = 1;
setTextField(slider1);
textField1 = textField;

// 13.Create textfield2 for the volume slider2.
c.gridx = 2;
setTextField(slider2);
textField2 = textField;
}

// 14.Creates a slider object.
public void setSlider(int orientation,
                      boolean paintTicks,
                      int minValue, int maxValue,
                      int initialValue) {
    setSlider(orientation,paintTicks,
```

```
        minimumValue, maximumValue, initialValue, 0, 0);
    }

// 15. Overload the previous above.
public void setSlider(int orientation,
                      boolean paintTicks,
                      int minValue, int maxValue,
                      int initialValue,
                      int majorTickSpacing,
                      int minorTickSpacing) {
    slider = new JSlider{orientation,
                        minValue, maxValue,
                        initialValue};
    slider.addChangeListener(new SliderListener());
    slider.setPaintTicks(paintTicks);
    slider.setMajorTickSpacing(majorTickSpacing);
    slider.setMinorTickSpacing(minorTickSpacing);
    gridbag.setConstraints(slider, c);
    container.add(slider);
}

// 16. Create the toggle button for the balance of channels.
public void setButton(String name) {
    JToggleButton button = new JToggleButton(name);
    button.setBackground(Color.lightGray);
    gridbag.setConstraints(button, c);
    button.addActionListener(new ButtonListener());
    container.add(button);
}

// 17. Create text field objects.
public void setTextField(JSlider slider) {
    textField = new JTextField(2);
    textField.setText(Integer.toString(slider.getValue()));
    gridbag.setConstraints(textField, c);
    container.add(textField);
}

// 18. Button listener for the channel balance
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent actEvt) {
        JToggleButton buttonTemp = (JToggleButton)
            actEvt.getSource();
        buttonPressed = buttonTemp.isSelected();
        slider2.setValue(slider1.getValue());
    }
}
```

```

}

// 19.The slider knob position change listener
class SliderListener implements ChangeListener {
    public void stateChanged(ChangeEvent chngEvt) {
        updateTextField(slider1.getValue(),
                        slider2.getValue());

        JSlider sliderTemp = (JSlider) chngEvt.getSource();
        if(buttonPressed) {
            if(sliderTemp == slider1) {
                slider2.setValue(slider1.getValue());
            }
            else if(sliderTemp == slider2) {
                slider1.setValue(slider2.getValue());
            }
        }
    }

    public void updateTextField(int currValue1, int currValue2) {
        textField1.setText(Integer.toString(currValue1));
        textField2.setText(Integer.toString(currValue2));
    }
}

```

## 代码详析

为小程序中能使用滑动条对象，上述代码的片段 - 1 声明了所需引用。片断 - 2 显示了对其他诸如文本字段、相关对象的布局等的声明。片段 - 3 使用方法 `getContentPane()` 为内容窗格获得一个句柄。并将小程序的背景改为浅灰色。然后，该小程序为配置的组件设置为网格状的布局。

片段 - 4 初始化了所有组件对象的约束字段，这些组件加在代码的下一个片段。片段 5 - 13 预先描述了对小程序不同组件的约束条件，然后调用相应的方法创建组件对象。片段 14 - 17 提供了实际创建所需组件对象的方法，然后按照指定的布局约束把它们固定到小程序上。

片段 - 18 为切换按钮提供了监听程序类。叫做 `ButtonLlistener` 的监听程序类实现 `ActionListener` 接口。这个类为接收 `ActionEvent` 类型对象为参数值的方法 `actionPerformed()` 提供代码。该方法为给切换按钮获得一个临时引用提供代码，并把切换按钮的选择状态存储在逻辑变量 `buttonPressed` 中。该方法的最后一个语句将 `slider2` (右声道) 的值设置为 `slider1` (左声道) 返回的值。

片段 - 19 处理由音量滑动条对象所激活的事件。内部类 `SliderListener` 实现 `ChangeListener` 类，并为 `stateChanged()` 方法提供代码。`stateChanged()` 方法接收 `ChangeEvent` 类型的对象作为它的参数值。基本上，`SliderListener` 类支持调整 `slider1` 或 `slider2` 的滑块时所需

的功能。当滑动条被调整时，`stateChanged()`方法调用`updateTextField()`方法，来修改位于滑动条上方的文本字段的值。`stateChanged()`方法的代码的其余部分处理平衡按钮被按下和滑动条被调整时的情况。在`if`和`else...if`子句里的代码显示了，`slider2`的滑块一移动，`slider1`就调整它的即时值，反之亦然。

# 第 13 章 菜单、工具条和工具提示

菜单、工具条和工具提示均属于标准 GUI 组件。在菜单条的项目上单击一次，就会打开一个带有子任务的下拉菜单。弹出式菜单允许单击鼠标右键然后出现一个菜单。工具条则将涉及某些关键性功能的组件集合在一起（通常是带有图标的按钮）。工具提示是当用户的鼠标指针暂停在一个接口组件上时弹出的一小段文字。

## 13.1 菜单条和菜单

菜单条是一个窄的矩形组件，它位于小程序或框架的顶部边缘。菜单条包括附有下拉菜单的标签。图 13-1 显示了一个 Swing 菜单条。

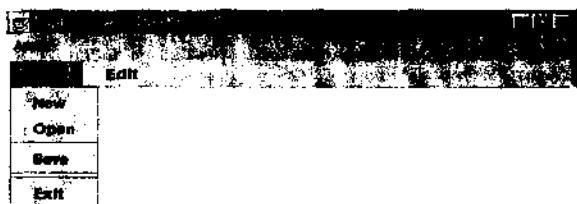


图 13-1 一个 Swing 菜单条和菜单

图 13-1 还显示了用来将一个操作明显区别于另一个操作的分隔线。分隔线就是如图所示的画在菜单上的简单的直线。该图中的分隔线将诸如创建一个新文档 (New)、打开已存在的文件 (Open)、关闭一个文件 (Close) 的这类操作与另一类操作如多种选择地保存一个文件 (Save 或 Save As) 分隔开来。

Swing 菜单条是由 Swing 类 JMenuBar 的对象代表的轻量级组件。类 JMenuBar 由类 JComponent 扩展而来，并存储在程序包 javax.swing 里。Swing 菜单由类 JMenu 表示，它作为 JMenuBar 存在同一程序包内。JMenu 是 JMenuItem 类的扩展类，而 JMenuItem 又是 AbstractButton 类的扩展类。图 13-2 显示了 JMenuBar 和 JMenu 类的层次结构。

### 13.1.1 JMenuBar 构造函数

在应用程序或小程序中，只有唯一的一个构造函数能有效地创建菜单条。最有代表性的例子是，需要创建一个菜单条，创建一系列菜单并将它们加到菜单条里，然后再把这个菜单条放到应用程序框架或小程序中。为了将菜单条放到它的容器里，需要使用 add () 方法，下面就是这个菜单条构造函数：

```
public JMenuBar ()
```

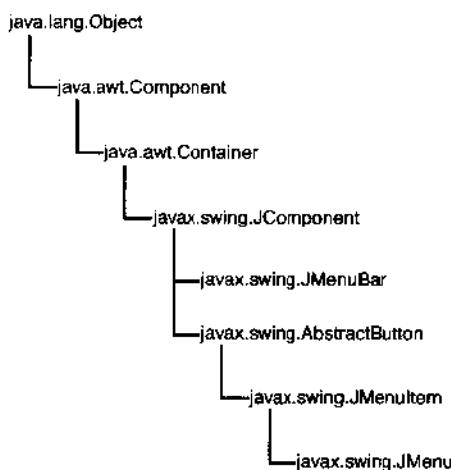


图 13-2 JMenuBar 和 JMenu 类的层次结构

### 13.1.2 JMenu 构造函数

一旦建立了一个菜单条，就可以用下面的构造函数加入菜单项：

```

public JMenu()
public JMenu(String label)
public JMenu(String label,boolean tornOffStatus)
  
```

第一个构造函数创建了一个没有任何标签的菜单，但可以通过使用其父类 AbstractButton 的方法 `setText()` 设置为任何标签。下一个构造函数创建了附加一个文本标签的菜单。第三个构造函数除额外的逻辑参数以外，执行同样的功能。这个逻辑参数用来声明该菜单是否是剥离式的，所谓剥离式菜单意味着它有自己的外观（look-and-feel）。

### 13.1.3 JMenuBar 和 JMenu 代码示例

清单 13.1 演示了 Swing 菜单条和菜单对象。该程序的输出结果显示在图 13-3 中。此程序创建了一个菜单条和一些菜单项，并使用 JMenuBar 和 JMenu 里的一些 API 方法为这些组件加上一定的特性。当菜单被加到菜单条中以后，菜单条又被放到小程序中。



图 13-3 一个显示菜单标签 (JMenu) 的 Swing 菜单条 (JMenuBar)

#### 清单 13.1 带有 JMenu (JMenuBar.java) 的 JMenuBar

```
// Demonstrates the Swing menu bar and menus
```

```
/*
 * < Applet code = TJMenuBar width = 450 height = 50 >
 * </Applet>
 */

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;

public class TJMenuBar extends JApplet {
    public void init() {
        // 1. Get the handle on the applet's content pane.
        Container container = this.getContentPane();

        // 2. Create a menu bar with bevel border and
        // add it to the applet.
        JMenuBar menuBar = new JMenuBar();
        menuBar.setBorder(new BevelBorder(BevelBorder.RAISED));
        container.add(menuBar, BorderLayout.NORTH);
        // 3. Create menus for a simple editor.
        JMenu fileMenu = new JMenu("File", true);
        JMenu editMenu = new JMenu("Edit");
        JMenu formatMenu = new JMenu("Format");
        JMenu optionsMenu = new JMenu("Options");

        // 4. Add the menus to the menu bar.
        menuBar.add(fileMenu);
        menuBar.add(editMenu);
        menuBar.add(formatMenu);
        menuBar.add(optionsMenu);

        // 5. Get the handle on each menu to the normal and
        // selected icons and the mnemonic (short-cut) key.
        for (int i = 0; i < menuBar.getMenuCount(); i++) {
            JMenu menu = menuBar.getMenu(i); // returns the menu
            menu.setIcon(new ImageIcon("red.gif"));

            // Set mnemonic key.
            String text = menu.getText(); // gets the menu label
            menu.setMnemonic(text.charAt(0)); // at first char
        }
    }
}
```

## 代码详析

片段-1 得到内容窗格的句柄。片段-2 创建了菜单条对象并将其附加到小程序上，此

菜单条被设计为有外凸斜角的边界。片段 -3 创建了必需的菜单，片断 -4 将这些菜单加到菜单条里。

片段 -5 使用由此菜单条对象的 `getMenuCount()` 方法取得的菜单项数量给菜单设计图标和记忆键。为了从菜单条中得到每一个菜单项，只需调用菜单条对象中的方法 `getMenu()`。该程序的输出结果显示在图 13-3 中。

## 13.2 菜单项

菜单项本质上就是位于下拉式菜单中的一种按钮。当最后一个用户选择了菜单项时，与其相关的动作就会被执行。一个菜单项可以用来打开另一个包含更多菜单项的菜单。

菜单项由类 `JMenuItem` 表示，这个类是 `AbstractButton` 的子类，存储在程序包 `javax.swing` 中。由 `JMenuItem` 扩展的两个更重要的子类分别名为 `JCheckBoxMenuItem` 和 `JRadioButtonMenuItem`。图 13-4 显示了这些组件的类层次结构。

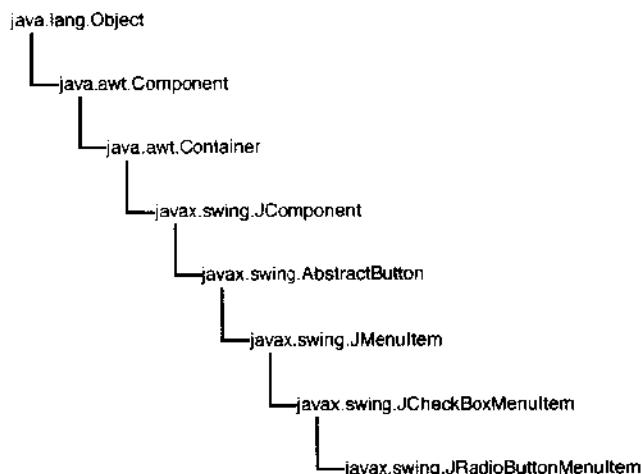


图 13-4 菜单项组件中的类层次结构

### 13.2.1 JMenuItem 构造函数

Swing 菜单项可以被创建为带有简单文本、带有图标或者二者兼有，还可以指定其记忆键。典型的做法是，创建一个菜单项并将其加到菜单中。下面是支持创建菜单项对象的构造函数列表：

```
public JMenuItem()
public JMenuItem(Icon icon)
public JMenuItem(String text)
public JMenuItem(String text, Icon icon)
public JMenuItem(String text, int mnemonic)
```

### 13.2.2 JMenuItem 代码示例

清单 13.2 创建了一个包含菜单条、菜单、子菜单和菜单项的小程序。可以通过选项(Options)菜单打开深一层的子菜单(见图 13-5)。



图 13-5 显示菜单、子菜单和菜单项的 Swing 小程序

清单 13.2 带有 JMenuItem ( TJMenu.java ) 的 JMenu

```
// Demonstrates the Swing menus, submenus, and menu items

/*
 * <Applet code = TJMenu width = 450 height = 50>
 * </Applet>
 */

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

public class TJMenu extends JApplet {
    public void init() {
        // 1. Get the handle on the applet's content pane.
        Container container = this.getContentPane();

        // 2. Add a menu bar to the applet.
        JMenuBar menuBar = new JMenuBar();
        menuBar.setBorder(new BevelBorder(BevelBorder.RAISED));
        menuBar.setBorderStylePainted(true);
        container.add(menuBar, BorderLayout.NORTH);

        // 3. Add the File menu and its menu items .
        JMenu fileMenu = new JMenu("File", true);
        menuBar.add(fileMenu);

        // 4. Add the submenus to the File menu.
        fileMenu.add(new JMenuItem("New"));
        fileMenu.add(new JMenuItem("Open"));
        fileMenu.addSeparator();
        fileMenu.add(new JMenuItem("Save"));
        fileMenu.add(new JMenuItem("Save As"));
    }
}
```

```
fileMenu.addSeparator();
fileMenu.addSeparator();

// 5. Add the Edit menu and its menu items.
JMenu editMenu = new JMenu("Edit");
menuBar.add(editMenu);

// 6. Add the submenus to the edit menu.
editMenu.add(new JMenuItem("Undo"));
editMenu.addSeparator();
editMenu.add(new JMenuItem("Cut"));
editMenu.add(new JMenuItem("Copy"));
editMenu.add(new JMenuItem("Paste"));

// 7. Create and add the Options menu and submenus
// and their items.
JMenu optionsMenu = new JMenu("Options");
menuBar.add(optionsMenu);

// 8. Add the submenus to the Options menu.
JMenu bookMarksMenu = new JMenu("Book Marks");
optionsMenu.add(bookMarksMenu);

// 9. Add the submenus to the Book Marks menu.
JMenuItem addMI = new JMenuItem("Add Alt - K");
bookMarksMenu.add(addMI);
JMenuItem editMI = new JMenuItem("Edit Alt - B");
bookMarksMenu.add(editMI);
JMenu guideMenu = new JMenu("Guide");
bookMarksMenu.add(guideMenu);

// 10. Add the submenus to the Guide menu.
JMenuItem whatIsNewMI = new JMenuItem("What's New");
whatIsNewMI.setMnemonic('N');
guideMenu.add(whatIsNewMI);
JMenuItem whatIsCoolMI = new JMenuItem("What's Cool");
whatIsCoolMI.setMnemonic('C');
guideMenu.add(whatIsCoolMI);

// 11. Finally, add two more submenus to the Options menu.
JMenuItem javaConsoleMI = new JMenuItem("Java Console");
optionsMenu.add(javaConsoleMI);
JMenuItem addressBookMI = new JMenuItem("Address Book");
optionsMenu.add(addressBookMI);
}

}
```

## 代码详析

在小程序 init() 方法内部，片段 - 1 获得该小程序的内容窗格的句柄。片段 - 2 创建了一个菜单条，并将它附加在小程序的上部。片段 3 和 4 创建了文件菜单和它的菜单项，然后文件（File）菜单被加到小程序上。类似地，片段 5 和 6 创建了编辑（Edit）菜单和它的菜单项。

片段 7-10 相当有趣。片段 - 7 创建了菜单标题“选项”（Options）并将其加到菜单条上。片段 - 8 创建了“书签”（Book Marks）子菜单并将其加到选项菜单上。片段 - 9 又为“书签”菜单创建了一个子菜单。片段 - 10 给一个名为“指南”（Guide）的子菜单创建并加上菜单项。片段 - 9 已经创建了子菜单“指南”（Guide）。片段 - 11 只是简单地将两个子菜单加到“选项”（Options）菜单上。

## 13.3 复选框菜单项

复选框菜单项就是那些类似于复选框的菜单项。可以通过选取或者不选取相关的图标来选定或取消选定一个菜单项。复选框菜单项通常用来选定或取消选定一个程序属性集。

复选框菜单项由 JCheckBoxMenuItem 类表示，它们象普通菜单项一样，可用同样的方法创建并加入到菜单之中。

```
public JCheckBoxMenuItem()
public JCheckBoxMenuItem(Icon icon)
public JCheckBoxMenuItem(String text)
public JCheckBoxMenuItem(String text, Icon icon)
public JCheckBoxMenuItem(String text, boolean state)
public JCheckBoxMenuItem (String text, Icon icon,
                           boolean state)
```

在此构造函数列表中，第一个构造函数创建了一个复选框菜单项，它开始没有被选定也没有任何文本或设置图标。下一个构造函数创建了一个没有被选定但是有设置图标的复选框菜单项。第三个构造函数创建了一个带有指定文本的取消选定的复选框菜单项。第四个构造函数类似于第三个构造函数，只是指定了被设置的图标。最后一个构造函数是万能型的，你可以在其中指定文本、图标和选择状态。

## 13.4 单选按钮菜单项

单选按钮菜单项类似于复选框菜单项，不过在单选按钮菜单项组中，每次只能选择一项。所以在选定了一个单选按钮菜单项以后，其他选定项便被取消。当最后一个用户需要从菜单项的有效列表中每次只选择一项时，就可以使用这些组件。

为了在程序中实现一个单选按钮菜单项，典型的做法是，用它的构造函数创建一个单选按钮，并将它加到使用 ButtonGroup 类型的对象的创建组中。下面是创建 JRadioButtonMenuItem 对象的构造函数列表：

```

public JRadioButtonMenuItem()
public JRadioButtonMenuItem(Icon icon)
public JRadioButtonMenuItem(String label)
public JRadioButtonMenuItem(String label, Icon icon)

```

这些构造函数创建了尚未选定的单选按钮菜单项。通过使用 `Icon` 类型的参数可以给组件使用指定的图标。名为 `label` 的参数是菜单项的可以提供的文本标签。

### 13.4.1 JCheckBoxMenuItem 和 JRadioButtonMenuItem 代码示例

清单 13.3 演示了怎样实现复选框菜单项和单选按钮菜单项。该程序是带有菜单条和菜单的小程序。“选项”（Options）菜单包括“字体”（Fonts）和“其他”（Others）子菜单。当单击“字体”（Fonts）菜单时，就会显示三个单选按钮菜单项。这些单选按钮菜单项表示字体，每次只能从中选定一项。类似地，当单击“Others”选项菜单时，就会显示一系列高级选项。这些选项由复选框菜单项表示，从中可以选定任意多项。该程序的输出结果如图 13-6 所示。

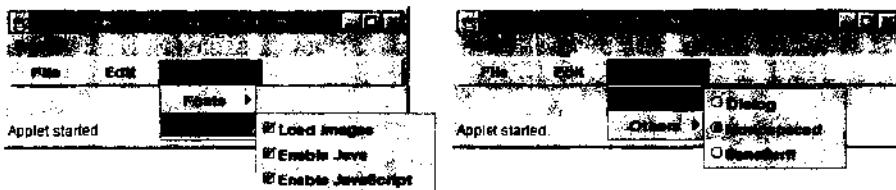


图 13-6 显示复选框菜单项（左）和单选按钮菜单项（右）的 Swing 小程序

清单 13.3 JCheckBoxMenuItem 和 JRadioButtonMenuItem (`TJCheckBoxMenuItem.java`)

```

menuBar.setBorder (new BevelBorder (BevelBorder.RAISED));
//menuBar.setBorderPainted (true);
container.add (menuBar, BorderLayout.NORTH);
// Demonstrates the check box menu items
// and radio button menu items
/*
 * < Applet code = TJCheckBoxMenuItem width = 450 height = 50 >
 * </Applet>
 */
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
public class TJCheckBoxMenuItem extends JApplet {
    public void init () {
        // 1. Get the handle on the applet's content pane.

```

```
Container container = this.getContentPane ();  
  
// 2.Create a menu bar and add it to the applet.  
JMenuBar menuBar = new JMenuBar ();  
  
// 3.Create and add the File menu and its menu items.  
JMenu fileMenu = new JMenu ("File", true);  
menuBar.add (fileMenu);  
  
// 4.Create the Edit menu and its menu items  
// and add them to the menu bar.  
JMenu editMenu = new JMenu ("Edit");  
menuBar.add (editMenu);  
  
// 5.Create the Options menu, submenus, and their items.  
JMenu optionsMenu = new JMenu ("Options");  
menuBar.add (optionsMenu);  
  
// 6.Create and add the Fonts options menu to the  
// Options menu.  
JMenu fontsOptionsMenu = new JMenu ("Fonts");  
optionsMenu.add (fontsOptionsMenu);  
  
// 7.Create radio button menu items and add them  
// to the Fonts option menu.  
JRadioButtonMenuItem rbItem;  
ButtonGroup group = new ButtonGroup ();  
String [] rbLabels = { "Dialog", "Monospaced", "SansSerif" };  
for (int i = 0; i < rbLabels.length; i  
) {  
    rbItem = new JRadioButtonMenuItem (rbLabels [i]);  
    fontsOptionsMenu.add (rbItem);  
    group.add (rbItem);  
}  
  
// 8.Create and add the Others options menu to the  
// Options menu.  
JMenu advancedOptionsMenu = new JMenu ("Others");  
optionsMenu.add (advancedOptionsMenu);  
  
// 9.Create and add the check box menu items to the  
// Others options menu.  
JCheckBoxMenuItem cbItem;  
String [] cbLabels = { "Load Images", "Enable Java",  
                     "Enable JavaScript" };  
for (int i = 0; i < cbLabels.length; i)  
    cbItem = new JCheckBoxMenuItem (cbLabels [i]);  
    advancedOptionsMenu.add (cbItem);
```

### 代码详析

片段 - 1 从小程序的内容窗格得到句柄。片段 - 2 创建了一个菜单条并把它加到小程序之中。片段 3 ~ 5 创建了不同的菜单并把它们加到菜单项上。片段 - 6 创建了“字体”(Fonts) 子菜单项并把它加到“选项”(Options) 菜单上。

单选按钮对象是在片段 - 7 里的 for 循环语句中创建的。单选按钮菜单项已经被加到“字体”(Fonts) 菜单里。注意，就象在这个片段里一样，必须把单选按钮菜单项集合在一起。

片段 - 8 创建了“其他”(Others) 菜单并把它加到“选项”(Options) 菜单上。片段 - 9 创建了复选框菜单项并把它们加到“其他”(Others) 菜单上。程序执行后的输出结果显示在前面的图 13-6 中。

## 13.5 弹出式菜单

弹出式菜单是与内容相关的且包含一系列选项的菜单。当鼠标的第二个键（通常是鼠标右键）在小程序或框架窗口里被按下时，就会弹出这样的菜单。用户还可以将光标移到开发者定义的热域上来使菜单弹出。

弹出式菜单经常被专业使用者作为访问重要选项的快捷键来使用。这些菜单通过使光标位置的选项有效使鼠标移动最小化。通常弹出式菜单包含了应用程序的关键性质。

#### 说明：

弹出式菜单并不意味着这些选项在系统的其他地方就无效。

Swing 弹出式菜单由 JPopupMenu 类表示，它存储在 javax.swing 程序包里。JPopupMenu 类是由 JComponent 类直接派生的，如图 13-7 所示。

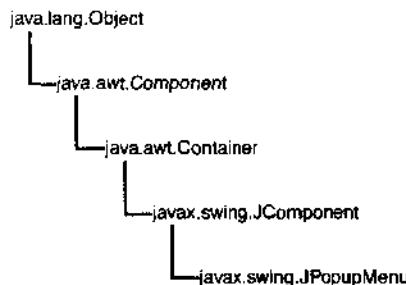


图 13-7 JPopupMenu 的类层次式结构

### 13.5.1 JPopupMenu 构造函数

必须使用下列 JPopupMenu 构造函数来创建弹出式菜单（见下页）：

```
public JPopupMenu()
public JPopupMenu(String label)
```

String类型的参数label将任何适合的文本字符串作为弹出式菜单的标题。

用 JMenuItem 类型的菜单项，可以创建菜单对象。因为弹出式菜单出现在鼠标被按下的地方，所以鼠标监听程序由某个组件注册，用该组件弹出式菜单将被显示。然后当鼠标被按下并释放时，就使用方法 show 来显示弹出式菜单。

### 13.5.2 JPopupMenu 代码示例

程序清单 13.4 创建了一个弹出式菜单并演示了怎样在文本区域组件中使用它。该例子还演示了 JPopupMenu 类中的一些方法。其输出结果列在图 13-8 中。

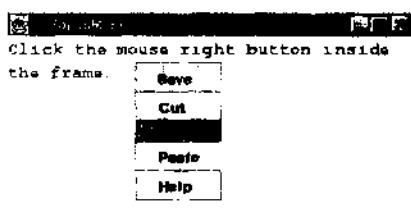


图 13-8 一个显示弹出式菜单的文本窗口

清单 13.4 显示 JPopupMenu (TJPopupMenu.java) 的文本窗口

```
// Demonstrates the Swing pop-up menus

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TJPopupMenu extends JFrame {
    JPopupMenu popupMenu;
    JMenuItem saveItem, cutItem, copyItem, pasteItem, helpItem;
    JTextArea textArea;
    Container container;

    public TJPopupMenu() {
        // 1. For frame closing.
        this.addWindowListener(new FrameClosing());
        // 2. Get a handle on the frame's content pane.
        container = this.getContentPane();
        // 3. Create and add the text area to the content pane.
        textArea = new JTextArea("Click the mouse right button inside"
                               + " \nthe frame.");
        textArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        container.add(textArea);
    }
}
```

```
// 4.Create a pop-up menu.  
popupMenu = new JPopupMenu("Test Popup Menu");  
  
// 5.Create menu items and add them to the pop-up menu.  
// Also add some separators as shown in the code.  
saveItem = new JMenuItem("Save");  
popupMenu.add(saveItem);  
  
popupMenu.addSeparator();  
  
cutItem = new JMenuItem("Cut");  
popupMenu.add(cutItem);  
  
copyItem = new JMenuItem("Copy");  
popupMenu.add(copyItem);  
  
pasteItem = new JMenuItem("Paste");  
popupMenu.add(pasteItem);  
  
popupMenu.addSeparator();  
  
helpItem = new JMenuItem("Help");  
popupMenu.add(helpItem);  
  
// 6.Add the mouse listener to the content pane.  
PopupMenuListener pml = new PopupMenuItemListener();  
textArea.addMouseListener(pml);  
}  
  
// 7.Mouse listener class.  
class PopupMenuItemListener extends MouseAdapter {  
    public void mousePressed(MouseEvent me) {  
        showPopup(me);  
    }  
  
    public void mouseReleased(MouseEvent me) {  
        showPopup(me);  
    }  
  
    private void showPopup(MouseEvent me) {  
        if (me.isPopupTrigger()) {  
            popupMenu.show(me.getComponent(),  
                           me.getX(), me.getY());  
        }  
    }  
}  
  
// 8.Window listener.  
class FrameClosing extends WindowAdapter {  
public void windowClosing(WindowEvent e) {  
    System.exit(0);  
}
```

```

    }

    // 9. The main method.
    public static void main(String[] args) {
        TJPopupMenu frame = new TJPopupMenu();
        frame.setTitle("TJPopupMenu");
        frame.setSize(350,150);
        frame.setVisible(true);
    }
}

```

### 代码详析

清单 13.4 创建了包含一个文本区域的应用程序框架 `TJPopupMenu`。当单击鼠标右键时，一个弹出式菜单就显示在鼠标单击处。

在 `TJPopupMenu` 类内，通过使用三种类型的 Swing 组件：`JPopupMenu`、`JMenuItem` 和 `JTextArea` 将某些对象作为它的数据成员来声明。剩下的数据成员 `container` 作为该框架的内容窗格的一个引用。

构造函数 `TJPopupMenu` 起主要作用。片段 - 1 注册了一个带有框架的窗口监听程序，当框架被关闭时仍包含被执行的代码。片段 - 2 得到该框架的内容窗格的句柄。而后的片段 - 3 创建了一个文本区域对象并将其加到该框架上。

片段 - 4 创建了一个弹出式菜单对象，片段 - 5 创建了一些菜单项如保存（Save）、剪切（Cut）、复制（Copy）、粘贴（Paste）和帮助（Help）。这些菜单项已经通过调用 `popupMenu` 对象中的 `add()` 方法加入到该弹出式菜单里。`add()` 方法将每一个菜单项作为它的一个参数。弹出式菜单也可使用 `addSeparator()` 方法加上分隔线。

片段 - 6 创建了一个叫做 `PopupMenuListener` 的鼠标监听程序对象。片段 - 7 定义了对应的类。然后鼠标监听程序被注册为文本区域对象。注意该文本区域对象是为了出现弹出式菜单必须单击的组件。在 `PopupMenuListener` 类的内部，`showPopup()` 方法包含在鼠标被按下或释放时要执行的代码。为了显示弹出式菜单，可以调用菜单对象中的下列方法：

```
Void show(Component component, int x, int y)
```

## 13.6 工具条

工具条就是一个显示一组动作、命令或功能控件的组件。一般说来，工具条中的组件都是按钮。工具条通常被放在靠近小程序或框架上部边缘的地方。

工具条对显示要频繁使用的组件非常有用。在工具条中也可以显示一些关键功能例如保存（Save）命令，以使用户能快速发现它们。工具条可以被设为可移动的或浮动的。浮动的工具条可以作为一个调色盘被引用。为了创建一个工具条，可以使用表示工具条的类 `JToolBar` 类。它存储在 `javax.swing` 程序包内。`JToolBar` 类是由 `JComponent` 类扩展的轻量级组件，如图 13-9 所示。

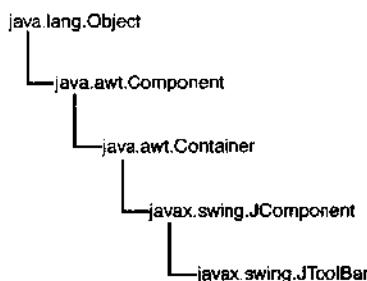


图 13-9 JToolBar 的类层次结构

### 13.6.1 JToolBar 构造函数

JToolBar 类只有一个有效的构造函数，而且不含参数：

```
public JToolBar()
```

为了创建一个工具条并加上按钮，典型的做法如下：

```
JToolBar toolBar = new JToolBar();
JButton button = new JButton(); toolBar.add(button);
```

也可以通过使用方法 add (Action 动作) 加上一个按钮，该按钮为工具条分配一个 Action 类型的动作。此方法返回一个按钮对象。

### 13.6.2 工具条分隔线

工具条指定分隔线在工具条组件间提供了分离空间，以突出指定的功能。可以调用工具条对象中的 addSeparator () 方法来加上分隔线。

**说明：**

工具条所用的分隔线不是使用类 JSeparator 创建的普通分隔线。工具条分隔线由类 Separator 表示，它是 JToolbar 的内部类。

### 13.6.3 JToolBar 代码示例

清单 13.5 演示了 JToolBar 类的一些功能。该例程序显示了嵌入一些带图标按钮的工具条（见图 13-10）。

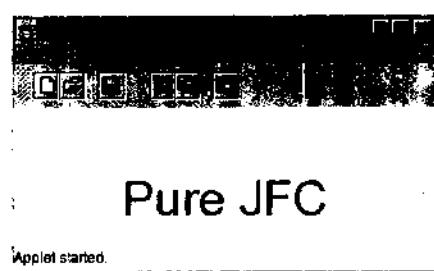


图 13-10 显示一个水平工具条的小程序

已创建的这个工具条是可浮动的，即工具条可以被拖到屏幕上的任何位置。如果想要使工具条不是可浮动的或将其容器固定在指定位置，那么执行下面的 set 方法（将其参数值设为 false）即可：

```
JToolBar toolBar = new JToolBar();
toolBar.setFloatable(false);
```

清单 13.5 带有按钮和图标的 JToolBar (TJToolBar.java)

```
/*
 * < Applet code = TJToolBar width = 350 height = 150 >
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;

public class TJToolBar extends JApplet {
    //1. Create some icons for buttons.
    Icon newIcon = new ImageIcon("new.gif");
    Icon openIcon = new ImageIcon("open.gif");
    Icon saveIcon = new ImageIcon("save.gif");
    Icon cutIcon = new ImageIcon("cut.gif");
    Icon copyIcon = new ImageIcon("copy.gif");
    Icon pasteIcon = new ImageIcon("paste.gif");
    public void init() {
        //2. Get the handle on the applets' content pane.
        Container container = this.getContentPane();
        //3. Create a toolbar object.
        JToolBar toolBar = new JToolBar();
        //4. Set the margin between toolbar border and its comps.
        toolBar.setMargin(new Insets(5,5,5,5));
        //5. Create some toolbar buttons.
        JButton button1 = new JButton(newIcon);
        JButton button2 = new JButton(openIcon);
        JButton button3 = new JButton(saveIcon);
        JButton button4 = new JButton(cutIcon);
        JButton button5 = new JButton(copyIcon);
        JButton button6 = new JButton(pasteIcon);
        //6. Add the buttons to the toolbar with separators.
        toolBar.add(button1);
        toolBar.add(button2);
        toolBar.addSeparator();
        toolBar.add(button3);
```

```
toolBar.addSeparator();
toolBar.addSeparator();
toolBar.add(button4);
toolBar.add(button5);
toolBar.addSeparator();
toolBar.add(button6);

//7. Make a panel with a label.
JPanel panel = new JPanel();
JLabel label = new JLabel("Pure JFC", JLabel.CENTER);
label.setPreferredSize(new Dimension(350,100));
label.setBackground(Color.white);
label.setFont(new Font("Dialog", Font.PLAIN, 40));
label.setOpaque(true);
panel.add(label);

//8. Add these components to the applet.
container.add(toolBar, BorderLayout.NORTH);
container.add(panel, BorderLayout.CENTER);
}
}
```

## 代码详析

在代码的片段 - 1 中，小程序 TJToolBar 将一系列图标作为小程序的数据成员。这些图标将显示在后面创建的按钮上。在 init () 方法里，片段 - 2 可获取小程序内容窗格的句柄。

片段 - 3 创建了一个工具条对象，片段 - 4 在工具条边框及其组件间设置了一些的空间。片段 - 5 创建了六个按钮并把它们加到工具条上。注意那些已经由工具条分隔线分隔开来的按钮。

片段 - 7 中创建的窗格及其内容简单地填在小程序中。片段 - 8 把工具条和窗格加到小程序中。工具条已被加到边框的上方。

### 提示：

为了使工具条在浮动的情况下正常工作，可以使用边界布局并把工具条加到四周的某一边上。

图 13-10（见前面）显示了该程序的输出结果。注意工具条一开始就被放到小程序窗口的顶部。该工具条包含四组由工具条分隔线划分的按钮。第一组和第三组每一组包括两个按钮，第二组和第四组每一组只有一个按钮。选择第一组中的按钮可以新建一个文档或打开已存在的文件。第二组显示的是保存文件的按钮。第三组和第四组的按钮可执行剪切、复制和粘贴功能。

图 13-11 显示的小程序中，工具条被再定位在其显示域的左边。工具条的重定位可以通过在工具条上的位置按下鼠标后拖动工具条来实现。

图 13-12 显示的小程序中，工具条完全移到了小程序的外边。

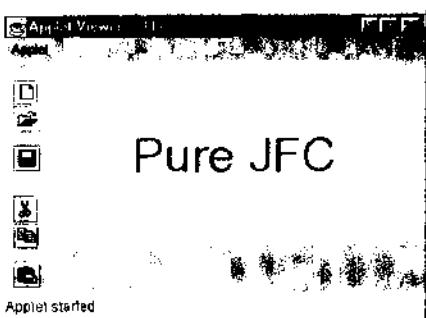


图 13-11 显示垂直放置的工具条的小程序

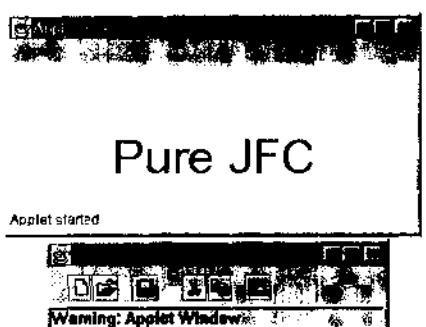


图 13-12 显示工具条位于边界外边的小程序

## 13.7 多控件的公共操作

大多数应用程序都有执行同一功能的多组控件。例如，在字处理程序中，剪切命令通常可以通过不同的控件如菜单项、工具条按钮和弹出式菜单中的菜单项来执行。在 Swing 程序中，为了使两个或更多的控件执行一个动作代码，没必要创建多个监听程序并将它们注册为各自的事件源。Swing 库提供了处理这些公共操作的简化系统。

具有两个或多个组件的一个公共操作由实现 javax.swing.Action 接口的监听程序类表示。这个 Action 接口扩展了 java.awt.event.ActionListener 接口。除了 actionPerformed() 方法以外，这个接口还包括几个抽象方法，用来为按钮或菜单项显示文本串或图标，使能否具有公共功能等等。这样所有需要执行公共操作的组件控件，都可以简单地作为通常监听程序内的 actionPerformed() 方法激活事件。为了将 Action 对象注册为对应的源，可以简单地使用 add() 方法，其中 Action 类型的监听程序对象作为其参数值。例如，JToolBar 类的 add 方法（Action 动作）将一个 JButton 对象加到工具条上，其中 Action 类型的监听程序对象注册为按钮。在其他类（如 JMenu 和 JPopupMenu）所支持的方法中也有类似的方法。

### 13.7.1 TAction 代码示例

清单 13.6 对文本区域上的一小段文本执行剪切、复制和粘贴操作。该程序被创建为应

用程序（如图 13-13 所示），因为小程序不能和系统剪贴板进行交互。剪切、复制和粘贴功能的组件控件在三个不同的源中都是有效的：编辑（Edit）菜单中的菜单项、弹出式菜单中的菜单项和工具条中的按钮。操作这些控件，既可以移动信息也可以从剪贴板上读版信息。

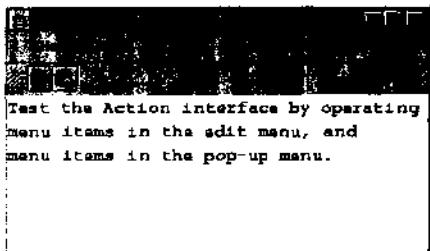


图 13-13 处理多控件公共操作的演示

**清单 13.6 处理多控件公共操作 (*Taction.java*)**

```
// Demonstrates how to handle actions common to multiple
// controls such as menu items (in menus and pop-up menus)
// and toolbar buttons.

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

// The Swing frame class
public class TAction extends JFrame {
    // Declare the data members
    CutAction cutAction = null;
    PasteAction pasteAction = null;
    Container container = null;
    JPopupMenu popupMenu = null;
    JTextArea textArea = null;

    // Icons to be used over the toolbar buttons
    // and menu items
    Icon cutIcon = new ImageIcon("cut.gif");
    Icon pasteIcon = new ImageIcon("paste.gif");

    public TAction() {
        // 1. Call the super with the prescribed title.
        super("TAction Frame");

        // 2. Get the handle on the frame's content pane.
        container = this.getContentPane();

        // 3. Create a panel and add it at the top of the frame.
        JPanel panel = new JPanel(false);
    }
}
```

```
container.add(panel, BorderLayout.NORTH);
panel.setLayout(new GridLayout(2,1));
// 4.Create a menu bar and attach it to the panel.
JMenuBar menuBar = new JMenuBar();
panel.add(menuBar);
// 5.Create the File and Edit menus.
JMenu fileMenu = new JMenu("File");
JMenu editMenu = new JMenu("Edit");
// 6.Add the menus to the menu bar.
menuBar.add(fileMenu);
menuBar.add(editMenu);
// 7.Create a toolbar and attach it to the panel.
JToolBar toolBar = new JToolBar();
panel.add(toolBar);
// 8.Create the Action objects for the cut and paste operations
cutAction = new CutAction("Cut",cutIcon);
pasteAction = new PasteAction("Paste",pasteIcon);
// 9.Add the sources of common actions to their parents.
JMenuItem actionCutItem = editMenu.add(cutAction);
JMenuItem actionPasteItem = editMenu.add(pasteAction);
JButton cutButton = toolBar.add(cutAction);
JButton pasteButton = toolBar.add(pasteAction);
cutButton.setText("");
// Tool bar buttons need not have
pasteButton.setText("");
// the labels
// 10.Create and add the text area to the content pane.
textArea = new JTextArea ("Test the Action interface"
+ "by operating the tool bar buttons."
+ "\nmenu items in the edit menu, and"
+ "\nmenu items in the pop-up menu.");
textArea.setFont(new Font("Monospaced",Font.PLAIN,14));
container.add(textArea);
// 11.Create a pop-up menu and add it to the text area.
popupMenu = new JPopupMenu("Test Popup Menu");
JMenuItem cutPopupMenuItem = popupMenu.add(cutAction);
JMenuItem pastePopupMenuItem = popupMenu.add(pasteAction);
// 12.Create a pop-up menu listener and register it with text area.
PopupMenuListener pml = new PopupMenuItemListener();
textArea.addMouseListener(pml);
// 13.To close the frame.
```

```
addWindowListener(new WindowEventHandler());
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
// 14. Set the size of the frame and display it.
setSize(350,200);
show();
}

// 15. Cut Action class that extends AbstractAction.
class CutAction extends AbstractAction {
    public CutAction(String label,Icon icon) {
        super(label,icon);
    }

    public void actionPerformed(ActionEvent ae) {
        textArea.cut();
    }
}

// 16. Paste Action class that extends AbstractAction.
class PasteAction extends AbstractAction {
    public PasteAction(String label,Icon icon) {
        super(label,icon);
    }

    public void actionPerformed(ActionEvent ae) {
        textArea.paste();
    }
}

// 17. The popup menu listener.
class PopupMenuListener extends MouseAdapter {
    public void mousePressed(MouseEvent me) {
        showPopup(me);
    }

    public void mouseReleased(MouseEvent me) {
        showPopup(me);
    }

    private void showPopup(MouseEvent me) {
        if (me.isPopupTrigger()) {
            popupMenu.show(me.getComponent(),
                          me.getX(),me.getY());
        }
    }
}

// 18. Listener class to close the frame.
```

```
class WindowEventHandler extends WindowAdapter {  
    public void windowClosing(WindowEvent evt) {  
        System.exit(0);  
    }  
}  
  
// 19. The main method.  
public static void main(String[] args) {  
    TAction actionFrame = new TAction();  
}
```

## 代码详析

TAction 框架类一开始就声明作为数据成员的动作引用、其他组件和图标。在片段 - 1 的 TAction 构造函数内部，其父框架构造函数叫做框架标题。片段 - 2 为框架的内容窗格得到句柄。片段 - 3 创建了一个两行一列的窗格并将它连到内容窗格上。

片段 - 4 创建了一个菜单条并将它连到窗格上。片段 5 和片段 6 创建了文件 (File) 菜单和编辑 (Edit) 菜单并将它们加到菜单条上，片段 - 7 将菜单条加到窗格上。

片段 8 和片段 9 定义了动作对象 (监听程序)，它包括执行剪切和粘贴操作的代码。将这些动作对象传递到 JMenu 和 JToolBar 的 add () 方法里，就创建了相应的菜单项和按钮。菜单项出现在与已注册的剪切和粘贴动作监听程序有关的菜单 (Edit) 里。同样地，按钮出现在带由菜单项注册的剪切和粘贴操作的工具条中。

片段 - 10 给框架创建并增加一个文本区域组件 (带有指定文本)。片段 - 11 创建了一个弹出式菜单，然后调用以剪切和粘贴动作对象 (监听程序) 作为参数的 add () 方法来创建它的菜单项。

片段 - 12 给出了显示弹出式菜单的监听程序代码，该监听程序已由文本区域注册。片段 - 13 显示了关闭框架的语句，片段 - 14 将框架重新定义为所需的尺寸并显示它。

片段 15 和片段 16 显示了由 AbstractAction 类扩展的监听程序类，它实现 Action 接口。注意 actionPerformed () 方法包含了在操作相应的控制小配件时将要执行的代码。

片段 - 17 实现单击右键时将要显示弹出式菜单的监听程序代码。片段 - 18 是关闭框架的监听程序类。片段 - 19 是创建 TAction 框架的主方法。

## 13.8 工具提示

工具提示是当用户将鼠标指针暂停在用户接口组件上时弹出的文本，它通常包括该组件的“这是什么”或“怎样操作”的信息。当一个组件没有说明文字时，工具提示就特别有用。例如，在工具条的按钮上只有图标的情况下，就需要工具提示来指明它们的用途。

### 提示:

工具提示不要太长。而且，基于人类交流的经验，当鼠标指针停在组件上时，工具提示出现的最佳时间是 700 微秒左右。

在 Swing 中，给组件实现标准的工具提示非常容易。类 JComponent 压缩了必要的 API。因为 Swing 组件是 JComponent 的扩展类，所以可以调用该组件对象中的 ToolTip 方法。下面列出的是 JComponent 中的 API 方法：

```
public void setToolTipText(String toolTipText)
```

此方法给组件对象设计了一个工具提示（ToolTips）文本，在组件中该方法以 toolTipText 值被调用。

```
public String getToolTipText()
```

此方法接收了前面方法设计的组件的工具提示（ToolTips）文本。

```
public String getToolTipText(MouseEvent evt)
```

此方法返回（已经设计好的）基于鼠标操作位置的组件的工具提示（ToolTips）文本。这在具有许多部分的组件（如 JTree、JTable、JTabbedPane 等等）里是有用的，它们需要由工具提示来指明。

```
public Point getToolTipLocation(MouseEvent evt)
```

此方法返回了在接收的组件坐标系统中工具提示（ToolTip）的位置。如果返回的是 null（缺省执行），它就会自动选择一个合适的位置。

客户工具提示（ToolTips）很少需要程序员来设计。为了创建有特定外观的工具提示（ToolTip），需要将 JToolTip 类实例化。因此，我们使用 JComponent 方法。

```
public JToolTip createToolTip()
```

类 JToolTip 存储在 javax.swing 程序包里。有关类 JToolTip 的方法的完整列表请见附录 A “JFC Swing 快速参考”。

### 13.8.1 JToolTip 代码示例

清单 13.7 显示了带有工具条的小程序（见图 13-14），工具条中包含执行不同操作的按钮。这些按钮只有图标，但你可以给它们加上工具提示来指明它们的目的。

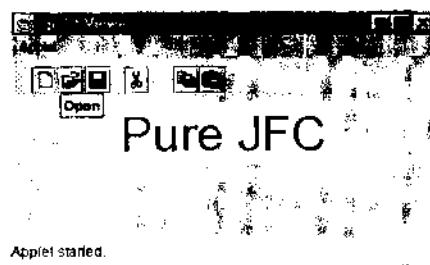


图 13-14 工具提示（TTotlTips）的演示

清单 13.7 ToolTip.java

```
// Demonstrates the Swing tool tips
```

```
/*
 * <Applet code=TToltip width=350 height=150>
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;

public class TToltip extends JApplet {
    // Create some icons for buttons.
    Icon newIcon = new ImageIcon("new.gif");
    Icon openIcon = new ImageIcon("open.gif");
    Icon saveIcon = new ImageIcon("save.gif");
    Icon cutIcon = new ImageIcon("cut.gif");
    Icon copyIcon = new ImageIcon("copy.gif");
    Icon pasteIcon = new ImageIcon("paste.gif");

    Icon[] icons = {newIcon, openIcon, saveIcon,
                    cutIcon, copyIcon, pasteIcon};
    String[] toolTips = {"New", "Open", "Save",
                         "Cut", "Copy", "Paste"};

    public void init() {
        // 1. Get the handle on the applet's content pane.
        Container container = this.getContentPane();

        // 2. Create a toolbar object.
        JToolBar toolBar = new JToolBar();

        // 3. Set the margin between the toolbar border and its comps.
        toolBar.setMargin(new Insets(5, 5, 5, 5));

        // 4. Create some buttons and ToolTips
        // and add them to the toolbar. Also add
        // the separators.
        for (int i = 0; i < 6; i)
        {
            JButton button = new JButton(icons[i]);
            button.setToolTipText(toolTips[i]);
            toolBar.add(button);
            if (i == 2 || i == 3 || i == 5)
                toolBar.addSeparator();
            if (i == 3)
                toolBar.addSeparator();
        }

        // 5. Make a panel with a label.
        JPanel panel = new JPanel();
```

```
JLabel label = new JLabel("Pure JFC",JLabel.CENTER);
//label.setPreferredSize(new Dimension(350,150));
label.setBackground(Color.white);
label.setFont(new Font("Dialog",Font.PLAIN,40));
label.setOpaque(true);
panel.add(label);

// 6.Add the toolbar and panel to the applet.
container.add(toolBar,BorderLayout.NORTH);
container.add(panel,BorderLayout.CENTER);
}

}
```

### 代码详析

上述小程序 TToolTip 从定义图标列表为数据成员开始，这些图标都用在工具条按钮上。在 init () 方法里，片段 -1 得到小程序内容窗格的句柄。片段 2 和片段 3 创建了工具条对象，在它的边界和组件间有特定的空白区域。

然后，片段 -4 创建了有指定图标的六个按钮并将它们附加到工具条上。注意已经使用 setToolTipText () 方法为这些按钮设计了工具提示（ToolTips）。片段 -5 创建了一个带有标签的窗格且把它填在小程序的中央。片段 -6 将工具条和窗格加到小程序的内容窗格上。

# 第 14 章 对话框和选项窗格

对话框是在父窗口里弹出的子窗口，用来显示反馈信息、确认操作、接收输入、显示工具等等。Swing 库支持名为 `JDialog` 的核心类来创建定制对话框。

另外，使用 `JOptionPane` 支持的静态方法，还可以创建更少代码的选项窗格（也叫做标准对话框）。选项窗格可以显示信息、确认动作、接收输入或组合这些功能。

这一章首先详细解释了实现定制对话框的过程，然后一步一步创建选项窗格。

## 14.1 对话框

Swing 库中的对话框由 `JDialog` 类表示，它是早期 AWT 对话框 `java.awt.Dialog`（见图 14-1）的扩展类。`JDialog` 是创建 Swing 对话框的主要的类，并且作为其父类是一个重量级组件。`JDialog` 类对创建定制对话框特别有用。内容窗格作为加到 Swing 对话框中的所有组件的容器来使用，可以用下面的方法将组件加到内容窗格里：

```
DIALOGBox.getContentPane().add(someComponent);
```

由于内容窗格在对话框的背景里，所以缺省的布局管理程序采用的是边界布局。

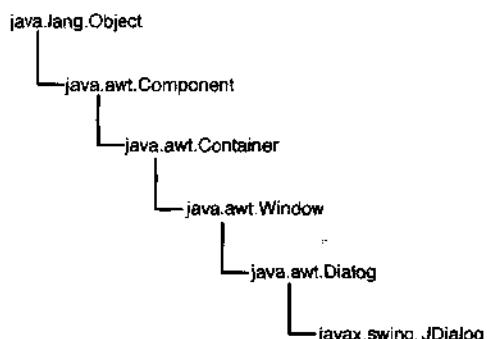


图 14-1 Swing 对话框的类层次结构

### 14.1.1 `JDialog` 构造函数

要创建一个定制对话框，可以使用由 `JDialog` 类支持的下列构造函数：

- `public JDialog()`
- `public JDialog(Frame owner)`
- `public JDialog(Frame owner, boolean modal)`
- `public JDialog(Frame owner, String title)`
- `public JDialog(Frame owner, String title, boolean modal)`

```
*public JDialog()
```

创建了一个没有标题和父框架的非模态对话框；一个隐藏对话框的缺省设置为它的父框架。一个非模态对话框甚至在被激活时也允许在其父窗口内输入信息，而模态对话框只要存在，就拒绝在应用程序的所有窗口内输入信息。

```
public JDialog(Frame owner)
```

还创建了一个没有任何标题的非模态对话框，但它有指定框架作为它的父框架。

```
public JDialog(Frame owner,boolean modal)
```

创建了一个指定类型的对话框（模态对话框或非模态对话框），它有指定的父框架，但是没有标题。

```
public JDialog(Frame owner, String title)
```

创建了一个有指定标题和指定父框架的非模态对话框。

```
public JDialog(Frame owner, String title,boolean modal)
```

创建了一个指定类型的对话框（模态对话框或非模态对话框），它有指定的标题和指定的父框架。

### 14.1.2 JDialog 代码示例

清单 14.1 显示了一个带对话框的小程序，在此对话框中，当用户按下按钮时就会显示当前的日期和时间。这个程序还演示了怎样将 Swing 对话框连到小程序而不是框架上。该程序的输出结果显示在图 14-2 中。

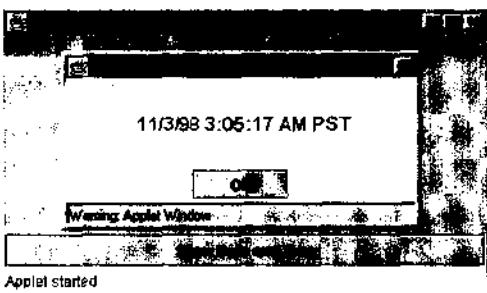


图 14-2 用 JDialog 定制日期和时间的小程序

清单 14.1 用 JDialog，来自定制日期和时间 (TJDialog.java)

```
/*
 * <Applet code = TJDialog width = 400 height = 200 >
 * </Applet >
 */
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;
import java.util.*;
import java.text.*;

public class TJDialog extends JApplet implements ActionListener {
    // 1. Declare the underlying frame to which the
    // dialog is attached.
    Frame frame = null;

    public void init() {
        // 2. Get a handle on the applet's content pane.
        Container container = this.getContentPane();

        // 3. Retrieve the underlying frame of the applet.
        // See snippet - 5 for the details of getParentFrame().
        frame = getParentFrame(container);
        // 4. Add a button to bring up the dialog box.
        JButton button = new JButton("Show Date and Time");
        button.addActionListener(this);
        container.add(button, BorderLayout.SOUTH);
    }

    // 5. Retrieve the frame of the applet's container.
    public Frame getParentFrame(Component comp) {
        if (comp instanceof Frame) return (JFrame)comp;
        for (Component c = comp; c != null; c = c.getParent()) {
            // Check only for the Frame, not the JFrame!
            if (c instanceof Frame)
                return (Frame)c;
        }
        return null; // if there are no frames
    }

    // 6. Display the date and time for the current locale.
    public String showDateAndTime() {
        DateFormat dateAndTimeFormat =
            DateFormat.getDateInstance(
                DateFormat.SHORT, DateFormat.LONG);
        String dateAndTime = dateAndTimeFormat.format(new Date());
        return dateAndTime;
    }

    // 7. When the button is displayed on the applet is operated...
    public void actionPerformed(ActionEvent ae) {
        String date_time = showDateAndTime();
        // When the applet's button is clicked, create the Swing
    }
}
```

```
// dialog box and display it.  
DateDialog dialog = new DateDialog(frame,date_time,true);  
}  
}  
  
// 8.Create a custom dialog box that extends JDialog  
// with a title and message.  
class DateDialog extends JDialog implements ActionListener {  
    Icon icon = new ImageIcon("clock.gif");  
  
    public DateDialog(Frame frame, String msg, boolean modal) {  
        // 9.Call the JDialog constructor.  
        super(frame, "Date & Time Dialog", modal);  
  
        // 10.Set the background color of DateDialog.  
        this.getContentPane().setBackground(Color.white);  
  
        // 11.Create a confirmation button.  
        JButton button = new JButton("OK");  
        button.setPreferredSize(new Dimension(60,25));  
        button.addActionListener(this);  
  
        // 12.Create a label to display the time and date.  
        JLabel label = new JLabel(msg,icon,JLabel.CENTER);  
        label.setFont(new Font("Dialog",Font.BOLD,22));  
  
        // 13.Add the message label and OK button to the dialog box.  
        this.getContentPane().add(label,BorderLayout.CENTER);  
  
        Box box = Box.createHorizontalBox();  
        box.add(box.createHorizontalGlue());  
        box.add(button);  
        box.add(box.createHorizontalGlue());  
        this.getContentPane().add(box,BorderLayout.SOUTH);  
  
        // 14.Resize the dialog box and position it at the center of  
        // the applet.  
        setSize(300,150);  
        Dimension dialogDim = getSize();  
        Dimension frameDim = frame.getSize();  
        Dimension screenSize = getToolkit().getScreenSize();  
        Point location = frame.getLocation();  
        location.translate(  
            (frameDim.width - dialogDim.width)/2,  
            (frameDim.height - dialogDim.height)/2);  
        location.x = Math.max(0,Math.min(location.x,  
            screenSize.width - getSize().width));  
        location.y = Math.max(0,Math.min(location.y,
```

```

        screenSize.height - getSize().height));
        setLocation(location.x, location.y);
        this.show();
    }

    // 15. When the OK button on the dialog box is clicked.
    public void actionPerformed(ActionEvent ae) {
        this.dispose();
    }
}

```

### 代码详析

代码片段 – 1 首先声明了小程序的容器中在下面的框架，该框架用做对话框的父框架，在 init () 方法里，片段 – 2 得到小程序内容窗格的句柄。

片段 – 3 获取下面的框架对象，并初始化小程序的数据成员。片段 – 5 显示了在片段 – 3 中调用方法的详情。需要递归地得到小程序容器的父窗口，并检查它们看是否恰有一个是框架。如果发现了一个父框架，就将它连到该对话框上。

片段 – 4 为小程序加上一个按钮并为其注册了动作监听程序。在动作监听程序（片段 – 7）里，可获得当前日期和时间并创建 Swing 对话框显示它们。显示在片段 – 6 中的方法从系统中接收了当前日期和时间，再按照当前位置将其格式化。

片段 – 8 使用一个 Swing 标签创建了一个显示日期和时间的 Swing 对话框。该对话框含有一个确认按钮，单击它将关闭对话框。由 JDialog 扩展的内部类 DateDialog 执行一个动作监听程序。在片段 – 9 中，使用 super () 语句创建了一个 JDialog 类型的模态对话框，它被连到已经显示了日期和时间的小程序下面的对话框上。

片段 11 和片段 13 创建了一个按钮和附加在对话框上的标签。片段 14 是在小程序的中心显示对话框的代码。为了做到这一点，需要知道屏幕尺寸和对话框尺寸，然后用简单的坐标变换将对话框居中显示。片段 15 是对话框中确认按钮的动作监听程序。当单击该按钮时，就关闭对话框。

## 14.2 选项窗格

上一节解释了怎样使用 JDialog 类来实现对话框。必须明确提供创建 JLabel 的代码，显示何信息、将对话框与其父框架或小程序关联、以及将对话框定位在其父框架内部的代码。用 JDialog 来实现所有的对话框是很麻烦的，幸运的是，Swing 库中包含了一种叫做 JOptionPane 的中间件类，用其静态方法只需几行代码就能创建选项窗格（即标准对话框）。

选项窗格可以用来显示反馈信息、确认或为应用程序输入信息。还有一个叫做选项对话框的选项窗格具有以上选项窗格的所有特征。

### 14.2.1 JOptionPane 类

JOptionPane 是包含创建模态化选项窗格的很方便的类。JOptionPane 是 JComponent 的

直接子类，它存储在 `javax.swing` 程序包内。`JOptionPane` 静态方法用来创建和显示 `JDialog` 类型的选项窗格。这些静态方法可以有效地以重载形式创建带有所希望参数列表的选项窗格。`JOptionPane` 也支持创建背景容器是内部框架的选项窗格的方法。

### 14.2.2 JOptionPane 构造函数

你可以创建一个选项窗格对象，它的参数与要创建的对话框参数相符。利用创建好的选项窗格对象，可以调用 `createDialog()` 方法创建一个对话框。下面列出的是可以有效创建 `JOptionPane` 类型对话框的构造函数：

```
public JOptionPane()
```

创建了带有测试 `JOptionPane` 消息的 `JOptionPane` 类型的对象。

```
public JOptionPane(Object message)
```

创建了一个 `JOptionPane` 类型的实例，它使用纯文本消息和用户接口发表的缺省选项显示指定消息。

```
public JOptionPane(Object message, int messageType)
```

创建了一个用指定的纯文本消息 (`messageType`) 和缺省选项显示消息的例子。这个消息类型参数可以取任何一个 `JOptionPane` 类型的字段 (`ERROR_MESSAGE`、`INFORMATION_MESSAGE`、`WARNING_MESSAGE`、`QUESTION_MESSAGE` 或 `PLAIN_MESSAGE`) 的值。

```
public JOptionPane(Object message, int messageType, int optionType)
```

创建了一个显示指定消息类型 (带有以上给出的字段中的任何一个) 和指定选项类型 (如 `DEFAULT_OPTION`、`YES_NO_OPTION`、`YES_NO_CANCEL_OPTION` 和 `OK_CANCEL_OPTION`) 的 `JOptionPane` 类型的例子。除了图标对象是特殊指定的以外，第五个构造函数类似于第四个构造函数：

```
public JOptionPane(Object message, int messageType,
                   int optionType, Icon icon)
```

```
public JOptionPane(Object message, int messageType,
                   int optionType, Icon icon,
                   Object[] options)
```

它创建了一个显示有指定的消息类型、选项类型、图标和选项的消息的 `JOptionPane` 类型的例子。

```
public JOptionPane(Object message, int messageType,
                   int optionType, Icon icon,
                   Object[] options,
                   Object initialValue)
```

创建了一个 `JOptionPane` 类型的例子，显示有指定的消息类型、选项类型、图标和选项

的消息，且有指定的初始选项。

上面的参数 options 是 Object 类型的对象数组。这种参数支持任何组件，不管是对话框里 Swing 按钮，还是定制标签的 Swing 按钮都可以。大体上，这种数组要么是组件数组，要么是字符串数组。如果将组件数组作为选项值来传递，那么这些组件就会显示在对话框里，而不是正规的按钮上。如果将（String 类型的）字符串标签数组作为选项值来传递，带有前述标签的 Swing 按钮就显示在对话框上。

#### 说明:

选项 (options) 对象可以既包含一些其他组件也可以包含字符串。组件对象直接加在窗格上；字符串对象在 JButton 中被打包然后加在窗格上。如果提供组件，则必须确保在单击组件时调用 setValue () 方法，为选项窗格设置用户选项。

### 14.2.3 创建选项窗格

在使用前面讨论的构造函数创建 JOptionPane 类型的实例后，就可以进一步使用 JOptionPane 中的“set”方法（详见附录 A：“JFC Swing 快速参考”）来构造选项窗格。

接着，可以使用下面列出的 JOptionPane 中的方法设计一个标准对话框（也作为选项窗格）：

```
public JDialog createDialog(Component parent, String title)
```

这个方法需要本对话框的父框架，还需要对话框的标题。为了创建并显示一个对话框，需要执行下面使用选项窗格对象的代码：

```
JOptionPane optionPane = new JOptionPane();
JDialog dialog = optionPane.createDialog(parent,
                                         title);
dialog.show();
```

也可以通过使用静态“show”方法如 showMessageDialog ()、showConfirmDialog ()、showInputDialog () 和 showOptionDialog () 来直接显示标准对话框（选项窗格）。还有例如 showInternalMessageDialog ()、showInternalConfirmDialog ()、showInternalInputDialog () 和 showInternalOptionDialog () 等方法也支持创建标准对话框。这些对话框包含它们背景中的 JInternalFrame 类型的框架。标准对话框是我们以后几节感兴趣的话题。

### 14.3 消息对话框

消息对话框可用五种方式之一显示纯文本消息，这五种方式是纯文本消息、信息、问题、错误或警告。JOptionPane 类支持使用信息、问题、错误或警告消息的图标来创建指定的消息对话框。每个图标的外观符合应用程序的外观（表 14.1）。

```
public static void showMessageDialog( Component parent, Object message)
```

为了在程序中显示消息对话框，可以直接使用有合适参数表的静态方法 showMessageDi-

alog()。作为选择，也可创建 JOptionPane 对象并调用 createDialog() 方法。createDialog() 方法返回调用该对象的 show() 方法，来显示的 Jdialog 类型的对话框。稍后将给出一个例子演示后一种方法。

消息类型	<i>Java LAF</i>	<i>Motif LAF</i>	<i>Windows LAF</i>
ERROR_MESSAGE			
INFORMATION_MESSAGE			
PLAIN_MESSAGE	无	无	无
QUESTION_MESSAGE			
WARNING_MESSAGE			

下面是 showMessageDialog() 的重载方法：

```
public static void showMessageDialog( Component parent, Object message)
```

这个方法创建并显示了一个消息对话框，在对话框底部，有一个消息确认按钮（OK 按钮）。该对话框的缺省标题为“Confirm”，指明用户可以单击 OK 按钮确认消息。这个确认标题（以问题形式出现的）并非确认什么重要事件；它只是使用户在读过消息以后单击按钮来关闭对话框。参数 parent 是使对话框在其中显示的框架。如果 parent 是空的（null），或者 parent 没有框架，就使用 DefaultFrame，这时对话框显示在屏幕的中央。下一个参数是要显示的消息字符串。

```
public static void showMessageDialog(
    Component parent,
    Object message,
    String title,
    int messageType)
```

此方法显示了一个带有缺省图标的的消息对话框，该图标是由 messageType 参数 ERROR\_MESSAGE、INFORMATION\_MESSAGE、WARNING\_MESSAGE、QUESTION\_MESSAGE 或 PLAIN\_MESSAGE 决定的。也可以指定显示在对话框顶部的标题。

```
public static void showMessageDialog(
    Component parent,
    Object message,
    String title,
    int messageType,
    Icon icon)
```

此方法引出一个显示指定消息和标题的对话框。你可以指定要显示的图标，而不是由参

数 messageType 决定的缺省图标。

#### 说明:

参数 messageType 依赖其类型决定对话框的外观和内容。例如，显示在对话框里的缺省图标是由参数 messageType 决定的。

### 14.3.1 消息选项窗格代码示例

清单 14.2 创建了一个在系统中显示日期和时间的选项窗格，如图 14-3 所示。该程序使用选项窗格对象创建了消息对话框，并通过调用对话框对象中的 show() 方法显示该对话框。由于这个程序也是显示了日期和时间，正如清单 14.1 中显示的一样，比较这两个程序会发现，使用标准对话框减少了代码的行数。

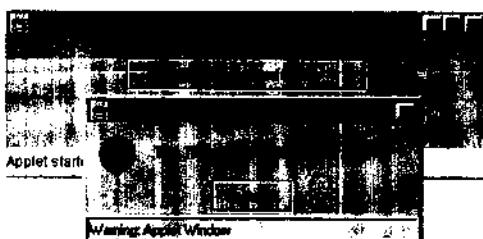


图 14-3 显示消息选项窗格的小程序

清单 14.2 日期和时间消息选项窗格 (TJOptionPane1.java)

```
/*
 * <Applet code=TJOptionPane1 width=400 height=75>
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;

public class TJOptionPane1 extends JApplet {
    Container container = null;

    public void init() {
        // 1. Get a handle on the applet's content pane.
        container = this.getContentPane();

        // 2. Add a box to the content pane.
        Box box = new Box(BoxLayout.X_AXIS);
        container.add(box);

        // 3. Create a button and add it to the box.
        JButton button = new JButton("Show Date and Time");
        box.add(button);
    }
}
```

```
button.setPreferredSize(new Dimension(200,25));
button.addActionListener(new ButtonListener());
box.add(Box.createGlue());
box.add(button);
box.add(Box.createGlue());
}

// 4. Listener class that displays a message dialog box.
class ButtonListener implements ActionListener {
    String title = "Example Message Dialog";
    int messageType = JOptionPane.INFORMATION_MESSAGE;
    int optionType = JOptionPane.DEFAULT_OPTION;
    Object message = null;
    Object[] options = {"Thanks!"};
    String dateAndTime = null;

    public void actionPerformed(ActionEvent e) {
        // 5. Obtain the date and time.
        dateAndTime = showDateAndTime();
        message = "The Time is " + dateAndTime;

        // 6. Create an option pane with specified attributes.
        JOptionPane pane = new JOptionPane(message,
                                             messageType,
                                             optionType,
                                             null,
                                             options);

        // 7. Create and show the dialog box to display the message.
        JDialog dialog = pane.createDialog(container, title);
        dialog.show();

        // 8. Reset the message to null.
        message = null;
    }

    // 9. Obtain the current date and time with the format
    // of the current locale.
    public String showDateAndTime() {
        DateFormat formatter = DateFormat.getDateInstance(
            DateFormat.SHORT,
            DateFormat.LONG);

        String dateAndTime = formatter.format(new Date());
        return dateAndTime;
    }
}
```

## 代码详析

TOptionPane 是利用 JOptionPane 类创建的一个 Swing 小程序，它显示消息选项窗格。此小程序声明了引用它的内容窗格作为数据字段。在 init() 方法内，片段 -1 获得内容窗格对象，并初始化对内容窗格的引用。片段 -2 将一个对话框加到小程序上，片段 -3 将一个按钮加到对话框上。当用户操作此按钮时，就会出现一个显示当前日期和时间的消息对话框。

片段 -4 显示了监听程序类的启始部分。在 actionPerformed() 方法里，片段 5 按照当前位置的格式取得日期和时间。片段 9 详细显示了怎样取得日期和时间。

片段 6 和片段 7 相当有趣。片段 6 创建了一个 JOptionPane 类型的对象，它的参数值是要显示的消息、消息类型、选项类型、图标（此情况下为空（null））和选项。其中要显示的消息是当前的日期和时间，消息类型是 INFORMATION\_MESSAGE，在监听程序类的数据成员列表中有其定义。

注意到此选项窗格使用了一个名为 Thanks! 的按钮，代替了缺省显示的 OK 按钮。为了达到这个目的，此选项窗格是由含有参数选项的构造函数来创建的，其参数选项将一系列对象作为它的值。

片段 7 创建并显示了带有被传递到 JOptionPane 对象的消息对话框。静态方法 createDialog() 将父容器和对话框的标题作为它的参数值。当运行小程序并单击按钮时，就会弹出带有当前日期和时间的对话框，如前面的图 14-3 所示。

## 14.4 确认对话框

对话框不仅可以反馈消息，也可以向用户提出问题。用户可以回答或者单击如 Yes、No 或 Cancel 这样的按钮来确认。为了显示确认对话框，Swing 库支持如下所示的可重载的“show”方法集：

```
public static int showConfirmDialog(
    Component parentComponent,
    Object message)
```

此方法创建并显示了一个有 Yes、No 和 Cancel 选项的模态对话框。此对话框的缺省标题为“Select an Option”。参数 parentComponent 决定了对话框将要连接到的框架。假如父框架的值为空或者 parentComponent 不支持框架，则使用缺省框架且在屏幕中央显示对话框。另一个参数消息是将要显示在对话框中的字符串。当用户选择 Yes、No 或 Cancel 时，此方法分别返回整数 0、1 或 2。

```
public static int showConfirmDialog(
    Component parentComponent,
    Object message,
    String title,
    int optionType)
```

此方法得到一个模态对话框，其中选项的数目由参数 optionType 决定。参数 parentComponent 决定了将要在其中显示对话框的框架。下一个参数消息是将要显示在对话框中的字符串对象。剩下的参数指明了在对话框如 YES\_NO\_OPTION 或 YES\_NO\_OPTION\_CANCEL 中消息是有效的。对话框对用户选择的 Yes、No 或 Cancel 选项分别返回整数 0、1 或 2。

```
public static int showConfirmDialog(  
    Component parentComponent,  
    Object message,  
    String title,  
    int optionType,  
    int messageType)
```

除了前述方法中给出的参数列表以外，本方法还允许指定消息类型。参数 messageType 指明了由常数 ERROR\_MESSAGE、INFORMATION\_MESSAGE、WARNING\_MESSAGE、QUESTION\_MESSAGE 或 PLAIN\_MESSAGE 指定的消息类型。选择上述常数就在对话框里设置了相应的按钮。此方法也对用户选择的 Yes、No 或 Cancel 选项分别返回整数 0、1 或 2。

```
public static int showConfirmDialog(  
    Component parentComponent,  
    Object message,  
    String title,  
    int optionType,  
    int messageType,  
    Icon icon)
```

此方法类似于前一种，除了另有一个参数 icon。这个参数允许在对话框上指定你喜欢的图标。此方法也对最终用户选择的 Yes、No 或 Cancel 选项分别返回整数 0、1 或 2。

#### 14.4.1 确认选项窗格代码示例

清单 14.3 演示了一个确认选项窗格。此小程序使用确认对话框来确认是否显示日期（如图 14-4 所示）。另一方面，如果用户选择了 No 按钮，则只显示当前的时间（图 14-5）。

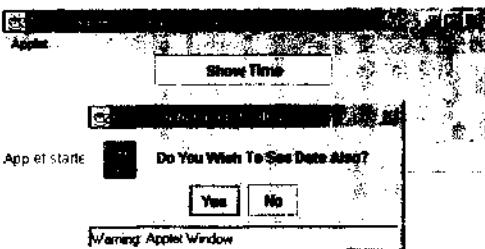


图 14-4 显示确认选项窗格的小程序



图 14-5 单击 Yes 按钮，调用显示时间和日期的消息选项窗格

**清单 14.3 TJOptionPane2.java**

```

/*
 * <Applet code = TJOptionPane2 width = 400 height = 75 >
 * </Applet >
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;

public class TJOptionPane2 extends JApplet {
    Container container = null;

    public void init() {
        // 1. Get a handle on the applet's content pane.
        container = this.getContentPane();

        // 2. Add a box to the content pane.
        Box box = new Box(BoxLayout.X_AXIS);
        container.add(box);

        // 3. Add a button to the box.
        JButton button = new JButton("Show Time");
        button.setPreferredSize(new Dimension(150, 25));
        button.addActionListener(new ButtonListener());
        box.add(Box.createGlue());
        box.add(button);
        box.add(Box.createGlue());
    }

    // 4. The listener class.
    class ButtonListener implements ActionListener {
        // 5. Argument values for the confirmation dialog box.
        Object confirmText = "Do You Wish To See Date Also?";
        String confirmTitle = "Date Confirmation Dialog";
        int optionType = JOptionPane.YES_NO_OPTION;
        int messageType = JOptionPane.QUESTION_MESSAGE;
    }
}

```

```
// 6. Argument values for the message dialog box.  
Object information = null;  
String title = "Message Display Dialog";  
int messageType2 = JOptionPane.INFORMATION_MESSAGE;  
  
// 7. Option selected.  
// If the selection is 'yes', selectedValue = 0;  
// If the selection is 'No', selectedValue = 1;  
int selectedValue;  
  
public void actionPerformed(ActionEvent e) {  
    // 8. Display the confirmation dialog box.  
    selectedValue = JOptionPane.showConfirmDialog(container,  
        confirmText, confirmTitle,  
        optionType, messageType1);  
  
    // 9. Fetch the time or date and time.  
    information = fetchInformation();  
  
    // 10. Display the message.  
    JOptionPane.showMessageDialog(container,  
        information, title,  
        messageType2);  
}  
  
// 11. Returns the time or date and time depending  
// on the Yes or No choice made.  
public String fetchInformation() {  
    DateFormat formatter = null;  
  
    if (selectedValue == 0) { //If it is Yes.  
        formatter = DateFormat.getDateInstance(  
            DateFormat.SHORT,  
            DateFormat.LONG);  
    }  
    else if (selectedValue == 1) { //If it is No.  
        formatter = DateFormat.getTimeInstance(  
            DateFormat.LONG);  
    }  
  
    // Format the time or date and time and return.  
    return(formatter.format(new Date()));  
}
```

## 代码详析

TOptionPane2 是一个 Swing 小程序。init () 方法简单地显示一个按钮，操作它可以看

到当前时间或者当前日期和时间。片段 1 得到小程序的内容窗格的句柄；片段 2 将一个框加到内容窗格上；片段 3 在对话框中增加一个按钮，同时加入对话框的监听程序。

片段 4 启动名为 ButtonListener 的监听程序类。片段 5 和片段 6 显示了此监听程序类的数据成员。这些数据成员是确认对话框和消息对话框的参数值。片段 7 声明了用来存储确认状态如 Yes 或 No 的变量 selecteValue。

在 actionPerformed 方法 () 内，片段 8 创建并显示了已经完成的确认对话框。此确认对话框显示 confirmText 参数的值，它具有由 optionType 参数值指定的 Yes 和 No 按钮。当用户选择其中一个按钮时，选择状态在参数 selecteValue 中存为 0 或 1。片段 9 取得片段 11 中的 fetchInformation () 方法返回的信息。fetchInformation () 方法返回基于 selecteValue 参数值的信息。

在片段 10 里，已经找回的信息显示在消息对话框里。注意，所有使用 JOptionPane 类创建的对话框都是模态对话框。因此，在清单 14.3 中创建的确认对话框处于等待状态，直到用户向其中输入信息。

## 14.5 输入对话框

输入对话框是用框架形式收集用户信息的对话框。这个信息用来设置应用程序的属性。Swing 库通过 JOptionPane 类支持输入对话框，不过这些对话框也是 JDialog 类型的对象。

输入对话框通过一个中间件组件如文本字段、组合框或列表来接收用户输入。在 Swing 里，输入对话框总是伴随着由用户接口管理器决定的组件而出现。调用由 JOptionPane 类支持的静态“show”方法，可以在程序中直接使用输入对话框。下面列出的是这些方法：

```
public static String showInputDialog(Object message)
```

此方法显示了一个要求用户输入的问题消息对话框。此对话框使用了缺省框架，所以被置于屏幕中央。message 指定了要显示的消息。

```
public static String showInputDialog(  
    Component parentComponent,  
    Object message)
```

此方法类似于前面的方法，只是在这个方法里指定了其父框架，此对话框将附加到它的中央。参数 parentComponent 指定了对话框的父框架，message 就是要显示的消息。

```
public static String showInputDialog(  
    Component parentComponent,  
    Object message,  
    String title,  
    int messageType)
```

此方法类似于前面两个方法，只是在这个方法里可以额外指定标题和消息类型。标题就是显示在对话框的标题条中的文本字符串。另一个增加的参数 messageType 取诸如 ERROR\_MESSAGE、INFORMATION\_MESSAGE、WARNING\_MESSAGE、QUESTION\_MESSAGE 等值。

MESSAGE 或 PLAIN\_MESSAGE 的值。这些值决定了对话框图标将带来所要求的对话框的外观。

```
public static Object showInputDialog(  
    Component parentComponent,  
    Object message,  
    String title,  
    int messageType,  
    Icon icon,  
    Object[] selectionValues,  
    Object initialValue)
```

此方法提示用户通过一个模态对话框输入信息，在此对话框里，前面方法中显示的初始选择、可能的选择和其他选项都可以定义。用户可以从 selectionValues 的列表中选择。假如此参数值为空 (null)，那么用户通常可以依靠 JTextField 来选择希望的值。

另一个增加的参数（与前面的方法比较而言）initialSelectionValue，是用来提醒用户的初始值。UI 决定了显示选择值的最有可能的方法，它通常依靠像 JComboBox、JList 或 JTextField 这样的组件。一旦用户输入一个值并确认后，此方法就会将这个值作为字符串返回。假如选择 Cancel 按钮，此方法就返回空值 (null)。

这里的图标就是要在对话框上显示的图标。此方法中的其他参数如 parentComponent、message、title 以及 messageType 正像在前面的方法中给出的和解释的一样。

#### 14.5.1 输入选项窗格示例

清单 14.4 演示了怎样去创建一个输入选项窗格。此程序是一个提示用户通过对话框输入的小程序。通过从排列显示的组合框（见图 14-6）中选定一个选项来提供输入。确认选项后，就会弹出一个显示所需信息的消息对话框（见图 14-7）。

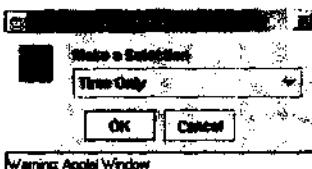


图 14-6 显示输入选项窗格的小程序

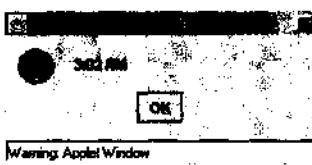


图 14-7 一个消息对话框

**清单 14.4 TJOptionPane3.java**

```
/*
 * <Applet code = TJOptionPane3 width = 400 height = 75 >
 * </Applet>
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.*;

public class TJOptionPane3 extends JApplet {
    Container container = null;

    public void init() {
        // 1. Get a handle on the applet's content pane.
        container = this.getContentPane();

        // 2. Add a box to the content pane.
        Box box = new Box(BoxLayout.X_AXIS);
        container.add(box);

        // 3. Add a button to display the time.
        JButton button = new JButton("Click Me");
        button.setPreferredSize(new Dimension(150, 25));
        button.addActionListener(new ButtonListener());
        box.add(Box.createGlue());
        box.add(button);
        box.add(Box.createGlue());
    }

    // 4. The button listener class.
    class ButtonListener implements ActionListener {
        // 5. Argument values for the option pane dialog box.
        String title1 = "Example Input Dialog";
        String message1 = "Make a Selection";
        int messageType1 = JOptionPane.QUESTION_MESSAGE;
        Icon icon = null;

        Object[] options = {"Date Only", "Time Only",
                           "Date and Time"};
        Object optionSelected = options[1];

        // 6. Selection status of check box.
        Object selectedValue;
        // This field stores the selection made in the check box
        // as a string when you press the 'ok' button. If you press
```

```
// the 'cancel' button, the 'null' value is received.  
  
// 7. Argument values for the message dialog box.  
Object message2 = null;  
String title2 = "Example Message Dialog";  
int messageType2 = JOptionPane.INFORMATION_MESSAGE;  
  
public void actionPerformed(ActionEvent e) {  
    // 8. Display the input dialog box to make a selection.  
    selectedValue = JOptionPane.showInputDialog(  
        container, message1, title1,  
        messageType1,  
        null, options, options[1]);  
  
    // 9. Get the date, time, or date and time.  
    if (selectedValue != null) { // i.e., if not 'cancel'  
        message2 = getInformation();  
  
        // 10. Finally, display the message dialog box.  
        JOptionPane.showMessageDialog(container, message2,  
            title2, messageType2);  
    }  
}  
  
// 11. Method to retrieve the information.  
public String getInformation() {  
    DateFormat formatter = null;  
  
    if (selectedValue == "Time Only") {  
        formatter = DateFormat.getTimeInstance(  
            DateFormat.SHORT);  
    }  
  
    else if (selectedValue == "Date Only") {  
        formatter = DateFormat.getDateInstance(  
            DateFormat.SHORT);  
    }  
  
    else if (selectedValue == "Date and Time") {  
        formatter = DateFormat.getDateTimeInstance(  
            DateFormat.SHORT,  
            DateFormat.SHORT);  
    }  
  
    // Now format the time or date and time and return.  
    String information = formatter.format(new Date());  
    return information;  
}
```

}

### 代码详析

此程序创建了一个叫做 `TOptionPane3` 的小程序。在此小程序的 `init()` 方法里，片段 1 得到小程序内容窗格的句柄。片段 2 和片段 3 通过框架容器将一个按钮加到小程序上。当用户单击按钮时，就会出现一个有多个选择项的对话框。有关的代码在由按钮监听程序代表的类中执行。

片段 4 显示了监听程序类的开始部分。片段 5 取得输入对话框的参数值。参数选项定义了显示在输入对话框上的一组选项，它们以下拉清单形式显示在输入对话框的组合框中。另一个 `optionSelected` 参数以缺省方式指定了显示在组合框里的值。

片段 6 显示被选定并存储的参数值。组合框中下拉列表中选定的值是 `String` 类型的。片段 7 显示了创建消息对话框的参数，当用户通过输入对话框作出选择时就会出现此消息对话框。

在 `actionPerformed()` 方法里，片段 8 给出了怎样显示带有一个消息和含有一系列可选项的组合框。片段 9 找到由方法 `getInformation()` 返回的信息。

#### 说明：

如果用户按下对话框中的 `Cancel` 按钮，输入对话框就返回空值 (`null`)。

片段 10 弹出带有已获取信息的消息对话框。当输入对话框弹出时此程序的输出结果显示在图 14-6 中。一旦用户通过选项完成了输入，就会显示包含所需信息的消息对话框。图 14-7 显示了带有消息对话框的小程序。

## 14.6 选项对话框

选项对话框是拥有所有对话框（如消息对话框、确认对话框和输入对话框）特性的对话框。也就是说，这些对话框的布局特征都被集中在一个对话框上。

为了在程序中实现一个选项对话框，可以调用下面的 `JOptionPane` 类的方法，它直接创建并显示一个选项对话框：

```
public static int showOptionDialog(  
    Component parentComponent,  
    Object message,  
    String title,  
    int optionType,  
    int messageType,  
    Icon icon,  
    Object[] options,  
    Object initialValue)
```

此方法带来一个带有指定标题、消息和图标的模态对话框。`parentComponent` 参数是要在其中显示对话框的框架。如果此参数被指定为空 (`null`) 或者参数 `parentComponent` 没有

框架，就会使用缺省框架。

参数 `messageType` 主要用来提供缺省的图标。初始选择由参数 `initialValue` 决定，而可选项的集合由参数 `optionType` 决定。如果 `optionType` 是 `YES_NO_OPTION` 或 `YES_NO_CANCEL_OPTION` 并且选项参数为空，则由可视组件来提供选项。其余的参数允许用户使用组件而不是按钮。此外，这个参数对显示带有字符串标签的按钮是非常有用的。关于这些参数的更多的信息，请参看前面的对话框中有关“`show`”方法的介绍。

# 第 15 章 文件选择器和颜色选择器

除了第 14 章讨论的选项窗格以外，Swing 库还支持两个“更标准”的对话框：文件选择器和颜色选择器。Swing 文件选择器对 AWT 文件对话框是可选择的。文件选择器类包含创建 Open 和 Save 对话框的方法，而且它支持 Swing 对话框使用自定义的文件视图和文件过滤器。Swing 颜色选择器显示了一个带有控件的对话框，这些控件允许用户选择任何复杂颜色。在 AWT 组件中没有与此等价的内容。

文件选择器和颜色选择器类与前一章讨论的选项窗格类似。当调用选择器对象中的某一方法时，就能创建 Swing 对话框的一个对象，来显示相应的选择器窗格。

## 15.1 文件选择器

文件选择器就是一个用来从文件系统中选择文件的对话框。Swing 文件选择器很容易创建，但它们的功能却很高级。文件选择器可以创建为两种形式：打开文件的对话框和保存文件的对话框。

### 说明：

文件选择器事实上并不能打开或保存文件，它们返回的是要打开或保存的文件对象。用户需要使用已经返回的文件对象来执行打开或保存操作。

Swing 文件选择器由 JComponent 的扩展类 JFileChooser 代表。JFileChooser 的类层次结构显示在图 15-1 中。JFileChooser 类存储在程序包 javax.swing 中。

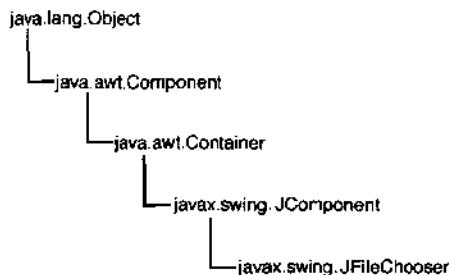


图 15-1 JFileChooser 的类层次结构

### 15.1.1 JFileChooser 构造函数

为了在应用程序中实现一个文件选择器，可以使用被支持的任何一个构造函数来创建 JFileChooser 类型的对象。注意，可以为文件选择器或文件系统的定制视图指定 File 类型的当前目录，或指定 String 类型的当前目录路径。定制视图指明了显示文件时将要附加到视窗上的图标，详细讨论见下节。下面是在 JFileChooser 类中有效的构造函数：

```
public JFileChooser()
public JFileChooser(String currentDirectoryPath)
public JFileChooser(File currentDirectory)
public JFileChooser(
    javax.swing.filechooser.FileSystemView
    fileSystemView)
public JFileChooser(File currentDirectory,
    javax.swing.filechooser.FileSystemView
    fileSystemView)
public JFileChooser(String currentDirectoryPath,
    javax.swing.filechooser.FileSystemView
    fileSystemView)
public JFileChooser(String currentDirectoryPath)
```

创建 JFileChooser，它指向用户的主目录。

```
public JFileChooser(File currentDirectory)
public JFileChooser(
    javax.swing.filechooser.FileSystemView
    fileSystemView)
```

使用参数指明缺省显示的路径，也可以将 File 对象的路径指定为 currentDirectory，String 对象的路径指定为 currentDirectoryPath。如果这些参数值被作为空值传递，那么文件选择器将指向主目录。文件选择器对象也可以使用定制文件视图的信息（作为 fileSystemView）创建。

最后两个构造函数：

```
public JFileChooser(File currentDirectory,
    javax.swing.filechooser.FileSystemView
    fileSystemView)
和
public JFileChooser(String currentDirectoryPath,
    javax.swing.filechooser.FileSystemView
    fileSystemView)
```

将这些参数组合来创建文件选择器对象。

### 15.1.2 JFileChooser 代码示例

清单 15.1 演示了怎样在程序中实现 Swing 文件选择器。此程序是一个带有菜单条的应用程序，它允许用户下拉展开文件菜单。下拉文件菜单包括 Open 和 Save As 菜单项，它们为打开文件或保存文件配置相应的文件选择器。在利用文件选择器选择了一个文件以后，就可以选择某个确认按钮。选择状态将显示在主框架里。图 15-2 和图 15-3 显示了“Open”类型和“Save”类型的文件选择器。

**清单 15.1 带有 Open 和 Save 的 JFileChooser 应用程序 (TJFileChooser.java)**

```
// Demonstrates the Swing file choosers

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

// 1. The class TJFileChooser frame.
public class TJFileChooser extends JFrame {
    JMenuItem openMI, saveMI, saveAsMI, exitMI;
    JFileChooser fileChooser;
    Container container;
    JLabel label;
    File selectedFile;
    public TJFileChooser() {
        // 2. To close the frame.
        this.addWindowListener(new FrameClosing());
        // 3. Get a handle on the frame's content pane.
        container = this.getContentPane();
        // 4. Create a label and add it to the container.
        label = new JLabel();
        container.add(label);
        // 5. Create and add a menu bar.
        JMenuBar menuBar = new JMenuBar();
        menuBar.setBorder(BorderFactory.createEtchedBorder());
        container.add(menuBar, BorderLayout.NORTH);
        // 6. Create and add the file menu and its menu items.
        JMenu fileMenu = new JMenu("File");
        menuBar.add(fileMenu);
        JMenuItem newMI = new JMenuItem("New");
        newMI.setEnabled(false);
        fileMenu.add(newMI);
        openMI = new JMenuItem("Open");
        openMI.addActionListener(new MIActionListener());
        fileMenu.add(openMI);
        fileMenu.addSeparator();
        saveMI = new JMenuItem("Save");
        saveMI.setEnabled(false);
        fileMenu.add(saveMI);
```

```
saveAsMI = new JMenuItem("Save As");
saveAsMI.addActionListener(new MIActionListener());
fileMenu.add(saveAsMI);

fileMenu.addSeparator();
fileMenu.addSeparator();

// 7. "Exit" menu item with an action listener.
exitMI = new JMenuItem("Exit");
exitMI.addActionListener(new MIActionListener());
fileMenu.add(exitMI);

// 8. Create and add the Edit menu.
JMenu editMenu = new JMenu("Edit");
menuBar.add(editMenu);

}

// 9. Menu action listener.
class MIActionListener implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        JMenuItem menuItem = (JMenuItem) ae.getSource();

        // 10. If the Open menu item is selected.
        if (menuItem == openMI) {
            if (file == null) {
                // Create a file chooser pointing to the
                // home directory
                fileChooser = new JFileChooser();
            }
            else {
                // Create a file chooser pointing to the
                // specified file path
                fileChooser = new JFileChooser(file);
            }

            // Show the dialog box (returns the option selected).
            int selected = fileChooser.showOpenDialog(container);

            // 11. If the Open button over the file chooser is pressed.
            if (selected == JFileChooser.APPROVE_OPTION) {
                // Get the selected file
                file = fileChooser.getSelectedFile();

                // Display the file name in the frame.
                label.setText("You have selected to open: " + file.getName());
                label.setHorizontalAlignment(JLabel.CENTER);
                return;
            }
        }
    }
}
```

```
// 12.If the Cancel button is pressed.  
else if (selected == fileChooser.CANCEL_OPTION) {  
    label.setText("You have not selected any file to open!");  
    label.setHorizontalAlignment(JLabel.CENTER);  
    return;  
}  
  
// 13.If the Save menu item is selected.  
else if (menuItem == _ saveAsMI) {  
    // Here comes your file - saving code...  
    fileChooser = new JFileChooser();  
    int selected = fileChooser.showSaveDialog(container);  
    selectedFile = new File("UNTITLED");  
    fileChooser.setSelectedFile(selectedFile);  
  
    if (selected == JFileChooser.APPROVE_OPTION) {  
        // Get the selected file  
        selectedFile = fileChooser.getSelectedFile();  
  
        // Display the selected file name in the frame.  
        label.setText("You have selected to save:" + file.getName());  
        label.setHorizontalAlignment(JLabel.CENTER);  
  
        return;  
    }  
  
    else if (selected == fileChooser.CANCEL_OPTION) {  
        label.setText("You have not selected any file to open!");  
        label.setHorizontalAlignment(JLabel.CENTER);  
  
        return;  
    }  
}  
  
else if (menuItem == exitMI) {  
    System.exit(0);  
}  
}  
}  
  
// 14.Listener to close the frame.  
class FrameClosing extends WindowAdapter {  
    public void windowClosing(WindowEvent e) {  
        System.exit(0);  
    }  
}  
  
// 15.The main method.  
public static void main(String[] args) {
```

```

TJFileChooser frame = new TJFileChooser();
frame.setTitle("TJFileChooser Example");
frame.setSize(450,300);
frame.setVisible(true);
}
}

```

### 代码详析

代码片段 1 从定义 TJFileChooser 类开始。片段 2 加上了一个执行关闭框架的代码的监听程序。片段 3 得到此应用程序框架的内容窗格的句柄。

片段 4 创建了一个标签，并把它加到框架的内容窗格上。片段 5 创建了一个带有蚀刻风格的边界的菜单条并把它加到框架的顶部。片段 6 提供了将不同的菜单项加到文件菜单中的代码。片段 7 显示了退出应用程序的菜单项。片段 8 将另外一个叫做 Edit 菜单加到菜单条上。

片段 9 是一个响应 Open 或 Save As 动作的监听程序。如果被选择的菜单项是 Open 选项，片段 10 就会执行将要发生动作。此代码创建了一个文件选择器对象，并显示用以打开文件的文件对话框。注意，此代码也保持了已经通过文件选择器定位的目录路径。当再一次激活文件选择器时，它就将前面选择的目录作为其当前目录。

片段 11 显示了用户单击 Open 按钮时要执行的代码。此代码调用文件选择器对象 getSelectedFile() 方法，获取要打开的文件对象。用这个文件可以创建一个图形图标并将它作为标签显示。片段 12 显示了按下 Cancel 按钮时要执行的动作。

片段 13 显示了在 Save As 菜单项上操作时的代码，它创建了一个“Save”类型的文件对话框并显示它。当你试着将它带有一个文件名保存时，就会返回相应的文件对象。可以使用这个文件对象来保存文件。片段 14 是执行关闭框架的代码的监听程序类。片段 15 是创建应用程序框架并使它可视的主方法。

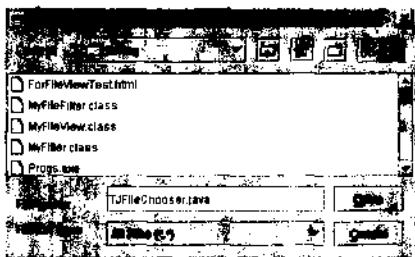


图 15-2 打开文件的文件选择器对话框

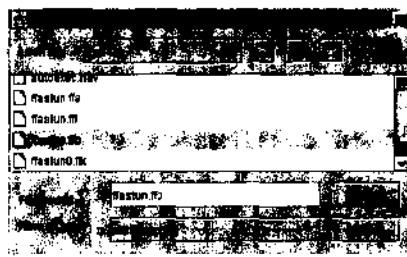


图 15-3 保存文件的文件选择器对话框

## 15.2 定制文件视图

有时候在应用程序中，显示在文件选择器中的文件与用来反映指定文件类型的图标有关。例如，在一个类似 Paint Shop 这样的程序中，图形文件与应用程序指定的图标相关联。Swing 文件选择器支持使用名为 FileView 的抽象类，作为文件定制视图。这个抽象类存在

javax.swing.filechooser 程序包里。

为了给文件选择器中的某个文件应用指定的视图，需要创建抽象类 FileView 的扩展类。此抽象类包含下列方法：

```
public abstract class FileView {  
    public abstract String getName(File f);  
    public abstract String getDescription(File f);  
    public abstract String getTypeDescription(File f);  
    public abstract Icon getIcon(File f);  
    public abstract Boolean isTraversable(File f);  
}
```

一旦用这些方法创建了扩展类（忽略类 FileView 内部的方法），就可以用下面的代码片段为文件选择器应用定制文件视图：

```
// Create an object of the custom file view  
MyFileView fileView = new MyFileView();  
// Assign the file view object to the file chooser object  
JFileChooser fileChooser = new JFileChooser();  
fileChooser.setFileView(fileView);
```

**警告:**

如果在定制文件视图类中要执行的任何方法恰好返回空值，则返回缺省 UI 的文件视图。

由类 FileView 为对话框提供定制文件视图的实现过程演示在程序清单 15.2 中。该程序将放在有关文件过滤器的讨论之后。

### 15.3 文件过滤器

文件过滤器是用来防止不需要的文件出现在文件选择器的文件列表中。当一个目录下有多种类型的文件而你感兴趣的只有一种时，它非常有用。Swing 文件选择器可以和一个由抽象类 FileFilter 扩展的文件过滤器联合。抽象类 FileFilter 只包含两种方法：

```
public abstract class FileFilter {  
    public abstract boolean accept(File f);  
    public abstract String getDescription();  
}
```

为了创建一个文件过滤器，需要创建 FileFilter 抽象类的扩展类。此扩展类由包含执行过滤过程的实际代码实现了覆盖抽象方法的方法。下一步就是创建过滤器对象并把它设计为文件过滤器对象。

**警告:**

不幸的是，在 JDK1.2 中有两个名为 FileFilter 的类。这些类出现在 java.io 和 javax.swing.filechooser 程序包中。当你在程序中访问这些类时，编辑器就会发出警告。

### 15.3.1 JFileFilter 代码示例

清单 15.2 演示了怎样实现一个文件过滤器并为这些文件提供定制视图。此程序基本上执行了抽象类 FileFilter 和 FileView。此执行类的对象被设计为文件选择器对象。图 15-4 显示了这个程序的输出结果。

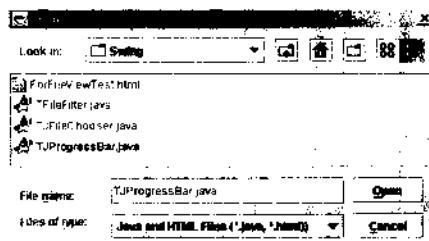


图 15-4 带有定制视图和文件过滤器的文件选择器

清单 15.2 带有文件过滤器和定制文件视图的 JFileFilter (TJFileFilter.java)

```
//Demonstrates the file filters and custom file view.  
import javax.swing.*;  
import javax.swing.border.*;  
import java.awt.*;  
import java.awt.event.*;  
import java.io.*;  
  
public class TJFileFilter extends JFrame {  
    JMenuItem openMI, exitMI;  
    JFileChooser fileChooser;  
    Container container;  
    JLabel label;  
    File file, selectedFile;  
  
    public TJFileFilter() {  
        // 1.To close the frame.  
        this.addWindowListener(new FrameClosing());  
  
        // 2.Get a handle on the frame's content pane.  
        container = this.getContentPane();  
  
        // 3.create a label and add it to the container.  
        label = new JLabel();  
        container.add(label);  
  
        // 4.Create and add a menu bar.  
        JMenuBar menuBar = new JMenuBar();  
        menuBar.setBorder(BorderFactory.createEtchedBorder());  
        container.add(menuBar, BorderLayout.NORTH);  
    }  
}
```

```
// 5.Create and add the file menu and its menu items.  
JMenu fileMenu = new JMenu("File");  
menuBar.add(fileMenu);  
  
openMI = new JMenuItem("Open");  
openMI.addActionListener(new MIActionListener());  
fileMenu.add(openMI);  
  
fileMenu.addSeparator();  
  
// 6."Exit" menu item with an action listener.  
exitMI = new JMenuItem("Exit");  
exitMI.addActionListener(new MIActionListener());  
fileMenu.add(exitMI);  
}  
  
// 7.Menu action listener.  
class MIActionListener implements ActionListener {  
    public void actionPerformed(ActionEvent ae) {  
        JMenuItem menuItem = (JMenuItem) ae.getSource();  
  
        // 8.if the "Open" menu item is selected.  
        if (menuItem == openMI) {  
            if (file == null) {  
                // Create a file chooser pointing to the  
                // home directory.  
                fileChooser = new JFileChooser();  
            }  
            else {  
                // Create a file chooser pointing to the  
                // specified file path.  
                fileChooser = new JFileChooser(file);  
            }  
  
            // 9.Add a file filter (object of MyFileFilter).  
            fileChooser.addChoosableFileFilter(new MyFileFilter());  
  
            // The following statement will assign the new filter  
            // discarding the other filters.  
            //fileChooser.setFileFilter(new MyFileFilter());  
  
            // 10.Set a custom view for the files (using the object of MyFileView).  
            fileChooser.setFileView(new MyFileView());  
  
            // 11.Show the dialog box (returns the option selected).  
            int selected = fileChooser.showOpenDialog(container);  
  
            // 12.if the Open button over the file chooser is pressed.  
            if (selected == JFileChooser.APPROVE_OPTION) {
```

```
// Get the selected file
file = fileChooser.getSelectedFile();

// Display the file name in the frame.
label.setText("You have selected to open: "
file.getName());
label.setHorizontalAlignment(JLabel.CENTER);

return;
}

// 13. if the Cancel button is pressed.
else if (selected == fileChooser.CANCEL_OPTION) {
    label.setText("You have not selected any file to open!");
    label.setHorizontalAlignment(JLabel.CENTER);
    return;
}

// 14. If the Exit button is pressed.
else if (menuItem == exitMI) {
    System.exit(0);
}

}

}

// 15. Listener to close the frame.
class FrameClosing extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
}

// 16. The main method.
public static void main(String[] args) {
    TFileFilter frame = new TFileFilter();
    frame.setTitle("TFileFilter Example");
    frame.setSize(450, 300);
    frame.setVisible(true);
}

}

// 17. Define a file filter class.
class MyFileFilter extends javax.swing.filechooser.FileFilter {

    public boolean accept(File file) {
        if (file.isDirectory()) { // Accept if the file system member is a directory
            return true;
        }
    }
}
```

```
// Retrieve the file name.  
String fileName = file.getName();  
// Note the index of '.' in a file name  
int periodIndex = fileName.lastIndexOf('.');  
// State of acceptance.  
boolean accepted = false;  
  
if (periodIndex > 0 && periodIndex < fileName.length() - 1) {  
    String extension = fileName.substring(periodIndex + 1).toLowerCase();  
    if (extension.equals("java")) // Check if the extension is ".java"  
        accepted = true;  
    else if(extension.equals("html")) // Check if the extension is .html.  
        accepted = true;  
}  
return accepted;  
}  
  
// Retrieve the description of the filter.  
public String getDescription() {  
    return "Java and HTML Files (*.java, *.html)";  
}  
}  
  
// 18. Custom file view class.  
class MyFileView extends javax.swing.filechooser.FileView {  
    // Define the required icon objects.  
    ImageIcon javaIcon = new ImageIcon("javaIcon.gif");  
    ImageIcon htmlIcon = new ImageIcon("htmlIcon.gif");  
  
    // Retrieve the file name.  
    public String getName(File f) {  
        return null; // L&F FileView will find this out  
                    // or use return f.getName();  
    }  
  
    // Retrieve the description of the file.  
    public String getDescription(File f) {  
        return null; // L&F FileView will find this out  
    }  
  
    // Returns the file type as Java.  
    public String getTypeDescription(File f) {  
        String extension = getExtension(f);  
        String type = null;  
        if(extension != null) {  
            if(extension.equals("java")) {  
                type = "Java File";  
            }  
        }  
        return type;  
    }  
}
```

```
        }

        if(extension.equals("gif")){
            type = "Image File";
        }
    }

    return type;
}

// Retrieve the relevant icon for the file view.
public Icon getIcon(File f) {
    String extension = getExtension(f);
    Icon icon = null;
    if (extension != null) {
        if(extension.equals("java")) {
            icon = javaIcon;
        }
        if(extension.equals("html")) {
            icon = htmlIcon;
        }
    }
    return icon;
}

// Whether a direction is traversable.
public Boolean isTraversable(File f) {
    return null; // L&F FileView will find this out
}

// Retrieve the extension of a file.
private String getExtension(File f) {
    String fileName = f.getName();
    int periodIndex = fileName.lastIndexOf('.');
    String extension = null;
    if(periodIndex > 0 && periodIndex < fileName.length() - 1) {
        extension = fileName.substring(periodIndex + 1).toLowerCase();
    }
    return extension;
}
}
```

## 代码详析

在这个程序中，片段 1 – 9 和片段 12 – 16 创建了类似于清单 15.1 中显示的应用程序。不过，文件 9 和片段 10 显示的代码将文件过滤器和文件视图对象设为定制的。方法 addChoosableFileFilter () 将 MyFileFilter 类型的文件过滤器对象设计为文件选择器。这个类只

能显示文件过滤器中扩展名为 .java 和 .html 的文件。片段 10 中的 setFileView () 方法为显示在文件选择器中的文件指定定制视图。

片段 17 定义了 javax.swing.filechooser.FileFilter 的扩展类 MyFileFilter。FileFilter 类实现了两个抽象方法: accept () 和 getDescription ()。accept () 方法依赖于是否在文件选择器中返回指定文件的布尔状态。而 getDescription () 将文件描述为“Java 和 HTML 文件”的任何文本。

片段 18 定义了叫做 MyFileFilter 的定制文件视图类,它是 javax.swing.filechooser.FileView 的扩展类。此定制文件视图类执行了 FileView 类的抽象方法。方法 getName ()、getDescription () 和 getTypeDescription () 分别返回文件名、文件的说明文本和文件类型的描述。isTraversable () 方法指定一个目录是可穿越的。getIcon () 方法是一个至关重要的方法,它返回与定制文件视图有关的图标。

## 15.4 颜色选择器

Swing 颜色选择器是一个显示操作及选择需要颜色的控制窗格的组件。颜色选择器包含一个带调色板和选择颜色的 RGB 滑块的多标签窗格。从 Swatches 窗格可以直接选择任何期望的颜色。假如所期望的颜色在 Swatches 窗格无效,可以操作 RGB 窗格上的滑块来得到这个颜色。

Swing 颜色选择器窗格由类 JColorChooser 表示,它是 JComponent 的扩展类。JColorChooser 的类层次结构如图 15-5 所示。JColorChooser 位于 javax.swing 程序包中。

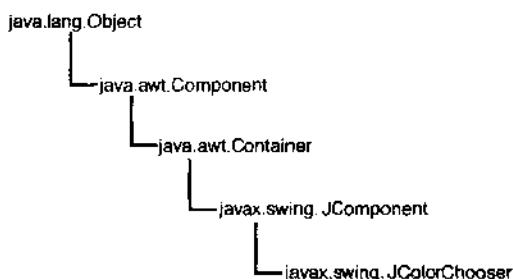


图 15-5 JColorChooser 的类层次结构

### 15.4.1 JColorChooser 的构造函数

要创建一个颜色选择器对象,可以使用 JColorChooser 支持的下列构造函数:

```

public JColorChooser()
public JColorChooser(Color initialColor)
public JColorChooser(javax.swing.colorchooser.ColorSelectionModel model)
  
```

在这些构造函数里, initialColor 参数是当颜色选择器弹出时所具有的初始颜色。第一个不带任何参数的构造函数创建一个以白色为初始颜色的颜色选择器窗格。下一个构造函数创建带有指定初始颜色的对象。第三个构造函数创建一个具有预先描述的颜色选择模型的颜色

选择器对象。名为 model 的参数是 ColorSelectionModel 类型的颜色选择模型。一旦创建了一个颜色选择器窗格，就可以通过使用 JColorChooser 支持的静态方法在 Swing 对话框中显示它。

### 15.4.2 JColorChooser 代码示例

清单 15.3 是一个简单的画图小程序（见图 15-6），通过在画布上拖动鼠标指针作图。当按下位于小程序上的 Show Color Chooser 按钮时，就会弹出一个颜色选择器窗格。图 15-7 显示了此颜色选择器窗格的 Swatches 标签，图 15-8 显示了 RGB 标签，它有利于选择更复杂的颜色。

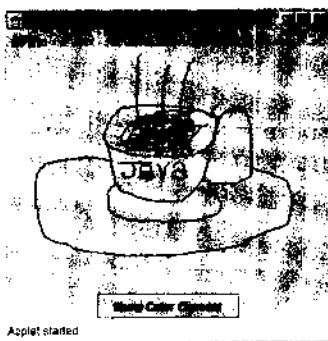


图 15-6 一个简单的画图小程序

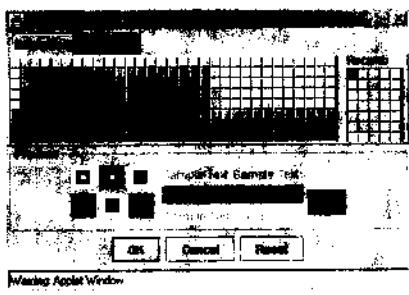


图 15-7 显示调色盘的颜色选择器的 Swatches 窗格

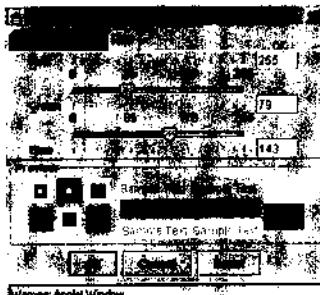


图 15-8 用以微调颜色的 RGB 窗格

清单 15.3 简单作图小程序中的 JcolorChooser (*TJColorChooser.java*)

```
/*
 * <Applet code=TJColorChooser width=400 height=400>
 * </Applet>
 */
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class TJColorChooser extends JApplet {
    Container container = null;
    DrawingPanel panel = null;
    JColorChooser colorchooser = null;
    JDialog dialog = null;
    int oldX, oldY, newX, newY;
    Color color = null;

    public void init() {
        // 1. Get a handle on the applet's content pane.
        container = this.getContentPane();

        // 2. Create a drawing panel and add it at the center
        // of the content pane of the applet.
        panel = new DrawingPanel();
        container.add(panel);

        // 3. Add a button at the bottom portion
        // of the applet to display the color chooser pane.
        JButton showButton = new JButton("Show Color Chooser");
        showButton.addActionListener(new ButtonListener());
        Box hBox = Box.createHorizontalBox();
        hBox.add(Box.createHorizontalGlue());
        hBox.add(showButton);
        hBox.add(Box.createHorizontalGlue());
        container.add(hBox, BorderLayout.SOUTH);

        // 4. Create a color chooser object and a dialog box
        // that displays the color chooser.
        colorChooser = new JColorChooser();
        dialog = JColorChooser.createDialog(
            container,
            "Color Chooser",
            false,
            colorChooser,
            new ButtonListener(),
            new ButtonListener());
    }

    // 5. This is where you perform your drawing with
    // the selected color.

    class DrawingPanel extends Panel {
        public DrawingPanel () {

```

```
setBackground(Color.white);
MyMouseListener mouseListener = new MyMouseListener();
addMouseListener(mouseListener);
addMouseMotionListener(mouseListener);
}

public void update(Graphics g) {
    g.setColor(color);
    paint(g);
}

public void paint(Graphics g) {
    g.drawLine(oldX, oldY, newX, newY);
}
}

// 6. Listener class that responds to all button actions.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();

        if (button.getText().equals("Show Color Chooser")) {
            dialog.show();
        }

        if (button.getText().equals("OK")) {
            //dialog.show();
            color = colorChooser.getColor();
        }

        else if(button.getText().equals("Cancel")) {
            dialog.dispose();
        }

        // Note: Don't have to handle the actions of the
        // 'Reset' button; this button has been implemented
        // to reset the color to the previously used color.
    }
}

// 7. Mouse listener class to draw on the canvas.
class MyMouseListener extends MouseAdapter
    implements MouseMotionListener {
    public void mousePressed(MouseEvent e) {
        oldX = e.getX(); oldY = e.getY();
        newX = e.getX(); newY = e.getY();
        panel.repaint();
    }
}
```

```
public void mouseDragged(MouseEvent e) {  
    oldX = newX; oldY = newY;  
    newX = e.getX(); newY = e.getY();  
  
    panel.repaint();  
}  
  
public void mouseMoved(MouseEvent e) {}  
}  
}
```

## 代码详析

小程序 TJColorChooser 由声明要显示的数据成员开始。在 init() 方法中，片段 1 得到此小程序内容窗格的句柄。片段 2 创建了一个画图窗格并把它放到小程序的中央。片段 3 创建了一个用来弹出 Color Chooser 对话框的按钮。

片段 4 创建了 JColorChooser 类型的对象。然后调用 JColorChooser 类的静态方法 createDialog() 创建一个对话框。createDialog() 方法将颜色选择器对象作为它的参数值之一，用来在对话框里显示颜色选择器。片段 5 定义了 DrawingPanel 类。DrawingPanel 已经被注册为鼠标移动监听程序。这个类使用 update() 和 paint() 方法来绘制从鼠标的当前坐标点到前一次坐标的直线。

片段 6 是按钮监听程序，通常指的是颜色选择器窗格上的 Show Color Chooser 按钮、OK 按钮和 Cancel 按钮。当单击 OK 按钮时，就会通过调用颜色选择器对象的 getColor() 方法重新找到选定的颜色。

### 说明：

为了得到在 JColorChooser 的 Swatches 窗格中无效的颜色，可以先 Swatches 窗格中选择一种近似的颜色，然后切换到 RGB 窗格并调节滑块来获得期望的颜色。如果需要，也可以使用 HSB 窗格。

片段 7 显示了鼠标监听程序类，它在窗格内获取鼠标坐标并调用 paint() 方法。为了执行此功能，必须单击或拖动鼠标。

# 第 16 章 树

树就是一个用来显示数据集的层次式用户接口。树的最常见用途就是，显示目录结构和与网络驱动器相关的文件。

JTree 类是用来创建极其高级的轻量级的树组件。树的设计使用了 Model – View – Controller (MVC) 建筑结构方法。通过对树组件的节点作出选择来执行一定的功能；树组件激活被接受执行所需功能的事件。

Swing 树为它的每一个节点显示一个定制外观的图标。不过，也可以为树的节点单元定制图形以达到定制外观的目的。Swing 树也支持可编辑的节点单元。

## 16.1 创建树

树中最基本的对象叫做节点 (node)，它表示在给定层次结构中的数据项。这样，树就是一个或多个节点的组合。其根节点，或简称根，就是这个数据层次结构的顶端节点。

在根节点内部的节点叫做子节点，或简称节点。没有子节点的节点叫做叶节点 (leaf node)。一个非叶节点的节点可以有任意多个子节点，包括没有节点。在每一列里，都能发现这样的节点，其父节点在前一列。

例如，在 Windows NT 文件系统里，其根节点是 c:\ , 在 c:\ 目录下的文件和所有的子目录都表示子节点。这些子目录下的文件不能再深入，所以它们就是文件系统树的叶。

### 16.1.1 树和节点类

Swing 树的所有功能分配给了几个类。不过，只有一个类 JTree 是所有类的中心。类 JTree 表示树组件并存储在 javax.swing 程序包里。JTree 的类层次结构如图 16-1 所示。

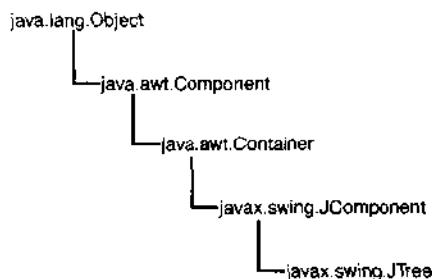


图 16-1 JTree 的类层次式结构

Swing 树由节点构成，这些节点是 TreeNode 类型的对象。TreeNode 是存储在程序包 javax.swing.tree 中的接口。这个接口包含能返回有关节点信息的方法，比如它的子节点数；如果此节点是叶节点，就返回它的父节点。还有另一个接口叫做 MutableTreeNode，是 TreeN-

ode 的扩展类。它只是简单的加上更多的抽象方法。

用 DefaultMutableTreeNode 类可以提供 MutableTreeNode 接口的缺省实现,而无需直接实现这些接口。DefaultMutableTreeNode 是 java.lang.Object 的扩展类。如果要由节点来构造树,可以创建 DefaultMutableTreeNode 类型的节点并通过它的构造函数为这个 JTree 对象指定节点的根。有关节点接口的方法列表和类 DefaultMutableTreeNode,请参考附录 A:“JFC Swing 快速参考”。

下面可以用来创建树节点的 DefaultMutableTreeNode 的构造函数:

```
public DefaultMutableTreeNode()
```

创建了一个没有父节点和任何初始子节点的树节点,但是这个节点允许加上子节点。

```
public DefaultMutableTreeNode(Object userObject, boolean allowsChildren)
```

带了一个用户对象和一个逻辑标志。显示用户对象 userObject 以描述节点,通常,这是一个字符串对象如 c:\win95。逻辑标志 allowsChildren 指明此节点是否允许有子节点(真),或者它是否是一个叶节点(假)。

```
public DefaultMutableTreeNode(Object userObject)
```

类似于前面的构造函数,但总是允许给它加上子节点。

#### 警告:

DefaultMutableTreeNode 类在多线程环境中使用时并不保险。如果要在多线程中使用这个类或 TreeNodes 树,就需要同步执行(使用树的根节点更适合)。

### 16.1.2 树路径

在程序包 javax.swing.tree 里有一个叫做 TreePath 的类,是通向指定节点的树的路径。通常,用这个类可以获得有关根节点和指定节点之间的中介节点的信息。关于它的方法请见附录 A。

### 16.1.3 JTree 构造函数

使用 JTree 的构造函数创建 Swing 树组件,有几种不同的方法。下面是类 JTree 中有效的七个构造函数:

```
public JTree()
```

创建了有缺省模型的 Swing 树。在后面的步骤里,可以使用方法 setModel() 给这棵树设计不同的模型。

```
public JTree(Hashtable value)
```

由哈希表创建了一棵不显示根的树。

```
public JTree(Object[] value)
```

也创建了一棵不显示根节点的树,它将指定数组的每个元素作为根节点的子节点。

```
public JTree(TreeModel newModel)
```

使用指定数据模型创建了一棵 Swing 树。

```
public JTree(TreeNode root,
boolean asksAllowsChildren)
```

创建了以指定的 TreeNode 作为其根的 JTree 对象。还可以指出是否允许树节点有子节点。

```
public JTree(TreeNode root)
```

还创建了以指定的 TreeNode 作为其根的树对象，但这棵树也可以显示根节点。

```
public JTree(Vector value)
```

创建了一个不显示根节点的 JTree 对象，它将指向量的每个元素作为根节点的子节点。

#### 16.1.4 JTree 代码示例

清单 16.1 显示了由节点创建树对象的例子代码。一开始，使用 DefaultMutableTreeNode 类的构造函数创建树节点。然后，以层次结构方式给父节点加上相应的子节点。节点系统的根被连接到树对象上。图 16-2 显示了该程序的输出结果。

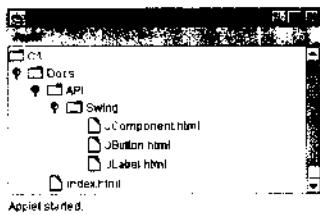


图 16-2 显示 Swing 树的小程序

##### 清单 16.1 显示节点和叶的 JTree(TJTree.java)

```
// Demonstrates how to create the Swing tree widget to
// represent hierarchical data.

/*
 * <Applet code = TJTree width = 350 height = 300>
 * </Applet>
 */

import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
import java.awt.event.*;

public class TJTree extends JApplet {
```

```
Container container;

public void init() {
    // 1.Get the handle on applet's content pane.
    container = this.getContentPane();

    // 2.Root node (col0 or root) of the tree.
    DefaultMutableTreeNode root =
        new DefaultMutableTreeNode("C:\\");

    // 3.Create the nodes in column 1.
    DefaultMutableTreeNode col11 =
        new DefaultMutableTreeNode("Docs");

    DefaultMutableTreeNode col12 =
        new DefaultMutableTreeNode("README");

    // 4.Add the nodes to the root.
    root.add(col11);
    root.add(col12);

    // 5.Create the nodes in column 2.
    DefaultMutableTreeNode col21 =
        new DefaultMutableTreeNode("API");
    DefaultMutableTreeNode col22 =
        new DefaultMutableTreeNode("index.html");

    // 6.Add the nodes to the node 1 in column 1.
    col11.add(col21);
    col11.add(col22);

    // 7.Create the nodes in column 3.
    DefaultMutableTreeNode col31 =
        new DefaultMutableTreeNode("Swing");

    // 8.Add the node to the node 2 in column 2.
    col21.add(col31);

    // 9.Create the nodes in column 4.
    DefaultMutableTreeNode col41 =
        new DefaultMutableTreeNode("JComponent.html");
    DefaultMutableTreeNode col42 =
        new DefaultMutableTreeNode("JButton.html");
    DefaultMutableTreeNode col43 =
        new DefaultMutableTreeNode("JLabel.html");

    // 10.Add the nodes to the node called Swing.
    col31.add(col41);
    col31.add(col42);
    col31.add(col43);
```

```

// 11. Attach the root (of nodes) to the tree object.
JTree tree = new JTree(root);

// 12. Add the tree to a scroll pane and add the scroll pane
// to the container.
JScrollPane scrollPane = new JScrollPane(tree);
container.add(scrollPane);

}

}

```

### 代码详析

清单 16.1 是一个叫做 TJTree 的 Swing 小程序。在 init()方法里，片段 1 得到小程序内容窗格的句柄。片段 2 创建树的根节点(c:\ \ 像在 Windows 系统中一样)。可以假设根节点存在于树的第 0 层(树节点可以看作有许多层)。片段 3 创建了两个子节点(在第 1 层)并将其加到片段 4 中给出的根上。

片段 5 创建了多于两个的子节点(在第 2 层上)。这些节点被加到第 1 层的节点 1 上，即节点“Docs”(片段 6)。片段 7 在第 3 层上创建了一个节点并把它加到(片段 8)第 2 层的节点 1 上，它是 API 节点。片段 9 显示了在第 4 层上创建节点并加到第 3 层(片段 10)的节点 1 上的代码。

最后，通过使用相应的构造函数将节点的根加到树对象上，请见片段 11。片段 12 显示了将树加到滚动窗格上的代码，此滚动窗格已经被加到小程序中。

## 16.2 树模型

树可以用来显示比图 16-2 中更多更复杂的数据。Swing 库支持叫做 TreeModel 的接口，它存储在 javax.swing.tree 程序包中。此接口通过执行下列方法建立树组件下层的数据：

```
public Object getChild()
```

返回树的根。

```
public Object getChild(Object parent,
                      int index)
```

返回在子节点数组中有效参数索引的子节点。参数 parent 是已经从数据源中获得的节点。

```
public int getChildCount(Object parent)
```

返回参数 parent 的子节点数。如果节点是叶它就返回零，或没有子节点。

```
public int getChildCount(Object node)
```

如果指定的节点是叶节点，则返回 True。

```
public int getIndexOfChild(Object parent,
                           Object child)
```

返回父节点指定的子节点的索引。

```
public void valueForPathChanged(TreePath path,  
                                Object newValue)
```

如果用户已经改变了由指定路径识别的数据,通告程序中有趣的部分。

```
public void addTreeModelListener(TreeModelListener l)
```

加上一个监听程序对象,当树的数据改变时它将执行特定的功能。

```
public void removeTreeModelListener(TreeModelListener l)
```

简单地删除监听程序对象。

在缺省情况下,树组件使用由类 DefaultTreeModel 表示的数据模型。DefaultTreeModel 对象包含基于树节点的模型数据。关于这个类的方法列表,请参看附录 A。

### 16.2.1 模型事件和监听程序

一旦树模型的数据发生了任何改变,树对象就激活 TreeModelEvent 类型的事件。模型事件包含有关树模型变化的信息。这些事件由 TreeModelListener 接口类型的监听程序对象接收。

这个监听程序接口包括下列必须执行的方法:

```
public void treeNodesChanged(TreeModelEvent e)
```

如果节点以某种方式改变,则调用上条语句。

```
public void treeNodesInserted(TreeModelEvent e)
```

如果节点已经插入到树中,则调用上条语句。

```
public void treeNodesRemoved(TreeModelEvent e)
```

如果节点已经从树中删除,则调用上条语句。

```
public void treeStructureChanged(TreeModelEvent e)
```

如果这棵树因为一个节点而彻底改变了结构,则调用上条语句。

为了在程序中使用接口 TreeModel,必须实现表示数据模型的类中的这些方法。模型类对象可以通过构造函数或通过调用树对象中的 setModel() 方法应用于树组件。

### 16.2.2 定制树模型代码示例

在清单 16.2 中,计算机中的文件系统显示在使用 Swing 树设计的框架内。例子中的树使用了执行实现 TreeModel 的类对象。注意,每个接口方法如何从文件系统中获得目录和文件。执行类对象通过它的构造函数被设计为树对象。这个程序的输出显示在图 16-3 中。

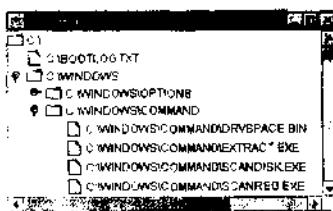


图 16-3 通过实现 TreeModel 显示文件系统

## 清单 16.2 利用定制树模型显示文件系统层次结构的 JTree(TTreeModel.java)

```
// Demonstrates how to use custom data models in trees.

import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.util.*;

public class TTreeModel extends JFrame {
    public TTreeModel() {
        super("TTreeModel"); // Give a title to the frame
        // 1.Create an object of FileSystemModel, and create a tree
        // with that model. Add the tree to a scroll pane, and add
        // the scroll pane to the frame.
        FileSystemModel fileSystemDataModel = new FileSystemModel();
        JTree tree = new JTree(fileSystemDataModel);
        JScrollPane scrollPane = new JScrollPane(tree);
        getContentPane().add(scrollPane);

        // 2.Configure the frame and display it.
        addWindowListener(new WindowEventHandler());
        setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        setSize(300,250);
        show();
    }

    // 3.The main method.
    public static void main(String [] args) {
        TTreeModel frame = new TTreeModel();
    }

    // 4.Define a listener class to close the frame.
    class WindowEventHandler extends WindowAdapter {
        public void windowClosing(WindowEvent evt) {
```

```
        System.exit(0);
    }
}
}

// 5. Custom data model that represents the file system data.
class FileSystemModel implements TreeModel {
    private String root; // The root identifier
    private Vector listeners; // Declare the listeners vector

    public FileSystemModel() {
        // 6. Get the home directory on your system.
        root = System.getProperty("user.home");

        // 7. For Windows 95, root is set to C:\ ("usr.home"
        // retrieves C:\WIN95).
        // You may implement a snippet like this for other OS also
        // like Windows NT, Macintosh, and so on.
        if (System.getProperty("os.name").equals("Windows 95")) {
            File tempFile = new File(root);
            root = tempFile.getParent();
        }

        // 8. Define the listeners vector.
        listeners = new Vector();
    }

    // 9. Retrieves the root of the tree hierarchy.
    public Object getRoot() {
        return (new File(root));
    }

    // 10. Retrieves the members in a directory based on an index.
    public Object getChild(Object parent, int index) {
        File directory = (File) parent;
        String[] directoryMembers = directory.list();
        return (new File(directory,directoryMembers[index]));
    }

    // 11. Retrieves the member count in a directory.
    public int getChildCount(Object parent) {
        File fileSystemMember = (File) parent;
        // fileSystemMember is a directory or file.

        // If a file system member is a directory.
        if (fileSystemMember.isDirectory()) {
            String[] directoryMembers = fileSystemMember.list();
            // Get the count of members in the directory.
        }
    }
}
```

```
    return directoryMembers.length;
}

// If the file system member is a file.
else {
    // Return the member count as zero.
    return 0;
}
}

// 12.Returns the index of a given member in its directory.
public int getChildIndex(Object parent, Object child) {
    File directory = (File) parent;
    File directoryMember = (File) child;
    String[] directoryMemberNames = directory.list();
    int result = -1;

    for (int i = 0; i < directoryMemberNames.length; ++ i) {
        if (directoryMember.getName().equals(
            directoryMemberNames[i])) {
            result = i;
            break;
        }
    }
    // If no member with such name in the directory.
    return result;
}

// 13.Indicates whether a directory member is a tree leaf.
public boolean isLeaf(Object node) {
    return ((File) node).isFile();
}

// 14.Method to add a tree model listener.
public void addTreeModelListener(TreeModelListener l) {
    if (l != null && !listeners.contains(l)) {
        listeners.addElement(l);
    }
}

// 15.Method to remove a tree model listener.
public void removeTreeModelListener(TreeModelListener l) {
    if (l != null) {
        listeners.removeElement(l);
    }
}

// A dumb method.
```

```
public void valueForPathChanged(TreePath path, Object newValue) {
    // Does Nothing!
}

// 16. Additional methods that fire events whenever the model is
// subjected to change. Possible changes can be nodes
// inserted, nodes removed, structure changed, and so on.
public void fireTreeNodesInserted(TreeModelEvent e) {
    Enumeration listenerCount = listeners.elements();
    while (listenerCount.hasMoreElements()) {
        TreeModelListener listener =
            (TreeModelListener) listenerCount.nextElement();
        listener.treeNodesInserted(e);
    }
}

public void fireTreeNodesRemoved(TreeModelEvent e) {
    Enumeration listenerCount = listeners.elements();
    while (listenerCount.hasMoreElements()) {
        TreeModelListener listener =
            (TreeModelListener) listenerCount.nextElement();
        listener.treeNodesRemoved(e);
    }
}

public void fireTreeNodesChanged(TreeModelEvent e) {
    Enumeration listenerCount = listeners.elements();
    while (listenerCount.hasMoreElements()) {
        TreeModelListener listener =
            (TreeModelListener) listenerCount.nextElement();
        listener.treeNodesChanged(e);
    }
}

public void fireTreeStructureChanged(TreeModelEvent e) {
    Enumeration listenerCount = listeners.elements();
    while (listenerCount.hasMoreElements()) {
        TreeModelListener listener =
            (TreeModelListener) listenerCount.nextElement();
        listener.treeStructureChanged(e);
    }
}
```

### 代码详析

在框架 TTreeModel 的构造函数里, 片段 1 创建了 FileSystemModel 类型的对象。这个对象通过它的构造函数赋予树对象。然后, 这棵树使用滚动窗格显示在框架里。片段 2 显示了配置和显示应用程序框架的代码。

片段 3 显示了创建框架对象的主方法。片段 4 是关闭窗口处理程序操作的代码。这个类的对象已被注册为显示在片段 2 中的框架。

片段 5 显示了树模型类, 它定义了运行应用程序的计算机文件系统数据的模型。在这个类的构造函数里, 片段 6 得到了系统的主目录, 这个主目录将作为树组件的根。片段 7 显示了取得根的代码, 这个根在 c:\ 而不是 Windows95 操作系统中的 c:\WIN95。处理其他操作系统也需要类似的代码。片段 8 定义了一个包含接收模型事件的模型监听程序的向量对象。当树模型中的数据改变时, 就会产生模型事件。

片段 9~15 显示了必须实现的接口 TreeModel 接口方法。片段 9 中的方法检索创建树的根的信息。片段 10 返回在给定索引中指定父节点的子节点。片段 11 得到指定目录下的节点数(文件和目录数)。片段 12 检索指定节点的索引。片段 13 确认给定节点是否为叶节点。片段 14 和片段 15 显示了怎样对树对象的注册监听程序加上或去除方法。片段 16 显示了激活不同事件所支持的其他方法。

## 16.3 用树完成交互

树的用户接口通常以两种方式执行: 扩展树节点来存取它的内部节点和叶节点; 或者为某些进一步的进程(如显示由叶节点代表的文件内容)选择叶节点。

为了处理树节点的选择, 有一个叫做 TreeSelectionModel 的单独接口。这个接口允许当前树选择的数据表示。而类 DefaultTreeSelectionModel 表示树选择的数据模型。下面是选择树节点的三种不同方式:

#### SINGLE\_TREE\_SELECTION

在这种方式里, 每次只能选择一个节点。

#### CONTIGUOUS\_TREE\_SELECTION

在这种方式里, 每次可以选择多个节点, 但是这些节点的路径必须互相连接。

#### DISCONTIGUOUS\_TREE\_SELECTION

用这种方式也可以没有约束地一次选择多个节点。使用方法 setSelectionMode(), 还可以给树赋予一种新的模型。

#### 说明:

在缺省状态下, 树选择模型的值是 DISCONTIGUOUS\_TREE\_SELECTION。

### 16.3.1 选择事件和监听程序

Swing 树激活 TreeSelectionEvent 类型的事件来响应选择节点的用户接口。TreeSelection-

Event 类刻画了树选择中变化的特征。这种选择可能沿任何树的路径发生。

选择事件需要一个选择监听程序对象来接收,该对象实现叫做 TreeSelectionListener 的监听程序接口。树选择监听程序能够判断出已经被提交以选择改变的节点状态。事件及其监听程序类都存储在 javax.swing.event 程序包中。

### 16.3.2 扩展事件和监听程序

当扩展树节点时,树对象就会激活 TreeExpansionEvent 类型的事件。类 TreeExpansionEvent 定义了节点沿着树的路径扩张或收缩的特性。调用事件对象的路径 getPath(),可接收扩张或收缩节点的路径。这个事件由监听程序类对象接收。监听程序类实现接口 TreeExpansionListener。这个事件及其监听程序类都存储在 javax.swing.event 程序包中。

### 16.3.2 JTree 选择事件代码示例

清单 16.3 是一个演示怎样通过使用对应的监听程序处理树选择的应用程序。这个应用程序框架已连接到拆分窗格上。此拆分窗格有两个滚动窗格:一个用来显示树组件,另一个包含显示所选 HTML 文件的编辑器窗格。在树的叶节点上单击鼠标作出选择以后,相应的文件就显示在编辑器窗格上,如图 16-4 所示。

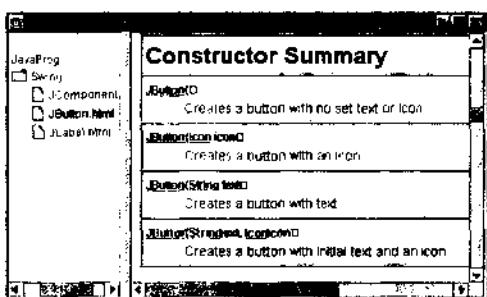


图 16-4 JTree 节点扩展和选择

清单 16.3 在编辑器窗格中显示文档的 JTree 节点选择 (TTreeSelectionEvent.java)

```
// Demonstrates the tree selection events and listeners.

import javax.swing.*; import javax.swing.tree.*; import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;

public class TTreeSelectionEvent extends JFrame {
    Container container;
    DefaultMutableTreeNode root;
    JScrollPane treeScrollPane;
    JScrollPane editorScrollPane;
    JEditorPane editorPane;
```

```
public TTreeSelectionEvent() {  
    // 1. Get the handle on applet's content pane and  
    // create the tree nodes  
    super("TTreeSelectionEvent");  
    container = this.getContentPane();  
    createTreeNodes();  
  
    // 2. Attach the root of the nodes to the tree object  
    // and set the tree to the SINGLE SELECTION mode.  
    // Also, register a tree selection listener with the  
    // tree object to listen to the respective events fired.  
    JTree tree = new JTree(root);  
    tree.getSelectionModel().setSelectionMode(  
        TreeSelectionModel.SINGLE_TREE_SELECTION);  
    tree.addTreeSelectionListener(new SelectionListener());  
  
    // 3. Create scroll panes to display the tree, and the  
    // editor component. Add the scroll panes to a split pane.  
    // Add the split pane to the applet.  
    treeScrollPane = new JScrollPane(tree);  
  
    editorPane = new JEditorPane();  
    editorScrollPane = new JScrollPane(editorPane);  
    JSplitPane splitPane = new JSplitPane(  
        JSplitPane.HORIZONTAL_SPLIT,  
        true, // Continuous Layout  
        treeScrollPane,  
        editorScrollPane);  
  
    // 4. Add the split pane to the frame.  
    container.add(splitPane);  
  
    // 5. Add the window closing listener.  
    addWindowListener(new WindowEventHandler());  
  
    // 6. Configure the frame.  
    setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);  
    setSize(350, 250);  
    show();  
}  
  
// 7. Method that creates the tree nodes.  
public void createTreeNodes() {  
    // 8. Root node (col0 (col - zero) or root) of the tree.  
    root = new DefaultMutableTreeNode("C: \ ");  
  
    // 9. Create the nodes in column 1.  
    // and add them to the root  
    DefaultMutableTreeNode col11 =
```

```
new DefaultMutableTreeNode("JavaProg");
root.add(col11);

// 10.Create the nodes in column 2.
DefaultMutableTreeNode col21 =
    new DefaultMutableTreeNode("Swing");

// 11.Add the node to the node 2 in column 2.
col11.add(col21);

// 12.Create the nodes in column 3.
DefaultMutableTreeNode col31 =
    new DefaultMutableTreeNode("JComponent.html");
DefaultMutableTreeNode col32 =
    new DefaultMutableTreeNode("JButton.html");
DefaultMutableTreeNode col33 =
    new DefaultMutableTreeNode("JLabel.html");

// 13.Add the nodes to the node called Swing.
col21.add(col31);
col21.add(col32);
col21.add(col33);
}

// 14.Define the tree selection listener.
class SelectionListener implements TreeSelectionListener {
    URL url;

    // 15.Method that needs to be implemented.
    public void valueChanged(TreeSelectionEvent se) {
        // Obtain the source.
        JTree tree = (JTree) se.getSource();
        // Obtain the selected node.
        DefaultMutableTreeNode selectedNode =
            (DefaultMutableTreeNode)
                tree.getLastSelectedPathComponent();
        // Obtain the selected node name.
        String selectedNodeName = selectedNode.toString();
        // If the node is a leaf,obtain its URL string.
        if (selectedNode.isLeaf()) {
            String urlString = "file:" +
                System.getProperty("user.dir") +
                System.getProperty("file.separator") +
                selectedNodeName;
            System.out.println(urlString);
        }
    }
}
```

代码解析

在清单 16.3 中，定义了叫做 `TTreeSelectionEvent` 的 Swing 框架。在此框架的构造函数里，片段 1 得到框架内容窗格的句柄并且执行方法 `createTreeNodes()`。这个代码的详情在片段 7 的方法中给出。

片段 2 通过节点系统的根创建了树对象。这棵树被构造为 SINGLE\_TREE\_SELECTION 模型。方法 `getSelectionModel()` 返回当前的选择模型，它的结构可以改变。方法 `setSelectionMode()` 赋予了新的选择模型。接下来，树节点选择监听程序通过使用方法 `addTreeSelectionListener()` 注册为树对象。

片段 3 创建了一个滚动窗格并加入树对象。另外，还创建了一个编辑器窗格并加到另一个滚动窗格上。接下来，这两个滚动窗格都被加到拆分窗格上。片段 4 显示了把拆分窗格连接到内容窗格的语句。片段 5 和片段 6 显示了加上框架关闭监听程序和构造框架的代码。

片段 7-13 定义了方法 `createTreeNodes()`，片段 1 也调用了这个方法。此方法创建了一系列节点对象，然后排列它们形成树的节点系统。

片段 14 定义了树的节点选择监听程序，它通过为 `valueChanged()` 接口方法提供代码

来实现接口 TreeSelectionListener。方法 valueChanged() 中的代码显示了编辑器窗格组件(在片段 3 中创建)中选定的文件。片段 16 显示了处理框架关闭的类。片段 17 是启动该程序的主方法。

## 16.4 绘制树节点单元

每一个树节点都与一个单元相关联，该单元显示描述该节点的文本标签和图像图标。图标指明该节点是否是叶节点。假如它不是叶节点，图标就指明它的状态，比如它是打开的或关闭的。

在缺省状态下，Swing 树利用 TreeCellRenderer 接口绘制节点单元。其缺省实现是 DefaultTreeCellRenderer 类。TreeCellRenderer 和 DefaultTreeCellRenderer 都存储在 javax.swing.tree 程序包中。

要定制树组件节点单元的绘制特性，由 DefaultTreeCellRenderer 类支持的功能通常就能满足。TreeCellRenderer 接口包含下面的方法：

```
public Component getTreeCellRendererComponent(JTree tree,
                                              Object value,
                                              boolean selected,
                                              boolean expanded,
                                              boolean leaf,
                                              int row,
                                              boolean hasFocus)
```

除了执行这个方法以外，类 DefaultTreeCellRenderer 还支持许多“set”和“get”方法。有些“set”方法将在清单 16.4 中演示。

### 提示:

DefaultTreeCellRenderer 类支持用“set”方法定制图标、文本、背景等等树节点。在开始时执行这些方法代替实现 TreeCellRenderer 接口。

尽管在许多环境下“set”方法已经足够了，但有时 DefaultTreeCellRenderer 类给出的方法不符合要求。这种情况下，需要创建一个类，该类实现接口 TreeCellRenderer 的方法。执行类的对象可以通过调用树对象上的方法 setCellRenderer(TreeCellRenderer 覆盖) 设计为树对象。

### 16.4.1 绘制节点单元的代码示例

清单 16.4 演示了怎样在 Swing 树中定制节点单元的绘制。此程序是一个利用类 DefaultTreeCellRenderer 支持的“set”方法的小程序。这样，开始时需要获得要绘制的树组件引用。此程序显示了怎样将节点单元的图标替换为感兴趣的图标、怎样改变节点标签的文本颜色以及怎样改变背景颜色等等。该程序的输出结果显示在图 16-5 中。

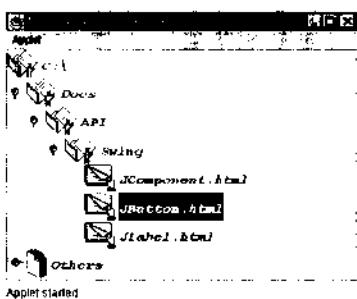


图 16-5 显示带有定制图标和标签的节点的 Swing 小程序

## 清单 16.4 DefaultTreeCellRenderer 示例 (TTreeCellRenderer.java)

```
// Demonstrates how to customize the display of
// the tree nodes (or leaves).

/*
 * <Applet code = TTreeCellRenderer width = 350 height = 300>
 * </Applet>
 */

import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
import java.awt.event.*;

public class TTreeCellRenderer extends JApplet {
    Container container;

    public void init() {
        // 1. Get the handle on applet's content pane.
        container = this.getContentPane();

        // 2. Root node (col0 or root) of the tree.
        DefaultMutableTreeNode root =
            new DefaultMutableTreeNode("C:\\");

        // 3. Create the nodes in column 1.
        DefaultMutableTreeNode col11 =
            new DefaultMutableTreeNode("Docs");
        DefaultMutableTreeNode col12 =
            new DefaultMutableTreeNode("Others");

        // 4. Add the nodes to the root.
        root.add(col11);
        root.add(col12);

        // 5. Create the nodes in column 2.
    }
}
```

```
DefaultMutableTreeNode col21 =
    new DefaultMutableTreeNode("API");

DefaultMutableTreeNode col22 =
    new DefaultMutableTreeNode("Bla...");

// 6.Add these nodes to node 1 in column 1.
col11.add(col21);
col12.add(col22);

// 7.Create the nodes in column 3.
DefaultMutableTreeNode col31 =
    new DefaultMutableTreeNode("Swing");

// 8.Add the node to node 2 in column 2.
col21.add(col31);

// 9.Create the nodes in column 4.
DefaultMutableTreeNode col41 =
    new DefaultMutableTreeNode("JComponent.html");
DefaultMutableTreeNode col42 =
    new DefaultMutableTreeNode("JButton.html");
DefaultMutableTreeNode col43 =
    new DefaultMutableTreeNode("JLabel.html");

// 10.Add the nodes to the node called Swing.
col31.add(col41);
col31.add(col42);
col31.add(col43);

// 11.Attach the root (of nodes) to the tree object.
JTree tree = new JTree(root);

// 12.Get the reference to the existing (default) renderer.
DefaultTreeCellRenderer renderer =
    (DefaultTreeCellRenderer) tree.getCellRenderer();

// 13.Prepare the cell height for the new icons.
tree.setRowHeight(30); // 30 pixels

// 14.Attach the new icons for the leaves nodes when opened,
// and nodes when closed.
renderer.setLeafIcon(new ImageIcon("leafIcon.gif"));
renderer.setOpenIcon(new ImageIcon("fileOpen.gif"));
renderer.setClosedIcon(new ImageIcon("fileClosed.gif"));

// 15.Customize the text colors.
renderer.setFont(new Font("Monospaced", // font name
    Font.BOLD|Font.ITALIC, // font type
    15)); // Font size.
```

```
renderer.setTextNonSelectionColor(Color.blue);
renderer.setTextSelectionColor(Color.white);

// 16. Customize the background of the tree node - cells.
renderer.setBackgroundNonSelectionColor(Color.white);
renderer.setBackgroundSelectionColor(Color.gray);
renderer.setBorderSelectionColor(Color.lightGray);

// 17. Finally, add the tree to a scroll pane and the scroll pane
// to the container.

JScrollPane scrollPane = new JScrollPane(tree);
container.add(scrollPane);

}
```

代码详析

在此小程序的 init () 方法里，片段 1 初始化了对小程序内容窗格的引用。片段 2 - 11 显示了由节点创建树的代码（也见清单 16.1）。片段 12 获得要绘制的缺省树单元引用。方法 getCellRenderer () 返回 TreeCellRenderer 接口类型，TreeCellRenderer 必须为 DefaultTreeCellRenderer 类型的类。这样就可以使用这个类的“set”方法。片段 13 准备了节点单元的高度以适应新的图标，它比缺省图标稍稍大一点。

片段 14 显示了定制叶节点和非叶节点图标的代码。方法 `setLeafIcon()` 为要绘制的单元赋予指定的图标。`setOpenIcon()` 和 `setClosedIcon()` 则为非叶节点的打开和关闭指定了图标。

片段 15 定制了节点单元中文本标签的字体和颜色。方法 `setFont()` 为文本标签赋予了规定的字体。后两个方法规定了当节点处于取消选定和选定状态时文本的颜色。

片段 16 为节点的不同状态（如取消选定和选定状态）定制了树节点单元的背景。方法 `setBorderSelectionColor()` 为节点单元的边界赋予了一种新颜色。注意在片段 14~16 中，基本上通过使用对缺省的绘制单元引用定制了节点单元的绘制方式。所以缺省单元绘制方式不再包含缺省设置，即它已经被定制了。最后，片段 17 将这棵树加到一个滚动窗格上，而此滚动窗格是加在小程序内容窗格上的。

## 16.5 可编辑的树节点

Swing 树通过允许用户编辑项目名来修改数据项（视图）。为了处理对树单元的编辑，Swing 库引入了一个叫做 TreeCellEditor 的设计级接口。此接口代表了树节点单元编辑器的抽象模型。DefaultTreeCellEditor 类实现这个接口，以便为单元编辑器提供缺省模型。接口 TreeCellEditor 中只有一个方法：

```
boolean expanded,
boolean leaf,
int row)
```

要使树节点成为可编辑的，需使参数值为真（true），并调用树对象的 `setEditable()` 方法。为了激活文本字段去执行编辑操作，必须在节点单元上连击三次鼠标、或者中间加一暂停的双击鼠标。完成编辑以后，按回车键确认。注意，当改变数据项时，树模型便激活事件 `TreeModelEvent`。

在清单 16.1 中，只要在片段 11 中简单地加入下面的语句，就可以使树组件成为可编辑的：

```
JTree tree = new JTree(root);
tree.setEditable(true);
```

图 16-6 显示了清单 16.1 中程序的输出结果，它带有一个激活的节点单元。

名为 `DefaultTreeCellEditor` 的树单元的缺省编辑器包含足够的方法来构造所需的编辑器。对多数实际问题来说，这么多的功能应该足够了。通过调用树对象的方法 `getCellEditor()` 可以得到缺省树单元编辑器的引用。这个方法获取 `TreeCellEditor` 接口类型的单元编辑器，它必需指定为 `DefaultTreeCellEditor` 类型的编辑器类。使用缺省树单元编辑器定制不能满足的情况下，只能实现 `TreeCellEditor` 接口来满足需求。

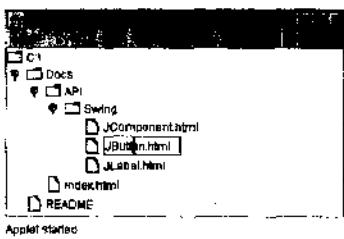


图 16-6 显示可编辑状态下节点的小程序

# 第 17 章 定时器和进度指示器

在程序中，经常会遇到某些需要按周期执行的任务。这些任务以单线程被处理，以使其他过程可以类似地执行。例如，为了执行一个动画，需要以一定的频率、周期性地显示图形，假定它是单线程的。Swing 定时器压缩了这些功能：它基本上就是一个以特定速率触发一个动作的软件元件。定时器在背景中使用单线程去执行任务。

这一章还将介绍进度条、进度监视器和进度监视器输入流。进度条用来指示操作的进度，特别是当这个操作是耗时，或操作仍在进行时用户可能以为系统已经超时。Swing 进度监视器弹出一个进度对话框，通过进度条来显示给定操作的进度。进度监视器输入流指的是从输入流中读取的进度。假如完成给定的任务需要一定的时间，就会弹出一个进度对话框来通知用户。在后面的几节里，将使用合适的例子程序讨论和演示这些问题。

## 17.1 定时器

Swing 定时器是能够以规定速率触发一个动作的软件对象。执行动作的控制速率由定时器的“延时（delay）”指定。延时表示在两个动作之间的时间周期，以毫秒计。

Swing 定时器由类 Timer 表示，该类是 java.lang.Object 的子类。定时器周期性地触发 ActionEvent 类型的事件，该事件由目标监听程序监听。监听程序类必须实现接口 ActionListener 并在方法 actionPerformed() 里装入执行代码。目标监听程序通过它的构造函数注册为定时器。

为了在程序中使用定时器，创建一个定时器对象并执行它的 start() 方法。start() 方法使得定时器以单线程和指定的延时激活事件。也可以通过将方法 setRepeats() 设置为假（false）来使定时器只激活一次事件。在任何例子中，都可以通过分别调用方法 stop() 和 restart() 来停止或重新启动定时器。图 17-1 描述了一个定时器的功能，该定时器周期性地执行一个有初始延时和事件延时的动作。

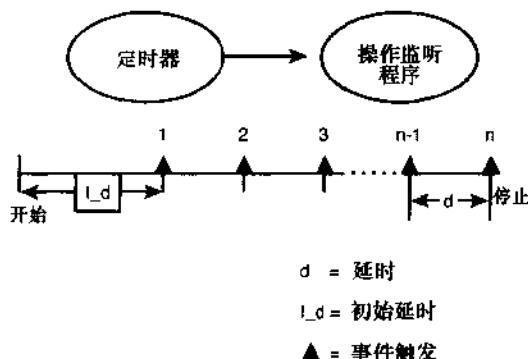


图 17-1 Swing 定时器的功能

为了执行方法 actionPerformed() 中的动作代码，系统可能会变得非常忙。如果定时器的延时设置得不够长，事件将可能排队等候。发生这种情况时，当前的动作代码一执行完，就会没有延迟地马上执行排队等候的事件，那就意味着所有的事件都一个接着一个被激活。为了避免这种情况发生，Swing 定时器对象缺省设置为合并动作代码。为了禁用合并特性，调用逻辑值为假的 setCoalesce() 方法即可。

### 17.1.1 定时器构造函数

Swing 定时器只带有一个构造函数，它要求指定延时和动作监听程序。动作监听程序实现 ActionListener 接口。下面是类 Timer 的构造函数：

```
Publiu Timer(int delay, ActionListener listener)
```

### 17.1.2 定时器代码示例

清单 17.1 是显示标题“Swing can flash!”的小程序。你会发现几个用来启动、重新启动或停止文本标签以不同颜色闪烁的控制按钮。另外，还有一个用来关闭或打开合并特性的切换按钮。注意，事实上合并是缺省的。还有用来改变初始延时和激活事件的定时器延时的滑动条。此程序的输出结果显示在图 17-2 中。

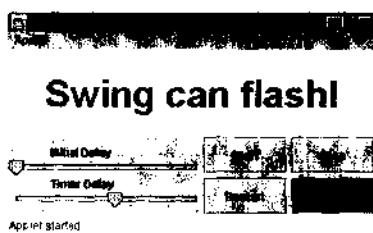


图 17-2 演示 Swing 定时器的小程序

#### 清单 17.1 定时器 (Ttimer.java)

```
/*
 * < Applet code = TTimer width = 400 height = 200 >
 * < /Applet >
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.*;

public class TTimer extends JApplet {
    Container container = null;
```

```
Timer timer = null;
JLabel label = null;
JSlider slider1 = null;
JSlider slider2 = null;
Color[] color = {Color.blue,Color.green,Color.red,
                 Color.yellow,Color.lightGray};

public void init() {
    // 1. Get the handle on the applet's content pane.
    container = this.getContentPane();

    // 2. Create a label and attach it to the applet.
    label = new JLabel("Swing can flash!",JLabel.CENTER);
    label.setBackground(Color.white);
    label.setFont(new Font("Dialog",Font.BOLD,40));
    label.setOpaque(true);
    container.add(label);

    // 3. Create a horizontal box and attach it at the
    // bottom portion of the content pane.
    Box box = Box.createHorizontalBox();
    container.add(box,BorderLayout.SOUTH);

    // 4. Create a vertical box and add it to the horizontal box.
    Box vbox1 = Box.createVerticalBox();
    box.add(vbox1);

    // 5. Create labels and sliders and attach them to
    // a vertical box.
    JLabel initDelay = new JLabel("Initial Delay",JLabel.CENTER);
    initDelay.setPreferredSize(new Dimension(200,25));
    vbox1.add(initDelay);
    slider1 = new JSlider(JSlider.HORIZONTAL,0,60000,0);
    slider1.addChangeListener(new SliderListener());
    vbox1.add(slider1);
    JLabel delay = new JLabel("Timer Delay",JLabel.CENTER);
    delay.setPreferredSize(new Dimension(200,25));
    vbox1.add(delay);
    slider2 = new JSlider(JSlider.HORIZONTAL,0,2000,1000);
    slider2.addChangeListener(new SliderListener());
    vbox1.add(slider2);

    // 6. Create another vertical box and add it to the
    // horizontal box.
    Box vbox2 = Box.createVerticalBox();
    box.add(vbox2);
```

```
// 7.Create a Swing panel with grid layout and add it to
// the vertical box created in Snippet 6.
 JPanel panel = new JPanel();
 panel.setLayout(new GridLayout(2,2,5,5));
 vbox2.add(panel);

// 8.Create Start, Stop, and Restart buttons, and add them
// to the panel.
 String[] buttonLabels = {"Start", "Stop", "Restart"};
 for (int i = 0; i < buttonLabels.length; i++) {
 JButton button = new JButton(buttonLabels[i]);
 button.addActionListener(new ButtonListener());
 panel.add(button);
 }

// 9.Add a toggle button to the panel.
 JToggleButton toggleButton = new JToggleButton("Coalesce");
 toggleButton.setSelected(true); // since coalesced by default
 toggleButton.addActionListener(new ToggleButtonListener());
 panel.add(toggleButton);

// 10.Create a timer object with prescribed delay and
// initial delay retrieved from the corresponding
// slider objects.
 timer = new Timer(slider2.getValue(),new TimerListener());
 timer.setInitialDelay(slider1.getValue()); //initial delay
}

// 11.Listener class that implements the "action code."
class Timerlistener implements ActionListener {
 int i;

 public void actionPerformed(ActionEvent e) {
 if (i == color.length) {
 i = 0;
 label.setForeground(color[i]);
 }
 else {
 label.setForeground(color[i]);
 }
 label.repaint();
 i++;
 }
}

// 12.Button listener that is used to start,stop,or
// restart the timer.
```

```
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        if (button.getText() == "Start") {
            timer.start();
        }
        else if (button.getText() == "Stop") {
            timer.stop();
        }
        else if (button.getText() == "Restart") {
            timer.restart();
        }
    }
}

// 13. Toggle button listener to change coalescing status.
class ToggleButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (timer.isCoalesce() == false)
            timer.setCoalesce(true);
        else if (timer.isCoalesce() == true)
            timer.setCoalesce(false);
    }
}

// 14. Slider listener to alter the initial delay or
// timer delay.
class SliderListener implements ChangeListener {
    public void stateChanged(ChangeEvent e) {
        JSlider slider = (JSlider) e.getSource();
        if (slider == slider1) {
            timer.setInitialDelay(slider1.getValue());
        }
        else if (slider == slider2) {
            timer.setDelay(slider2.getValue());
        }
    }
}
```

## 代码详析

小程序 Timer 声明了不同的组件作为它的数据成员。字段颜色由显示颜色的数组表示，它还可产生闪烁效果。片段 1 得到小程序内容窗格的句柄。片段 2 创建了一个 Swing 标签并

将它连接到内容窗格上。稍后，将使用定时器来操作这个标签以显示不同的颜色。

片段 3~5 使用框架容器来定位滑动条和各个指示器标签。类似地，片段 6~9 使用框架来加上 Start、Stop 和 Restart 按钮，以及用于合并的切换按钮。片段 10 创建了一个给各个滑动条设置赋予延时值的定时器对象。定时器构造函数还执行了实现代码的目标监听程序。下一步，定时器被设置了初始延时，如果单击了 Start 按钮，定时器将在初始延时内启动。

片段 11~14 分别是定时器、按压式按钮、切换按钮和滑动条的监听程序类。在定时器监听程序里，标签“Swing can flash!”的前景被赋予为不同的颜色以产生闪烁效果。这个代码放在方法 actionPerformed() 里。注意，不需要创建任何线程来处理标题的闪烁效果，定时器对象以单线程处理这个过程。

当单击 Start 按钮时，就会调用定时器对象的 start() 方法。这个功能已在 ButtonListener 类中实现。start() 方法启动 actionPerformed() 之中的代码。类似地，其他方法 stop() 和 restart() 的代码已经放入按钮监听程序里。切换按钮监听程序按照它的状态赋予合并属性。类 SliderListener 包含调节定时器延时和定时器初始延时的代码。

## 17.2 进度条

对人机交互的研究说明，用户一般不满意多于两秒的软件响应时间。假如延时时间很长，用户可能会推断系统已经停止运行了。所以，为了使用户在等待软件完成定时任务时更满意，使用进度条应是一个好主意。

Swing 中的进步条通过框在一个区间里的整数值显示进行中过程的进度情况。进度条通过显示正在着色的颜色条完成的百分比和及时显示的数值来通知用户进度情况。

### 说明：

进度条很好地指示了超过 5 秒的延时。假如延时是从一到五秒，就可以使用“系统忙碌”指针。

Swing 进度条由类 JProgressBar 表示，它是如图 17-3 所示的 JComponent 的扩展类。这个类也实现 SwingConstants 接口。在 AWT 中没有等价的组件。就像 Swing 滑动条一样，进度条由最小值、最大值和当前值来定义。即其数据模型可以由接口 BoundedRangeModel 来表示。这个数据模型接口在程序包 javax.swing 中是可用的。

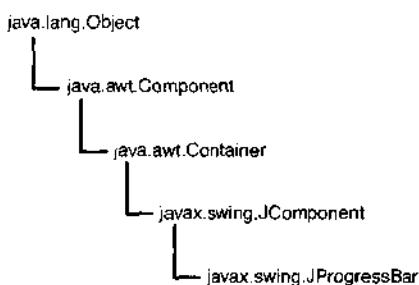


图 17-3 JProgressBar 的类层次结构

### 17.2.1 JProgressBar 构造函数

为了在应用程序中实现 Swing 进度条，可以利用它的任何一个构造函数来创建 JProgressBar 类型的对象。这些构造函数需要一些参数，如边界区域的数据模型、方向及最大值和最小值。下面是 JProgressBar 类的构造函数：

```
public JProgressBar()
```

上述函数创建了一个水平进度条，它的最大值和最小值分别是缺省值 0 和 100。可以调用 setMinimum() 和 setMaximum() 方法，来改变这些值。

```
public JProgressBar(int orient)
```

创建了一个有指定 HORIZONTAL 方向和 VERTICAL 方向的进度条。这些属性分别由接口 SwingConstants 的常数表示。它的最大值和最小值使用缺省值。

```
public JProgressBar(int min, int max)
```

创建了有指定最小值和最大值的进度条，缺省的方向为水平方向。

```
public JProgressBar(int orient, int min, int max)
```

创建了有指定方向、最小值及最大值的进度条。

```
public JProgressBar(BoundedRangeModel newModel)
```

需要 BoundedRangeModel 类型的数据模型对象，它包含指定进度条范围的数值。要查看有关这个接口的成员，请参考附录 A “JFC Swing 快速参考”。还可以使用类 DefaultBoundedRangeModel，它已经实现了有缺省设置的 BoundedRangeModel 类。

### 17.2.2 进度条代码示例

清单 17.2 利用 Swing 进度条指示用特定自然数表示的进度。可以在文本字段里键入最高数来指定它，该文本字段显示在小程序中。当单击 Start 按钮时，计算操作开始并且立即用进度条显示计算完成的百分比。在小程序右上角的文本字段里显示了自然数的总和（见图 17-4）。

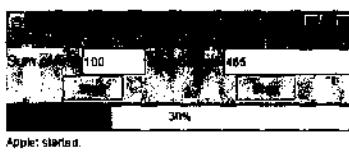


图 17-4 显示 Swing 进度条的小程序

清单 17.2 进度条 (TJProgressBar.java)

```
/*
 * <Applet code=TJProgressBar width=400 height=100>
 * </Applet>
```

```
*/  
  
import javax.swing.*;  
import javax.swing.border.*;  
import java.awt.*;  
import java.awt.event.*;  
  
public class TJProgressBar extends JApplet {  
    Container container = null;  
    JButton startButton, stopButton;  
    JTextField inputTextField, outputTextField;  
    JProgressBar pBar = null;  
    Timer timer = null;  
  
    static int sum = 0;  
    static int counter = 0;  
  
    public void init() {  
        // 1. Get the handle on the content pane and  
        // assign the grid layout.  
        container = this.getContentPane();  
        container.setLayout(new GridLayout(3,1));  
  
        // 2. Add a horizontal box to the container.  
        Box hbox1 = Box.createHorizontalBox();  
        container.add(hbox1);  
  
        // 3. Add labels and input and output text fields  
        // to the horizontal box.  
        hbox1.add(Box.createHorizontalGlue());  
        JLabel label1 = new JLabel("Sum of first ",JLabel.LEFT);  
        label1.setFont(new Font("Dialog",Font.PLAIN,15));  
        hbox1.add(label1);  
  
        inputTextField = new JTextField("100",4);  
        hbox1.add(inputTextField);  
  
        JLabel label2 = new JLabel(" numbers is ",JLabel.LEFT);  
        label2.setFont(new Font("Dialog",Font.PLAIN,15));  
        hbox1.add(label2);  
  
        outputTextField = new JTextField(10);  
        hbox1.add(outputTextField);  
        hbox1.add(Box.createHorizontalGlue());  
  
        // 4. Add another horizontal box to the container.  
        Box hbox2 = Box.createHorizontalBox();  
        container.add(hbox2);  
  
        // 5. Add Start and Stop buttons to the container.
```

```
startButton = new JButton("Start");
startButton.addActionListener(new ButtonListener());
hbox2.add(Box.createHorizontalGlue());
hbox2.add(startButton);
hbox2.add(Box.createHorizontalGlue());
stopButton = new JButton("Stop");
stopButton.addActionListener(new ButtonListener());
hbox2.add(Box.createHorizontalGlue());
hbox2.add(stopButton);
hbox2.add(Box.createHorizontalGlue());

// 6.Create and add a progress bar to the remaining
// display area.
pBar = new JProgressBar();
pBar.setStringPainted(true);
Border border = BorderFactory.createLineBorder(Color.red, 2);
pBar.setBorder(border);
pBar.setBackground(Color.white);
pBar.setForeground(Color.blue);
pBar.setMinimum(0);

pBar.setMaximum(Integer.parseInt(inputTextField.getText()));
container.add(pBar);

// 7.Create a timer object.
timer = new Timer(0,new TimerListener());
}

// 8.Timer listener that computes the sum of natural numbers,
// indicates the computation progress, and displays the result.
class TimerListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (Integer.parseInt(inputTextField.getText()) > 0){
            counter++;
            sum = sum+counter;
            pBar.setValue(counter);
            outputTextField.setText(Integer.toString(sum));
        }
        else {
            outputTextField.setText("0");
        }

        if (counter >= Integer.parseInt(inputTextField.getText()))
            timer.stop();
    }
}
```

```
// 9. Button Listener that actually starts or stops the
// process.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        if (button.getText() == "Start") {
            outputTextField.setText("");
            if (inputTextField.getText() != "") {
                pBar.setMaximum(Integer.parseInt(
                    inputTextField.getText()));
                sum = 0;
                counter = 0;
                timer.start();
            }
        }
        else if (button.getText() == "Stop") {
            timer.stop();
            outputTextField.setText("");
            sum = 0;
            counter = 0;
            pBar.setValue(0);
        }
    }
}
```

## 代码详析

这个小程序的开头声明了它的内容窗格、Start 和 Stop 按钮、输入及输出文本字段、进度条和定时器作为它的数据成员。在 init () 方法里，片段 1 得到小程序内容窗格的句柄，并将小程序赋予网格布局。片段 2 将一水平方框连到容器上，此容器上还有两个标签和输入及输出文本字段（片段 3）。

片段 4 将另一个水平方框加到内容窗格上，片段 5 附加上 Start 和 Stop 按钮。片段 6 创建一个进度条并将其附在小程序的底部。

片段 8 执行定时器监听程序。在 actionPerformed () 方法里，计算出指定自然数的总和。进度条的当前值已修改，总和显示在输出文本字段里。片段 9 显示了执行开始或停止计算的代码的按钮监听程序。此片段还重新设置了计数器和输出文本字段。

## 17.3 进度监视器

为了监视耗时任务的进度，Swing 库支持进度监视器。进度监视器就是一个对象，它监

视指定任务的进度并通过弹出一对话框来指示进度状态。该对话框显示一个进度条，其中有最大值、最小值和当前值。

可以使用 `setMillisToDecideToPopup()` 方法来决定程序需要多长时间弹出或不弹出进度对话框。缺省设置是 500 毫秒。一旦对话框已经弹出，使用方法 `setMillisToPopup()` 来决定程序应等待多长时间再开始显示状况。程序等待的时间越长，显示的状态就越精确，缺省设置是 2000 毫秒。

Swing 进度监视器由类 `ProgressMonitor` 表示，它是类 `java.lang.Object` 的扩展类。类 `ProgressMonitor` 存储在 `javax.swing` 程序包里。

### 17.3.1 ProgressMonitor 构造函数

为了创建进度监视器，可以使用 `ProgressMonitor` 构造函数，它需要的信息如：附在进度对话框上的 `parentComponent`、指示操作的消息（`message`）、指示操作状态的注释（`note`）和进度区域的最小边界及最大边界。注意，由 `message` 指示的描述消息在进度过程中不能改变，描述操作状态的注释可以通过 `setNote()` 方法改变。下面是 `ProgressMonitor` 类的构造函数：

```
public ProgressMonitor(Component parentComponent,
    Object message,
    String note,
    int min,
    int max)
```

### 17.3.2 ProgressMonitor 代码示例

清单 17.3 是一个计算不超过指定数的自然数的总和。当单击 Start 按钮时，总和显示在右边的文本字段里，直至达到最大值。在显示计算进度的指定时间后，会弹出进度条（见图 17-5）。此程序本质上使用进度监视器来指示进度。

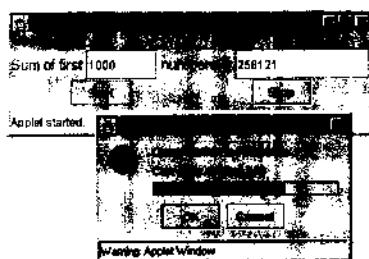


图 17-5 显示进度监视器的小程序

清单 17.3 ProgressMonitor (TprogressMonitor.java)

```
/*
 * <Applet code = TProgressMonitor width = 400 height = 100>
 * </Applet>
```

```
/*
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

public class TProgressMonitor extends JApplet {
    Container container = null;
    JButton startButton, stopButton;
    JTextField inputTextField, outputTextField;
    ProgressMonitor pMonitor = null;
    Timer timer = null;
    static int sum = 0;
    static int counter = 0;

    public void init() {
        // 1. Get the handle on the content pane and
        // assign the grid layout.
        container = this.getContentPane();
        container.setLayout(new GridLayout(2,1));
        // 2. Add a horizontal box to the container.
        Box hbox1 = Box.createHorizontalBox();
        container.add(hbox1);
        // 3. Add labels and input and output text fields
        // to the horizontal box.
        hbox1.add(Box.createHorizontalGlue());
        JLabel label1 = new JLabel("Sum of first ",JLabel.LEFT);
        label1.setFont(new Font("Dialog",Font.PLAIN,15));
        hbox1.add(label1);
        inputTextField = new JTextField("100",4);
        hbox1.add(inputTextField);
        JLabel label2 = new JLabel(" numbers is ",JLabel.LEFT);
        label2.setFont(new Font("Dialog",Font.PLAIN,15));
        hbox1.add(label2);
        outputTextField = new JTextField(10);
        hbox1.add(outputTextField);
        hbox1.add(Box.createHorizontalGlue());
        // 4. Add another horizontal box to the container.
        Box hbox2 = Box.createHorizontalBox();
        container.add(hbox2);
        // 5. Add Start and Stop buttons to the container.
```

```
startButton = new JButton("Start");
startButton.addActionListener(new ButtonListener());
hbox2.add(Box.createHorizontalGlue());
hbox2.add(startButton);
hbox2.add(Box.createHorizontalGlue());
stopButton = new JButton("Stop");
stopButton.addActionListener(new ButtonListener());
hbox2.add(Box.createHorizontalGlue());
hbox2.add(stopButton);
hbox2.add(Box.createHorizontalGlue());

// 6.Create a timer object.
timer = new Timer(0,new TimerListener());
}

// 7.Timer listener that computes the sum of natural numbers,
// indicates the computation progress, and displays the result.
class TimerListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (Integer.parseInt(inputTextField.getText()) > 0) {
            counter++;
            sum = sum + counter;
            pMonitor.setProgress(counter);
            pMonitor.setNote("Currently Adding " + counter);
            outputTextField.setText(Integer.toString(sum));
        }
        else {
            outputTextField.setText("0");
        }
        if (counter >= Integer.parseInt(inputTextField.getText())){
            timer.stop();
            startButton.setEnabled(true);
        }
    }
}

// 8.Button listener that actually starts or stops the process.
class ButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        if (button.getText() == "Start") {
            startButton.setEnabled(false);
            //9.Create a progress monitor.
            pMonitor = new ProgressMonitor(container,
```

代码详析

在 init () 方法里，片段 1 得到内容窗格的句柄，并准备了两行和一列的网格布局。

片段 2 创建了一个水平方框并将其加到容器上。片段 3 创建了一对标签和输入输出文本字段，并将它们加到前一片段创建的水平方框里。片段 4 创建了另一个水平方框并把它附在容器上，片段 5 再将 Start 和 Stop 按钮加到这个方框里。

片段 6 创建了一个没有延时的定时器对象。片段 7 显示了定时器监听程序，在定时器对象每次调用时修改自然数总和。在 actionPerformed() 方法中的 setProgress() 函数修改进度条的当前值。另一个 setNote() 方法循环地修改显示在进度对话框里的计数器值。

片段 8 的代码显示了选择 Start 或 Stop 按钮时将要发生的事情。在 if 语句的主体中，创建了带有进度条最小最大值的进度监视器。在下一个语句中，方法 setMillisToPopup() 为在计算开始后显示的进度监视器赋予了时间周期。当单击 Stop 按钮时，进度监视器就会关闭，计算值也会重置。

#### 17.4 输入流的进度监视器

Swing 库支持一个叫做 `ProgressMonitorInputStream` 的单独的类，以监视读取输入流的进度。这个类是一种监视过滤器，它是 `FilterInputStream` 类的扩展类，如图 17-6 所示。通常，

在应用程序读入文件时使用这个类。

`ProgressMonitorInputStream` 类实际地创建了一个进度监视器对话框。可以指定进度监视器对话框（当它弹出时）。对话框显示了指示读取输入流进度的进度条。你不必指定进度条的最大值或最小值，因为 API 可以根据输入源程序（比如一个文件）的大小计算出这个信息。

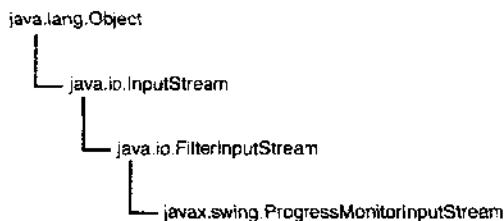


图 17-6 `ProgressMonitorInputStream` 的类层次结构

这个进度对话框包含了 `OK` 按钮和 `Cancel` 按钮。如果单击 `OK` 按钮，就会关闭进度对话框，但是会继续完成输入流的读取操作。如果单击 `Cancel` 按钮，对输入流的读取将中断，而且会抛出一个 `InterruptedException` 异常。

#### 说明：

像其他的输入流一样，在完成对指定输入流的读取后，`ProgressMonitorInputStream` 类型的流将关闭。

### 17.4.1 `ProgressMonitorInputStream` 构造函数

为了在应用程序中实现进度监视器输入流过滤器，可使用下列构造函数创建 `ProgressMonitorInputStream` 类的对象：

```

public ProgressMonitorInputStream(Component parentComponent,
                                  Object message,
                                  InputStream input)
    
```

这个构造函数需要像 `message parentComponent` 这样的说明性参数，来附加将要显示的对话框和输入流对象。

### 17.4.2 `ProgressMonitorInputStream` 代码示例

清单 17.4 是一个简单地读入文件并将它写在计算机屏幕上的应用程序。需要在如下的命令行里指定文件名，例如 `file.txt`。

```
java TProgressMonitorIS < file.txt >
```

这个程序开始读取文件，如果读取过程要花很多时间，就会弹出一个带有进度条的进度监视器对话框。要预先声明弹出的时间周期，否则使用缺省的时间周期。这个程序使用了一个框架作为进度监视器对话框的父框架。当在命令提示符状态下运行此程序时，此程序开始读取文件并且显示附在框架上的进度监视器，如图 17-7 所示。

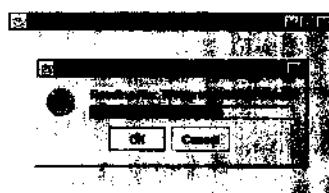


图 17-7 显示文件输入进度监视器的小程序

## 清单 17.4 ProgressMonitorInputStream (TProgressMonitorIS.java)

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

class TProgressMonitorIS {
    static ProgressMonitorInputStream progressInput = null;
    static JFrame frame = null;
    static WindowEventHandler handler = null;

    public static void main(String[] args) {
        // 1. Create a frame and get its content pane.
        frame = new JFrame("Parent Frame");
        Container container = frame.getContentPane();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(
            WindowConstants.DISPOSE_ON_CLOSE);
        frame.setSize(350, 200);
        frame.show();

        // 2. Event handler for closing the window.
        handler = new WindowEventHandler();
        frame.addWindowListener(handler);

        // 3. Give the correct usage at the command line.
        // in case it is needed
        if (args.length != 1) {
            System.out.println("Legal Usage: "
                "java TProgressMonitorIS <file> ");
            System.exit(-1);
        }

        try {
            // 4. Create a file and file input stream objects.
            File file = new File(args[0]);
            FileInputStream inputStream = new FileInputStream(file);

```

```
// 5. Create the progress monitor for the input stream.  
progressInput = new ProgressMonitorInputStream(  
    container,  
    "Reading File: " + args[0],  
    inputStream);  
  
// 6. Get a handle on the progress monitor of the  
// progressInput object, and assign the time periods  
// to decide to pop up and to pop up the dialog box.  
ProgressMonitor monitor =  
    progressInput.getProgressMonitor();  
monitor.setMillisToDecideToPopup(0);  
monitor.setMillisToPopup(0);  
  
// 7. Read the bytes and write on the screen.  
int curByte;  
while((curByte = progressInput.read()) != -1) {  
    System.out.print((char) curByte);  
  
    try {  
        // 8. Make this thread sleep, if the process  
        // is too quick to notice the progress dialog box.  
        Thread.sleep(0); // Set this to some value,  
        // say 100, on faster machines  
    } catch (InterruptedException ie) {  
        System.out.println(  
            "Thread Interruption Occurred!");  
    }  
  
    // 9. Close the progress input stream filter.  
    if(monitor.isCanceled()) {  
        progressInput.close();  
    }  
}  
10.//Close the progressInputstream.  
progressInput.close();  
} catch (Exception e) {  
    System.out.println("....File Reading Interrupted....");  
}  
}  
}  
  
// 11. Create the listener class for window closing.  
class WindowEventHandler extends WindowAdapter {  
    public void windowClosing(WindowEvent evt) {  
        System.exit(-1);  
    }  
}
```

## 代码详析

在此应用程序的 main() 方法里，片段 1 创建了一个框架并得到其内容窗格的句柄。片段 2 注册了一个监听程序对象（见片段 11）来关闭此框架。如果命令行的语法不正确，片段 3 就会给出反馈。此代码还显示要使用的正确语法。

在片段 4 中，创建了一个文件对象和一个文件输入流对象。片段 5 创建了一个进度监视器输入流对象。注意，此构造函数需要这个对象为进度监视器对话框弹出父框架。这个构造函数将文件输入流作为参数值。在片段 6 中，获得进度监视器对话框的句柄。根据这个引用，给用来决定是否弹出以及何时弹出进度对话框的时间周期赋值。片段 7 显示了通过进度监视器从输入流中读取字节的代码。进度监视器作为跟踪读取输入流进度的过滤器使用。这个代码还将此特性发送到计算机屏幕上。片段 8 显示的代码可以用来控制观察进度对话框的当前线程的速度。

如果单击了进度对话框上的 Cancel 按钮，片段 9 就会关闭进度监视器输入流。最后，必须关闭进度监视器输入流，如片段 10 所示。

# 第三部分 语 法 参 考

附录 A JFC Swing 快速参考

附录 B AWT 事件类快速参考

附录 C Swing 中重要的 AWT 类

附录 D 辅助选项快速参考



## 附录 A JFC Swing 快速参考

这个快速参考包括四个部分：附录 A，“JFC Swing 快速参考”；附录 B，“AWT 事件类快速参考”；附录 C，“Swing 中重要的 AWT 类”；附录 D，“辅助选项快速参考”。附录 A 给出了类的层次图，以及 Swing 程序包中类和接口的参考资料。不过，在这里省略了具有特殊外观的程序包的详细资料，例如 javax.swing.plaf.metal、com.sun.java.swing.plaf.motif 等。为了更易于应用，在附录中提供了每一个接口、类和内部类的简短描述。

### 怎样使用快速参考

在“Swing 程序包详细描述”中，开始是程序包的名字，然后是包中的接口和类。其中首先给出了 Swing 库中程序包的详细描述，然后是关于接口和类的详细描述。可以在给定程序包内按字母顺序搜索访问这些类和接口。

附录 B 和附录 C 的快速参考覆盖了 AWT 事件类和重要的 AWT 类。请记住，在实现 Swing 组件时需要 AWT 事件和布局类。附录 D 给出了 JFC 中包含的可访问性 API 的快速参考。要访问附录 B、C 和 D 中的类和接口，同样可以用上一段中所介绍的方法。

#### 说明：

在类和接口的详细描述中，如果在内部类字段、构造函数或方法的前面没有范围修饰符（如 public、protected 或 package），则可以假设它是 public 的。

### Swing 程序包详细描述

下面提供 Swing 程序包中接口和类的快速阅览。

#### 程序包名：javax.swing

该程序包包括一系列轻量级的 Swing 组件，它们是独立于平台的，而且可以由用户配置成所期望的外观类型。

#### Interfaces

Action	MenuItem
BoundedRangeModel	MutableComboBoxModel
ButtonModel	Renderer
CellEditor	RootPaneContainer
ComboBoxEditor	Scrollable
ComboBoxModel	ScrollPaneConstants
DesktopManager	SingleSelectionModel

Icon	SwingConstants
JComboBox.KeySelectionManager	UIDefaults.ActiveValue
ListCellRenderer	UIDefaults.LazyValue
ListModel	WindowConstants
ListSelectionModel	

## Classes

AbstractAction	JPanel
AbstractButton	JPasswordField
AbstractListModel	JPopupMenu
BorderFactory	JPopupMenu.Separator
Box	JProgressBar
Box.Filler	JRadioButton
BoxLayout	JRadioButtonMenuItem
ButtonGroup	JRootPane
CellRendererPane	JScrollBar
DebugGraphics	JScrollPane
DefaultBoundedRangeModel	JSeparator
DefaultButtonModel	JSlider
DefaultCellEditor	JSplitPane
DefaultComboBoxModel	JTabbedPane
DefaultDesktopManager	JTable
DefaultFocusManager	JTextArea
DefaultListCellRenderer	JTextField
DefaultListCellRenderer.UIResource	JTextPane
DefaultListModel	JToggleButton
DefaultListSelectionModel	JToggleButton.ToggleButtonModel
DefaultSingleSelectionModel	JToolBar
FocusManager	JToolBar.Separator
GrayFilter	JToolTip
ImageIcon	JTree
JApplet	JTree.DynamicUtilTreeNode
JButton	JTree.EmptySelectionModel
JCheckBox	JViewport
JCheckBoxMenuItem	JWindow
JColorChooser	KeyStroke
JComboBox	LookAndFeel
JComponent	MenuSelectionManager
JDesktopPane	OverlayLayout
JDialog	ProgressMonitor
JEditorPane	ProgressMonitorInputStream
JFileChooser	RepaintManager
JFrame	ScrollPaneLayout

JInternalFrame	ScrollPaneLayout.UIResource
JInternalFrame.JDesktopIcon	SizeRequirements
JLabel	SwingUtilities
JLayeredPane	Timer
JList	ToolTipManager
JMenu	Defaults
JMenuBar	UIManager
JMenuItem	UIManager.LookAndFeelInfo
JOptionPane	ViewportLayout

**Exception**

UnsupportedLookAndFeelException

**程序包名： javax.swing.border**

此程序包包含的接口和类能在 Swing 组件的周围绘制复杂的、使人印象深刻的边界。

**Interface**

Border

**Classes**

AbstractBorder	LineBorder
BevelBorder	MatteBorder
CompoundBorder	SoftBevelBorder
EmptyBorder	TitledBorder
EtchedBorder	

**程序包名： javax.swing.colorchooser**

此程序包包含 javax.swing 程序包中 JColorChooser 组件使用的接口和类。

**Interface**

ColorSelectionModel

**Classes**

AbstractColorChooserPanel
ColorChooserComponentFactory
DefaultColorSelectionModel

**程序包名： javax.swing.event**

此程序包包含 Swing 组件的事件接口和监听程序类。

**Interfaces**

AncestorListener	MenuKeyListener
CaretListener	MouseListener
CellEditorListener	MouseInputListener
ChangeListener	PopupMenuListener

DocumentEvent	TableColumnModelListener
DocumentEvent.ElementChange	TableModelListener
DocumentListener	TreeExpansionListener
HyperlinkListener	TreeModelListener
InternalFrameListener	TreeSelectionListener
ListDataListener	TreeWillExpandListener
ListSelectionListener	UndoableEditListener
MenuDragMouseListener	

**Classes**

AncestorEvent	MouseEvent
CaretEvent	MouseEvent
ChangeEvent	MouseInputAdapter
DocumentEvent.EventType	PopupMenuEvent
EventListenerList	SwingPropertyChangeSupport
HyperlinkEvent	TableColumnModelEvent
HyperlinkEvent.EventType	TableModelEvent
InternalFrameAdapter	TreeExpansionEvent
InternalFrameEvent	TreeModelEvent
ListDataEvent	TreeSelectionEvent
ListSelectionEvent	UndoableEditEvent
MenuDragMouseEvent	

**程序包名： javax.swing.filechooser**

此程序包包含 javax.swing 程序包中 JFileChooser 组件使用的接口和类。

**Classes**

FileFilter
FileSystemView
Fileview

**程序包名： javax.swing.plaf**

此程序包提供了一些接口和抽象类，它们用于对可插入外观性能提供支持。

**Interface**

UIResource
------------

**Classes**

BorderUIResource
BorderUIResource.BevelBorderUIResource
BorderUIResource.CompoundBorderUIResource
BorderUIResource.EmptyBorderUIResource
BorderUIResource.FchedBorderUIResource
BorderUIResource.LineBorderUIResource

BorderUIResource, MatteBorderUIResource  
BorderUIResource, TitledBorderUIResource  
ButtonUI  
ColorChooserUI  
ColorUIResource  
ComboBoxUI  
ComponentUI  
DesktopIconUI  
DesktopPaneUI

### 程序包名: javax.swing.plaf.basic

此程序包提供按照基本外观类型建立的用户接口对象。

#### Interface

ComboPopup

#### Classes

BasicArrowButton	asicInternalFrameTitlePane
BasicBorders	BasicInternalFrameUI
BasicBorders.ButtonBorder	BasicLabelUI
BasicBorders.FieldBorder	BasicListUI
BasicBorders.MarginBorder	BasicLookAndFeel
BasicBorders.MenuBarBorder	BasicMenuBarUI
BasicBorders.RadioButtonBorder	BasicMenuItemUI
BasicBorders.SplitPaneBorder	BasicMenuUI
BasicBorders.ToggleButtonBorder	BasicOptionPaneUI
BasicButtonListener	BasicOptionPaneUI.ButtonAreaLayout
BasicButtonUI	BasicPanelUI
BasicCheckBoxMenuItemUI	BasicPasswordFieldUI
BasicCheckBoxUI	BasicPopupMenuSeparatorUI
BasicColorChocserUI	BasicPopupMenuUI
BasicComboBoxEditor	BasicProgressBarUI
BasicComboBoxEditor.UIResource	BasicRadioButtonMenuItemUI
BasicComboBoxRenderer	BasicRadioButtonUI
BasicComboBoxRenderer.UIResource	BasicScrollBarUI
BasicComboBoxUI	BasicScrollPaneUI
BasicComboBoxPopup	BasicSeparatorUI
BasicDesktopIconUI	BasicSliderUI
BasicDesktopPaneUI	BasicSplitPaneDivider
BasicDirectoryModel	BasicSplitPaneUI
BasicEditorPaneUI	BasicTabbedPaneUI
BasicFileChooserUI	BasicTableHeaderUI
BasicGraphicsUtils	BasicTableUI
BasicIconFactory	BasicTextAreaUI

BasicTextFieldUI	BasicToolBarSeparatorUI
BasicTextPaneUI	BasicToolBarUI
BasicTextUI	BasicToolTipUI
BasicTextUI.BasicCaret	BasicTreeUI
BasicTextUI.BasicHighlighter	BasicViewportUI
BasicToggleButtonUI	DefaultMenuLayoutUI

### 程序包名: javax.swing.plaf.metal

此程序包包含按照 Metal 或 Java 外观类型建立的用户接口对象。

#### Classes

DefaultMetalTheme	MetalIconFactory.FileIcon16
MetalBorders	MetalIconFactory.FolderIcon16
MetalBorders.ButtonBorder	MetalIconFactory.TreeControlIcon
MetalBorders.Flush3DBorder	MetalIconFactory.TreeFolderIcon
MetalBorders.InternalFrameBorder	MetalIconFactory.TreeLeafIcon
MetalBorders.MenuBarBorder	MetalInternalFrameUI
MetalBorders.MenuItemBorder	MetalLabelUI
MetalBorders.PopupMenuBorder	MetalLookAndFeel
MetalBorders.RolloverButtonBorder	MetalPopupMenuSeparatorUI
MetalBorders.ScrollPaneBorder	MetalProgressBarUI
MetalBorders.TextFieldBorder	MetalRadioButtonUI
MetalBordersToolBarBorder	MetalScrollBarUI
MetalButtonUI	MetalScrollPane
MetalCheckBoxIcon	MetalScrollPaneUI
MetalCheckBoxUI	MetalSeparatorUI
MetalComboBoxButton	MetalSliderUI
MetalComboBoxEditor	MetalSplitPaneUI
MetalComboBoxEditor.UIResource	MetalTabbedPaneUI
MetalComboBoxIcon	MetalTextFieldUI
MetalComboBoxUI	MetalTheme
MetalDesktopIconUI	MetalToggleButtonUI
MetalFileChooserUI	MetalToolBarUI
MetalIconFactory	MetalToolTipUI
	MetalTreeUI

### 程序包名: javax.swing.plaf.multi

此程序包包含的组件对象使用户能将辅助外观类型和缺省外观类型相结合，以产生多元外观类型。

#### Classes

MultiButtonUI	MultiColorChooserUI
---------------	---------------------

MultiComboBoxUI	MultiScrollPaneUI
MultiDesktopIconUI	MultiSeparatorUI
MultiDesktopPaneUI	MultiSliderUI
MultiFileChooserUI	MultiSplitPaneUI
MultiInternalFrameUI	MultiTabbedPaneUI
MultiLabelUI	MultiTableHeaderUI
MultiListUI	MultiTableUI
MultiLookAndFeel	MultiTextUI
MultiMenuBarUI	MultiToolBarUI
MultiMenuItemUI	MultiToolTipUI
MultiOptionPaneUI	MultiTreeUI
MultiPanelUI	MultiViewportUI
MultiPopupMenuUI	
MultiProgressBarUI	
MultiScrollBarUI	

**程序包名： javax.swing.table**

此程序包包含 javax.swing 中由 JTable 支持的接口和类。

**Interface**

TableCellEditor	TableModel
TableCellRenderer	TableModel

**Classes**

AbstractTableModel	DefaultTableModel
DefaultTableCellRenderer	JTableHeader
DefaultTableCellRenderer.UIRenderer	TableColumn
DefaultTableColumnModel	

**程序包名： javax.swing.text**

此程序包包含处理 Swing 文本组件的接口和类。

**Interface**

AbstractDocument.AttributeContext	Highlighter.Highlight
AbstractDocument.ContentHighlighter	HighlightPainter
AttributeSet	Keymap
AttributeSet.CharacterAttribute	MutableAttributeSet
AttributeSet.ColorAttribute	Position
AttributeSet.FontAttribute	Style
AttributeSet.ParagraphAttribute	StyledDocument
Caret	ViewableView
Document	TabExpander
Element	ViewFactory
Highlighter	

## Classes

AbstractDocument  
AbstractDocument.ElementEdit  
AbstractWriter  
BoxView  
ComponentView  
CompositeView  
DefaultCaret  
DefaultEditorKit  
DefaultEditorKit.BeepAction  
DefaultEditorKit.CopyAction  
DefaultEditorKit.CutAction  
DefaultEditorKit.DefaultKeyTypedAction  
DefaultEditorKit.InsertBreakAction  
DefaultEditorKit.InsertContentAction  
DefaultEditorKit.InsertTabAction  
DefaultEditorKit.PasteAction  
DefaultHighlighter  
DefaultHighlighter.DefaultHighlightPainter  
DefaultStyledDocument  
DefaultStyledDocument.AttributeUndeleteEdit  
DefaultStyledDocument.ElementSpec  
DefaultTextUI  
EditorKit  
ElementIterator  
FieldView  
GapContent  
IconView  
JTextComponent  
JTextComponent.KeyBinding  
LabelView  
LabelView2D  
LayeredHighlighter  
LayeredHighlighter.LayerPainter  
ParagraphView  
PasswordView  
PlainDocument  
PlainView  
Position.Bias  
Segment  
SimpleAttributeSet  
StringContent

StyleConstants  
StyleConstants.CharacterConstants  
StyleConstants.ColorConstants  
StyleConstants.FontConstants  
StyleConstants.ParagraphConstants  
StyleContext  
StyledEditorKit  
StyledEditorKit.AlignmentAction  
StyledEditorKit.BoldAction  
StyledEditorKit.FontFamilyAction  
StyledEditorKit.FontSizeAction  
StyledEditorKit.ForegroundAction  
StyledEditorKit.ItalicAction  
StyledEditorKit.StyledTextAction  
StyledEditorKit.UnderlineAction  
TableView  
TabSet  
TabStop  
TextAction  
Utilities  
View  
WrappedPlainView

### Exceptions

BadLocationException  
ChangedCharSetException

## 程序包名： javax.swing.text.html

此程序包包含创建 HTML 文本编辑器的类。

### Classes

BlockView	HTMLEditorKit.LinkController
CSS	HTMLEditorKit.Parser
CSS.Attribute	HTMLEditorKit.ParserCallback
FormView	HTMLFrameHyperlinkEvent
HTML	HTMLWriter
HTML.Attribute	InlineView
HTML.Tag	istView
HTML.UnknownTag	initalHTMLWriter
HTMLDocument	ObjectView
HTMLDocument.Iterator	Option
HTMLEditorKit	ParagraphView
HTMLEditorKit.HTMLFactory	StyleSheet
HTMLEditorKit.HTMLTextAction	StyleSheet.BoxPainter

HMLEditorKit.InsertHMLTextAction    StyleSheet.ListPainter

### 程序包名: javax.swing.tree

此程序包提供处理 javax.swing 包中 JTree 的接口和类。

#### Interface

MutableTreeNode	TreeModel
RowMapper	TreeNode
TreeCellEditor	TreeSelectionModel
TreeCellRenderer	

#### Classes

utCache	DefaultTreeModel
AbstractLayoutCache.NodeDimensions	DefaultTreeSelectionModel
DefaultMutableTreeNode	FixedHeightLayoutCache
DefaultTreeCellEditor	TreePath
DefaultTreeCellRenderer	VariableHeightLayoutCache

#### Exception

ExpandVetoException

### 程序包名: javax.swing.undo

此程序包支持在应用程序（比如文本编辑器）中的撤消/恢复特性。

#### Interface

StateEditable	
UndoableEdit	

#### Classes

AbstractUndoableEdit	UndoableEditSupport
CompoundEdit	UndoManager
StateEdit	

#### Exception

CannotRedoException	
CannotUndoException	

## 类和接口的详细描述

这一节给出下列程序包的详细描述：

javax.swing	
javax.swing.border	
javax.swing.colorchooser	
javax.swing.event	
javax.swing.filechooser	

---

```
javax.swing.plaf
javax.swing.table
javax.swing.text
javax.swing.text.html
javax.swing.tree
javax.swing.undo
```

## **javax.swing**

### Action 接口

```
public abstract interface Action
    extends ActionListener
```

此接口允许简化由多个 GUI 控件执行的公共功能的实现，这些控件有：工具条上的按钮、常规菜单或弹出式菜单上的菜单项等等。

#### Fields

static String	DEFAULT
static String	LONG_DESCRIPTION
static String	NAME
static String	SHORT_DESCRIPTION
static String	SMALL_ICON

#### Methods

void	addPropertyChangeListener (PropertyChangeListener listener)
Object	getValue (String key)
boolean	isEnabled ()
void	putValue (String key, Object value)
void	removePropertyChangeListener (PropertyChangeListener listener)
void	setEnabled (boolean b)

### BoundedRangeModel 接口

```
public abstract interface BoundedRangeModel
```

此接口定义了如滑动块、滚动条、进度条等组件的数据模式，这些组件的最大值及最小值（范围）已经被界定。

#### Methods

void	addChangeListener (ChangeListener x)
int	getExtent ()
int	getMaximum ()
int	getMinimum ()

```

int           getValue ()
boolean      getValueIsAdjusting ()
void         removeChangeListener (ChangeListener x)
void         setExtent (int newExtent)
void         setMaximum (int newMaximum)
void         setMinimum (int newMinimum)
void         setRangeProperties (int value, int extent, int
                                min, int max, boolean adjusting)
void         setValue (int newValue)
void         setValueIsAdjusting (boolean b)

```

## ButtonModel 接口

```

public abstract interface ButtonModel
    extends ItemSelectable

```

此接口定义了处理 Swing 按钮各种状态的设计级模式。

### Methods

```

void         addActionListener (ActionListener l)
void         addChangeListener (ChangeListener l)
void         addItemListener (ItemListener l)
String       getActionCommand ()
int          getMnemonic ()
boolean      isArmed ()
boolean      isEnabled ()
boolean      isPressed ()
boolean      isRollover ()
boolean      isSelected ()
void         removeActionListener (ActionListener l)
void         removeChangeListener (ChangeListener l)
void         removeItemListener (ItemListener l)
void         setActionCommand (String s)
void         setArmed (boolean b)
void         setEnabled (boolean b)
void         setGroup (ButtonGroup group)
void         setMnemonic (int akey)
void         setPressed (boolean b)
void         setRollover (boolean b)
void         setSelected (boolean b)

```

## CellEditor 接口

```

public abstract interface CellEditor

```

这是一个设计级接口，它定义了单元编辑器的功能。这些单元编辑器由诸如列表、表等

允许编辑的组件单元使用。

#### Methods

void	addCellEditorListener (CellEditorListener l)
void	cancelCellEditing ()
Object	getCellEditorValue ()
boolean	isCellEditable (EventObject anEvent)
void	removeCellEditorListener (CellEditorListener l)
boolean	shouldSelectCell (EventObject anEvent)
boolean	stopCellEditing ()

### ComboBoxEditor 接口

```
public abstract interface ComboBoxEditor
```

此接口定义了专门由 Swing 组合框使用的编辑器的方法。

#### Methods

void	addActionListener (ActionListener l)
Component	getEditorComponent ()
Object	getItem ()
void	removeActionListener (ActionListener l)
void	selectAll ()
void	setItem (Object anObject)

### ComboBoxModel 接口

```
public abstract interface ComboBoxModel  
extends ListModel
```

此接口声明了一些抽象方法，用来对组合框中选定的数据项进行赋值和检索。除了其使用的列表模式以外，组合框也需要这个功能。

#### Methods

Object	getSelectedItem ()
void	setSelectedItem (Object anItem)

### DesktopManager 接口

```
public abstract interface DesktopManager
```

此接口定义了桌面管理器的设计级功能。桌面管理器处理外观属性，包括对桌面窗格和内部框架的关闭、改变大小等等。

#### Methods

void	activateFrame (JInternalFrame f)
void	beginDraggingFrame (JComponent f)
void	beginResizingFrame (JComponent f, int direction)
void	closeFrame (JInternalFrame f)

```

void deactivateFrame (JInternalFrame f)
void deiconifyFrame (JInternalFrame f)
void dragFrame (JComponent f, int newX, int newY)
void endDraggingFrame (JComponent f)
void endResizingFrame (JComponent f)
void iconifyFrame (JInternalFrame f)
void maximizeFrame (JInternalFrame f)
void minimizeFrame (JInternalFrame f)
void openFrame (JInternalFrame f)
void resizeFrame (JComponent f, int newX, int newY, int
newWidth, int newHeight)
void setBoundsForFrame (JComponent f, int newX, int newY, int
newWidth, int newHeight)

```

## Icon 接口

```
public abstract interface Icon
```

此接口为图标定义了方法，这种图标一般是用来装饰 Swing 组件的。

### Methods

```

int getIconHeight ()
int getIconWidth ()
void paintIcon (Component c, Graphics g, int x, int y)

```

## JComboBox.KeySelectionManager 接口

```
public abstract static interface JComboBox.KeySelectionManager
```

此接口定义了一个由组合框使用的密钥选择管理器。此接口包含一个检索组合框项的行索引的方法，如果应用了密钥和组合框模式。

### Method

```
int selectionForKey (char aKey, ComboBoxModel aModel)
```

## ListCellRenderer 接口

```
public abstract interface ListCellRenderer
```

此接口定义了一种获得组件的方法，这个组件绘制带有指定图标和标签的 Swing 列表单元。这个组件的功能类似于橡皮图章。

### Method

```
Component getListCellRendererComponent (JList list, Object value,
int index, boolean isSelected, boolean cellHasFocus)
```

## ListModel 接口

```
public abstract interface ListModel
```

此接口为显示在 Swing 列表中的数据定义了模式。

void	addListDataListener (ListDataListener l)
Object	getElementAt (int index)
int	getSize ()
void	removeListDataListener (ListDataListener l)

### ListSelectionModel 接口

`public abstract interface ListSelectionModel`

此接口为列表组件存储选择数据定义了模式，如选择的类型、选择有关的索引等等。

#### Fields

static int	MULTIPLE_INTERVAL_SELECTION
static int	SINGLE_INTERVAL_SELECTION
static int	SINGLE_SELECTION

#### Methods

void	addListSelectionListener (ListSelectionListener x)
void	addSelectionInterval (int index0, int index1)
void	clearSelection ()
int	getAnchorSelectionIndex ()
int	getLeadSelectionIndex ()
int	getMaxSelectionIndex ()
int	getMinSelectionIndex ()
int	getSelectionMode ()
boolean	getValueIsAdjusting ()
void	insertIndexInterval (int index, int length, boolean before)
boolean	isSelectedIndex (int index)
boolean	isSelectionEmpty ()
void	removeIndexInterval (int index0, int index1)
void	removeListSelectionListener (ListSelectionListener x)
void	removeSelectionInterval (int index0, int index1)
void	setAnchorSelectionIndex (int index)
void	setLeadSelectionIndex (int index)
void	setSelectionInterval (int index0, int index1)
void	setSelectionMode (int selectionMode)
void	setValueIsAdjusting (boolean valueIsAdjusting)

### MenuItem 接口

`public abstract interface MenuItem`

此接口是用来在菜单层次结构中移动菜单并选择菜单项。加在菜单上的组件或菜单项必须实现这个接口。

**Methods**

```

Component      getComponent ()
MenuElement [] getSubElements ()
void          menuSelectionChanged (boolean isIncluded)
void          processKeyEvent (KeyEvent event, MenuElement []
path, MenuSelectionManager manager)
void processMouseEvent (MouseEvent event,
MenuElement [] path, MenuSelectionManager
manager)

```

**MutableComboBoxModel 接口**

```

public abstract interface MutableComboBoxModel
    extends ComboBoxModel

```

此接口定义了以后将要改变的组合框模式。此接口是常规组合框模式的简单扩展。

**Methods**

```

void      addElement (Object obj)
void      insertElementAt (Object obj, int index)
void      removeElement (Object obj)
void      removeElementAt (int index)

```

**Renderer 接口**

```

public abstract interface Renderer

```

此接口定义了一个绘制程序，它可以在 Swing 组件上绘制数值或标语。

**Methods**

```

Component      getComponent ()
void          setValue (Object aValue, boolean isSelected)

```

**RootPaneContainer 接口**

```

public abstract interface RootPaneContainer

```

此接口由这样一些组件来实现，它们必需为内容窗格、玻璃窗格、分层窗格和根窗格提供访问方法。你会看到实现这个接口的容器如： JApplet、 JFrame、 JInternalFrame、 JDialog 和 JWindow。

**Methods**

```

Container      getContentPane ()
Component      getGlassPane ()
JLayeredPane   getLayeredPane ()
JRootPane     getRootPane ()
void          setContentPane (Container contentPane)
void          setGlassPane (Component glassPane)

```

```
void setLayeredPane (JLayeredPane layeredPane)
```

## Scrollable 接口

```
public abstract interface Scrollable
```

此接口定义了检索各种可滚动属性的数据的方法，如视口的大小以及容器或支持滚动的组件的滚动增量。

### Methods

Dimension	getPreferredScrollableViewportSize ()
int	getScrollableBlockIncrement (Rectangle visibleRect, int orientation, int direction)
boolean	getScrollableTracksViewportHeight ()
boolean	getScrollableTracksViewportWidth ()
int	getScrollableUnitIncrement (Rectangle visibleRect, int orientation, int direction)

## ScrollPaneConstants 接口

```
public abstract interface ScrollPaneConstants
```

此接口定义了一个通常由支持滚动的组件使用的常量。

### Fields

static String	COLUMN_HEADER
static String	HORIZONTAL_SCROLLBAR
static int	HORIZONTAL_SCROLLBAR_ALWAYS
static int	HORIZONTAL_SCROLLBAR_AS_NEEDED
static int	HORIZONTAL_SCROLLBAR_NEVER
static String	HORIZONTAL_SCROLLBAR_POLICY
static String	LOWER_LEFT_CORNER
static String	LOWER_RIGHT_CORNER
static String	ROW_HEADER
static String	UPPER_LEFT_CORNER
static String	UPPER_RIGHT_CORNER
static String	VERTICAL_SCROLLBAR
static int	VERTICAL_SCROLLBAR_ALWAYS
static int	VERTICAL_SCROLLBAR_AS_NEEDED
static int	VERTICAL_SCROLLBAR_NEVER
static String	VERTICAL_SCROLLBAR_POLICY
static String	VIEWPORT

## SingleSelectionModel 接口

```
public abstract interface SingleSelectionModel
```

此接口定义了从 Swing 组件中选择一个项目的选模式。

**Methods**

void	addChangeListener (ChangeListener listener)
void	clearSelection ()
int	getSelectedIndex ()
boolean	isSelected ()
void	removeChangeListener (ChangeListener listener)
void	setSelectedIndex (int index)

**SwingConstants 接口**

```
Public abstract interface SwingConstants
```

此接口定义了通常需要在显示区域内定位和排列 Swing 组件的一列常量。

**Fields**

static int	BOTTOM
static int	CENTER
static int	EAST
static int	HORIZONTAL
static int	LEADING
static int	LEFT
static int	NORTH
static int	NORTH_EAST
static int	NORTH_WEST
static int	RIGHT
static int	SOUTH
static int	SOUTH_EAST
static int	SOUTH_WEST
static int	TOP
static int	TRAILING
static int	VERTICAL
static int	WEST

**UIDefaults.ActiveValue 接口**

```
public abstract static interface UIDefaults.ActiveValue
```

此类表示一个活动值 (active value)，该活动值允许存储缺省的表格。每当访问该缺省表格时，就会用相应的“get”方法构造这个缺省的表格。

**Method**

Object	createValue (UIDefaults table)
--------	--------------------------------

**UIDefaults.LazyValue 接口**

```
public abstract static interface UIDefaults.LazyValue
```

此类表示一个惰值 (lazy value)，该惰值也允许存储一个缺省的表格。只有在第一次访问时，才会构造这个表。惰值对那些难以构造而且很少被检索的缺省值来说是很有用的。

#### Method

object	createValue (UIDefaults table)
--------	--------------------------------

### WindowConstants 接口

```
public abstract interface WindowConstants
```

此接口定义了一些常量，用来指定在关闭窗口时需要做什么。典型的情况是，在关闭 Swing 框架或对话框时需要使用这些常量。

#### Fields

static int	DISPOSE_ON_CLOSE
static int	DO_NOTHING_ON_CLOSE
static int	HIDE_ON_CLOSE

### AbstractAction 类

```
java.lang.Object
|
+
-- javax.swing.AbstractAction
public abstract class AbstractAction
    extends Object
    implements Action, Cloneable, Serializable
```

这个类只实现了 Action 接口的部分功能。其访问方法由缺省代码定义。只需要给这个抽象类子类中的 actionPerformed () 方法提供代码。

#### Fields

protected SwingPropertyChangeSupport	changeSupport
protected boolean	enabled

#### Constructors

AbstractAction ()	
AbstractAction (String name)	
AbstractAction (String name, Icon icon)	

#### Methods

void	addPropertyChangeListener (PropertyChangeListener listener)
protected Object	clone () //To clone an abstract action
protected void	firePropertyChange (String propertyName, Object oldValue, Object newValue)
Object	getValue (String key)
boolean	isEnabled ()

```

void          putValue (String key, Object newValue)
void          removePropertyChangeListener
              (PropertyChangeListener listener)
void          setEnabled (boolean newValue)

```

## AbstractButton 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton

public abstract class AbstractButton
    extends JComponent
    implements ItemSelectable, SwingConstants

```

这个类为存储按钮、菜单项和菜单的大量功能提供了公用场所。注意，Swing 的 JButton、JToggleButton、JCheckBox、JRadioButton、 JMenuItem、JMenu、JCheckBoxMenuItem 和 JRadioButtonMenuItem 都是从这个类扩展得到的。

### Inner Classes

protected class	AbstractButton.AccessibleAbstractButton
protected class	AbstractButton.ButtonChangeListener

### Fields

protected ActionListener	actionListener
static String	BORDER_PAINTED_CHANGED_PROPERTY
protected ChangeEvent	changeEvent
protected ChangeListener	changeListener
static String	CONTENT_AREA_FILLED_CHANGED_PROPERTY
static String	DISABLED_ICON_CHANGED_PROPERTY
static String	DISABLED_SELECTED_ICON_CHANGED_PROPERTY
static String	FOCUS_PAINTED_CHANGED_PROPERTY
static String	HORIZONTAL_ALIGNMENT_CHANGED_PROPERTY
static String HORIZONTAL_TEXT_POSITION_CHANGED_PROPERTY	
static String	ICON_CHANGED_PROPERTY
protected ItemListener	itemListener
static String	MARGIN_CHANGED_PROPERTY
static String	MNEMONIC_CHANGED_PROPERTY
protected ButtonModel	model
static String	MODEL_CHANGED_PROPERTY
static String	PRESSED_ICON_CHANGED_PROPERTY
static String	ROLLOVER_ENABLED_CHANGED_PROPERTY

static String	ROLLOVER_ICON_CHANGED_PROPERTY
static String	ROLLOVER_SELECTED_ICON_CHANGED_PROPERTY
static String	SELECTED_ICON_CHANGED_PROPERTY
static String	TEXT_CHANGED_PROPERTY
static String	VERTICAL_ALIGNMENT_CHANGED_PROPERTY
static String	VERTICAL_TEXT_POSITION_CHANGED_PROPERTY

**Constructor**

AbstractButton ()

**Methods**

void	addActionListener (ActionListener l)
void	addChangeListener (ChangeListener l)
void	addItemListener (ItemListener l)
protected int	checkHorizontalKey (int key, String exception)
protected int	checkVerticalKey (int key, String exception)
protected ActionListener	createActionListener ()
protected ChangeListener	createChangeListener ()
protected ItemListener	createItemListener ()
void	doClick ()
void	doClick (int pressTime)
protected void	fireActionPerformed (ActionEvent event)
protected void	fireItemStateChanged (ItemEvent event)
protected void	fireStateChanged ()
String	getActionCommand ()
Icon	getDisabledIcon ()
Icon	getDisabledSelectedIcon ()
int	getHorizontalAlignment ()
int	getHorizontalTextPosition ()
Icon	getIcon ()
String	getLabel ()
// Deprecated. - Use the method	
Insets	getText ()
int	getMargin ()
ButtonModel	getMnemonic ()
Icon	getModel ()
Icon	getPressedIcon ()
Icon	getRolloverIcon ()
Icon	getRolloverSelectedIcon ()
Icon	getSelectedIcon ()
Object []	getSelectedObjects ()
String	getText ()

```
ButtonUI           getUI ()
int               getVerticalAlignment ()
int               getVerticalTextPosition ()
protected void    init (String text, Icon icon)
boolean          isBorderPainted ()
boolean          isContentAreaFilled ()
boolean          isFocusPainted ()
boolean          isRolloverEnabled ()
boolean          isSelected ()
protected void    paramString ()
protected String  paramString () {
    // This method obtains a string that
    // represents the abstract button.
}

void              removeActionListener (ActionListener l)
void              removeChangeListener (ChangeListener l)
void              removeItemListener (ItemListener l)
void              setActionCommand (String
actionCommand)
void              setBorderPainted (boolean b)
void              setContentAreaFilled (boolean b)
void              setDisabledIcon (Icon disabledIcon)
void              setDisabledSelectedIcon (Icon
disabledSelectedIcon)
void              setEnabled (boolean b)
void              setFocusPainted (boolean b)
void              setHorizontalAlignment (int alignment)
void              setHorizontalTextPosition (int textPosition)
void              setIcon (Icon defaultIcon)
void              setLabel (String label)
void              // Deprecated. - Use the method setText (text)
setMargin (Insets m)
void              setMnemonic (char mnemonic)
void              setMnemonic (int mnemonic)
void              setModel (ButtonModel newModel)
void              setPressedIcon (Icon pressedIcon)
void              setRolloverEnabled (boolean b)
void              setRolloverIcon (Icon rolloverIcon)
void              setRolloverSelectedIcon (Icon
rolloverSelectedIcon)
void              setSelected (boolean b)
void              setSelectedIcon (Icon selectedIcon)
void              setText (String text)
void              setUI (ButtonUI ui)
void              setVerticalAlignment (int alignment)
void              setVerticalTextPosition (int textPosition)
```

```
void updateUI ()
```

## AbstractListModel 类

```
java.lang.Object
|
+-- javax.swing.AbstractListModel
public abstract class AbstractListModel
    extends Object
    implements ListModel, Serializable
```

这个类提供了接口 ListModel 的抽象实现。表示定制数据模式的类可以扩展这个类并重载它的方法。

### Field

```
protected EventListenerList listenerList
```

### Constructor

```
AbstractListModel ()
```

### Methods

void	addListDataListener (ListDataListener l)
protected void	fireContentsChanged (Object source, int index0, int index1)
protected void	fireIntervalAdded (Object source, int index0, int index1)
protected void	fireIntervalRemoved (Object source, int index0, int index1)
void	removeListDataListener (ListDataListener l)

## BorderFactory 类

```
java.lang.Object
|
+-- javax.swing.BorderFactory
public class BorderFactory
    extends Object
```

这个类表示提供了 Swing 中所引入的大量标准边界的类工厂。通过调用这个类的特定静态方法，可以得到感兴趣的边界对象，而不必实例化边界类。

### Methods

static Border	createBevelBorder (int type)
static Border	createBevelBorder (int type, Color highlight, Color shadow)
static Border	createBevelBorder (int type, Color highlightOuter, Color highlightInner, Color shadowOuter, Color shadowInner)
static CompoundBorder	createCompoundBorder ()

```

static CompoundBorder    createCompoundBorder (Border outsideBorder,
                                              Border insideBorder)
static Border           createEmptyBorder ()
static Border           createEmptyBorder (int top, int left, int bottom,
                                             int right)
static Border           createEtchedBorder ()
static Border           createEtchedBorder (Color highlight,
                                              Color shadow)
static Border           createLineBorder (Color color)
static Border           createLineBorder (Color color, int thickness)
static Border           createLoweredBevelBorder ()
static MatteBorder      createMatteBorder (int top, int left, int bottom,
                                              int right, Color color)
static MatteBorder      createMatteBorder (int top, int left, int bottom,
                                              int right, Icon titleIcon)
static Border           createRaisedBevelBorder ()
static TitledBorder     createTitledBorder (Border border)
static TitledBorder     createTitledBorder (Border border, String title)
static TitledBorder     createTitledBorder (Border border, String title,
                                              int titleJustification, int titlePosition)
static TitledBorder     createTitledBorder (Border border, String title,
                                              int titleJustification, int titlePosition,
                                              Font titleFont)
static TitledBorder     createTitledBorder (Border border, String title,
                                              int titleJustification, int titlePosition,
                                              Font titleFont, Color titleColor)
static TitledBorder     createTitledBorder (String title)

```

## Box 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.Box

public class Box
    extends Container
    implements Accessible

```

这个类表示了一个轻量级的容器，它已经被预先赋以了方框布局类型。可以直接创建这种类的对象，并按方框布局的规则添加组件。

### Inner Classes

protected class	Box.AccessibleBox
static class	Box.Filler // An invisible and light-weight

component that helps to fill in spacing between components, or container and components.

**Field**

```
protected AccessibleContext accessibleContext
```

**Constructor**

```
Box (int axis)
```

**Methods**

static Component	createGlue ()
static Box	createHorizontalBox ()
static Component	createHorizontalGlue ()
static Component	createHorizontalStrut (int width)
static Component	createRigidArea (Dimension d)
static Box	createVerticalBox ()
static Component	createVerticalGlue ()
static Component	createVerticalStrut (int height)
AccessibleContext	getAccessibleContext ()
void	setLayout (LayoutManager l)

**Box.Filler 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- javax.swing.Box.Filler
public static class Box.Filler
    extends Component
    implements Accessible
```

这是一个不可见的轻量级组件，它帮助填充按方框布局排列的各组件间的空间。

**Inner Class**

```
protected class Box.Filler.AccessibleBoxFiller
```

**Field**

```
protected AccessibleContext accessibleContext
```

**Constructor**

```
Box.Filler (Dimension min, Dimension pref, Dimension max)
```

**Methods**

void	changeShape (Dimension min, Dimension pref, Dimension max)
AccessibleContext	getAccessibleContext ()
Dimension	getMaximumSize ()
Dimension	getMinimumSize ()

Dimension                   getPreferredSize ()

## BoxLayout 类

```
java.lang.Object
|
+-- javax.swing.BoxLayout
public class BoxLayout
    extends Object
    implements LayoutManager2, Serializable
```

这个类表示一个布局管理器。可以用这个布局选择代替 AWT 中引入的网格袋布局。可以按照为布局所指定的坐标轴垂直或水平地摆放组件。

### Fields

static int	X_AXIS // Specifies that the components be placed from left to right
static int	Y_AXIS // Specifies that the components be placed from top to bottom

### Constructor

BoxLayout (Container target, int axis)

### Methods

void	addLayoutComponent (Component comp, Object constraints)
void	addLayoutComponent (String name, Component comp)
float	getLayoutAlignmentX (Container target)
float	getLayoutAlignmentY (Container target)
void	invalidateLayout (Container target)
void	layoutContainer (Container target)
Dimension	maximumLayoutSize (Container target)
Dimension	minimumLayoutSize (Container target)
Dimension	preferredLayoutSize (Container target)
void	removeLayoutComponent (Component comp)

## ButtonGroup 类

```
java.lang.Object
|
+-- javax.swing.ButtonGroup
public class ButtonGroup
    extends Object
    implements Serializable
```

这个类表示创建一组按钮的机制，这些按钮每次只能选择一个（它们是相互排斥的）。必须使用这个类才能形成一组 Swing 单选按钮，它们的选择是互斥的。

### Field

protected Vector buttons

**Constructor**

```
ButtonGroup ()
```

**Methods**

void	add (AbstractButton b)
Enumeration	getElements ()
ButtonModel	getSelection ()
boolean	isSelected (ButtonModel m)
void	remove (AbstractButton b)
void	setSelected (ButtonModel m, boolean b)

**CellRendererPane 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.CellRendererPane
public class CellRendererPane
    extends Container
    implements Accessible
```

这个类有助于在单元绘制程序和使用该绘制程序的组件之间建立联系。这个类的对象在防止对 repaint () 方法和 invalidate () 方法的无谓调用方面是很有用的。这个类可以用于 Swing 表、树和列表。

**Inner Class**

```
protected class CellRendererPane.AccessibleCellRendererPane
```

**Field**

```
protected AccessibleContext accessibleContext
```

**Constructor**

```
CellRendererPane ()
```

**Methods**

protected void	addImpl (Component x, Object constraints, int index)
AccessibleContext	getAccessibleContext ()
void	invalidate ()
void	paint (Graphics g)
void	paintComponent (Graphics g, Component c,                     Container p, int x, int y, int w, int h)
void	paintComponent (Graphics g, Component c,                     Container p, int x, int y, int w, int h,                     boolean shouldValidate)
void	paintComponent (Graphics g, Component c,

```

        Container p, Rectangle r)
void update (Graphics g)

```

## DebugGraphics 类

```

java.lang.Object
|
+-- java.awt.Graphics
    |
    +-- javax.swing.DebugGraphics
public class DebugGraphics
    extends Graphics

```

这个类支持图形调试。一旦改变组件的调试选项，就会自动产生这个类的实例。使用 JComponent 中的 setDebugGraphicsOptions () 方法，可以给组件赋予新的选项。

### Fields

static int	BUFFERED_OPTION
static int	FLASH_OPTION
static int	LOG_OPTION
static int	NONE_OPTION

### Constructors

```

DebugGraphics ()
DebugGraphics (Graphics graphics)
DebugGraphics (Graphics graphics, JComponent component)

```

### Methods

void	clearRect (int x, int y, int width, int height)
void	clipRect (int x, int y, int width, int height)
void	copyArea (int x, int y, int width, int height,            int destX, int destY)
Graphics	create ()
Graphics	create (int x, int y, int width, int height)
void	dispose ()
void	draw3DRect (int x, int y, int width, int              height, boolean raised)
void	drawArc (int x, int y, int width, int height,           int startAngle, int arcAngle)
void	drawBytes (byte [] data, int offset, int             length, int x, int y)
void	drawChars (char [] data, int offset, int             length, int x, int y)
boolean	drawImage (Image img, int x, int y, Color             bgcolor, ImageObserver observer)
boolean	drawImage (Image img, int x, int y,             ImageObserver observer)
boolean	drawImage (Image img, int x, int y, int width, int height,

```
Color bgcolor, ImageObserver observer)
boolean drawImage (Image img, int x, int y, int width,
int height, ImageObserver observer)
boolean drawImage (Image img, int dx1, int dy1, int
dx2, int dy2, int sx1, int sy1, int sx2, int
sy2, Color bgcolor, ImageObserver observer)
boolean drawImage (Image img, int dx1, int dy1, int
dx2, int dy2, int sx1, int sy1, int sx2, int
sy2, ImageObserver observer)
void drawLine (int x1, int y1, int x2, int y2)
void drawOval (int x, int y, int width, int height)
void drawPolygon (int [] xPoints, int [] yPoints, int nPoints)
void drawPolyline (int [] xPoints, int [] yPoints, int nPoints)
void drawRect (int x, int y, int width, int height)
void drawRoundRect (int x, int y, int width, int
height, int arcWidth, int arcHeight)
void drawString (String aString, int x, int y)
void fill3DRect (int x, int y, int width, int
height, boolean raised)
void fillArc (int x, int y, int width, int height,
int startAngle, int arcAngle)
void fillOval (int x, int y, int width, int height)
void fillPolygon (int [] xPoints, int [] yPoints, int nPoints)
void fillRect (int x, int y, int width, int height)
void fillRoundRect (int x, int y, int width, int
height, int arcWidth, int arcHeight)
static Color flashColor ()
static int flashCount ()
static int flashTime ()
Shape getClip ()
Rectangle getClipBounds ()
Color getColor ()
int getDebugOptions ()
Font getFont ()
FontMetrics getFontMetrics ()
FontMetrics getFontMetrics (Font f)
boolean isDrawingBuffer ()
static PrintStream logStream ()
void setClip (int x, int y, int width, int height)
void setClip (Shape clip)
void setColor (Color aColor)
void setDebugOptions (int options)
static void setFlashColor (Color flashColor)
static void setFlashCount (int flashCount)
static void setFlashTime (int flashTime)
void setFont (Font aFont)
static void setLogStream (PrintStream stream)
```

```

void           setPaintMode ()
void           setXORMode (Color aColor)
void           translate (int x, int y)

```

## DefaultBoundedRangeModel 类

```

java.lang.Object
|
+-- javax.swing.DefaultBoundedRangeModel
public class DefaultBoundedRangeModel
    extends Object
    implements BoundedRangeModel, Serializable

```

这个类表示一个界限范围模式，它一般被组件用来显示界限值（如最大值及最小值）的范围。这个类是接口 BoundedRangeModel 的缺省实现。

### Fields

```

protected ChangeEvent      changeEvent
protected EventListenerList listenerList
Constructors
DefaultBoundedRangeModel ()
DefaultBoundedRangeModel (int value, int extent, int min, int max)

```

### Methods

```

void           addChangeListener (ChangeListener l)
protected void fireStateChanged ()
int            getExtent ()
int            getMaximum ()
int            getMinimum ()
int            getValue ()
boolean        getValueIsAdjusting ()
void           removeChangeListener (ChangeListener l)
void           setExtent (int n)
void           setMaximum (int n)
void           setMinimum (int n)
void           setRangeProperties (int newValue, int
                                   newExtent, int newMin, int newMax, boolean
                                   adjusting)
void           setValue (int n)
void           setValueIsAdjusting (boolean b)
String         toString ()

```

## DefaultButtonModel 类

```

java.lang.Object
|
+-- javax.swing.DefaultButtonModel
public class DefaultButtonModel
    extends Object
    implements ButtonModel, Serializable

```

这个类表示 Swing 按钮状态的缺省数据模式。以编程方式操作按钮的给定状态时，需要调用这个类的方法。

### Fields

protected String	actionCommand
static int	ARMED
protected ChangeEvent	changeEvent
static int	ENABLED
protected ButtonGroup	group
protected EventListenerList	listenerList
protected int	mnemonic
static int	PRESSED
static int	ROLLOVER
static int	SELECTED
protected int	stateMask

### Constructor

DefaultButtonModel ()

### Methods

void	addActionListener (ActionListener l)
void	addChangeListener (ChangeListener l)
void	addItemListener (ItemListener l)
protected void	fireActionPerformed (ActionEvent e)
protected void	fireItemStateChanged (ItemEvent e)
protected void	fireStateChanged ()
String	getActionCommand ()
int	getMnemonic ()
Object []	getSelectedObjects ()
boolean	isArmed ()
boolean	isEnabled ()
boolean	isPressed ()
boolean	isRollover ()
boolean	isSelected ()
void	removeActionListener (ActionListener l)
void	removeChangeListener (ChangeListener l)
void	removeItemListener (ItemListener l)
void	setActionCommand (String actionCommand)
void	setArmed (boolean b)
void	setEnabled (boolean b)
void	setGroup (ButtonGroup group)
void	setMnemonic (int key)
void	setPressed (boolean b)
void	setRollover (boolean b)

```
void setSelected (boolean b)
```

## DefaultCellEditor 类

```
java.lang.Object
|
+-- javax.swing.DefaultCellEditor
public class DefaultCellEditor
    extends Object
    implements TableCellEditor, TreeCellEditor, Serializable
```

这个类表示 Swing 表格和树的缺省单元编辑器。

### Inner Class

```
protected class DefaultCellEditor.EditorDelegate
```

### Fields

protected ChangeEvent	changeEvent
protected int	clickCountToStart
protected DefaultCellEditor.EditorDelegate	delegate
protected JComponent	editorComponent
protected EventListenerList	listenerList

### Constructors

```
DefaultCellEditor (JCheckBox x)
DefaultCellEditor (JComboBox x)
DefaultCellEditor (JTextField x)
```

### Methods

void	addCellEditorListener (CellEditorListener l)
void	cancelCellEditing ()
protected void	fireEditingCanceled ()
protected void	fireEditingStopped ()
Object	getCellEditorValue ()
int	getClickCountToStart ()
Component	getComponent ()
Component	getTableCellEditorComponent (JTable table, Object value, boolean isSelected, int row, int column)
Component	getTreeCellEditorComponent (JTree tree, Object value, boolean isSelected, boolean expanded, boolean leaf, int row)
boolean	isCellEditable (EventObject anEvent)
void	removeCellEditorListener (CellEditorListener l)
void	setClickCountToStart (int count)
boolean	shouldSelectCell (EventObject anEvent)
boolean	stopCellEditing ()

## DefaultComboBoxModel 类

```
java.lang.Object
|
+-- javax.swing.AbstractListModel
|
+-- javax.swing.DefaultComboBoxModel
public class DefaultComboBoxModel
    extends AbstractListModel
    implements MutableComboBoxModel, Serializable
```

这个类表示 Swing 组合框的缺省数据模式。这个类基本上是接口 `MutableComboBoxModel` 的标准实现。

### Constructors

```
DefaultComboBoxModel ()
DefaultComboBoxModel (Object [] items)
DefaultComboBoxModel (Vector v)
```

### Methods

void	<code>addElement (Object anObject)</code>
Object	<code>getElementAt (int index)</code>
int	<code>getIndexOf (Object anObject)</code>
Object	<code>getSelectedItem ()</code>
int	<code>getSize ()</code>
void	<code>insertElementAt (Object anObject, int index)</code>
void	<code>removeAllElements ()</code>
void	<code>removeElement (Object anObject)</code>
void	<code>removeElementAt (int index)</code>
void	<code>setSelectedItem (Object anObject)</code>

## DefaultDesktopManager 类

```
java.lang.Object
|
+-- javax.swing.DefaultDesktopManager
public class DefaultDesktopManager
    extends Object
    implements DesktopManager, Serializable
```

这个类表示接口 `DesktopManager` 的缺省实现。该缺省实现支持管理其父容器中 Swing 内部框架的基本功能（请参看关于方法的描述）。

### Constructor

```
DefaultDesktopManager ()
```

### Methods

void	<code>activateFrame (JInternalFrame f)</code>
void	<code>beginDraggingFrame (JComponent f)</code>

```

void beginResizingFrame (JComponent f, int direction)
void closeFrame (JInternalFrame f)
void deactivateFrame (JInternalFrame f)
void deiconifyFrame (JInternalFrame f)
void dragFrame (JComponent f, int newX, int newY)
//Calls setBoundsForFrame () with the new values.
void endDraggingFrame (JComponent f)
void endResizingFrame (JComponent f)
protected Rectangle getBoundsForIconOf (JInternalFrame f)
protected Rectangle getPreviousBounds (JInternalFrame f)
void iconifyFrame (JInternalFrame f)
void maximizeFrame (JInternalFrame f)
void minimizeFrame (JInternalFrame f)
void openFrame (JInternalFrame f)
protected void removeIconFor (JInternalFrame f)
void resizeFrame (JComponent f, int newX, int
newY, int newWidth, int newHeight)
void setBoundsForFrame (JComponent f, int newX,
int newY, int newWidth, int newHeight)
protected void setPreviousBounds (JInternalFrame f, Rectangle r)
protected void setWasIcon (JInternalFrame f, Boolean value)
protected boolean wasIcon (JInternalFrame f)

```

## DefaultFocusManager 类

```

java.lang.Object
+
+ -- javax.swing.FocusManager
|
+ -- javax.swing.DefaultFocusManager
public class DefaultFocusManager
    extends FocusManager

```

这个类表示在 Swing 中缺省使用的焦点管理器。

### Constructor

```
DefaultFocusManager ()
```

### Methods

boolean	compareTabOrder (Component a, Component b)
void	focusNextComponent (Component aComponent)
void	focusPreviousComponent (Component aComponent)
Component	getComponentAfter (Container aContainer, Component aComponent)
Component	getComponentBefore (Container aContainer, Component aComponent)
Component	getFirstComponent (Container aContainer)
Component	getLastComponent (Container aContainer)
void	processKeyEvent (Component focusedComponent, KeyEvent anEvent)

## DefaultListCellRenderer 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel
|
+-- javax.swing.DefaultListCellRenderer

public class DefaultListCellRenderer
    extends JLabel
    implements ListCellRenderer, Serializable

```

这个类表示 Swing 列表的缺省单元绘制程序。单元绘制程序组件是显示图标和文本字符串的 Swing 标签。

### Inner Class

```
static class DefaultListCellRenderer.UIRenderer
```

### Field

```
protected static Border noFocusBorder
```

### Constructor

```
DefaultListCellRenderer ()
```

### Methods

Component	getListCellRendererComponent (JList list, Object value, int index, boolean isSelected, boolean cellHasFocus)
-----------	---

## DefaultListCellRenderer.UIRenderer 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel
|
+-- javax.swing.DefaultListCellRenderer
|
+-- javax.swing.DefaultListCellRenderer.
    UIRenderer

public static class DefaultListCellRenderer.UIRenderer
    extends DefaultListCellRenderer
    implements UIRenderer

```

这个类表示 DefaultListCellRenderer 的 UI 子类，它实现了 UIResource 接口。DefaultListCellRenderer 并不直接实现 UIResource，所以应用程序可以安全地用 DefaultListCellRenderer 子类重载这个 cellRenderer 属性。

### Constructor

```
DefaultListCellRenderer,UIResource ()
```

## DefaultListModel 类

```
java.lang.Object
|
+-- javax.swing.AbstractListModel
    |
    +-- javax.swing.DefaultListModel
public class DefaultListModel
    extends AbstractListModel
```

这个类表示缺省列表模式。这个类基本上就是实用工具 java.util.Vector 的实现。假如显示的列表数据是向量形式的，这个类就很有用。

### Constructor

```
DefaultListModel ()
```

### Methods

void	add (int index, Object element)
void	addElement (Object obj)
int	capacity ()
void	clear ()
boolean	contains (Object elem)
void	copyInto (Object [] anArray)
Object	elementAt (int index)
Enumeration	elements ()
void	ensureCapacity (int minCapacity)
Object	firstElement ()
Object	get (int index)
Object	getElementAt (int index)
int	getSize ()
int	indexOf (Object elem)
int	indexOf (Object elem, int index)
void	insertElementAt (Object obj, int index)
boolean	isEmpty ()
Object	lastElement ()
int	lastIndexOf (Object elem)
int	lastIndexOf (Object elem, int index)
Object	remove (int index)
void	removeAllElements ()
boolean	removeElement (Object obj)
void	removeElementAt (int index)

---

void	removeRange (int fromIndex, int toIndex)
Object	set (int index, Object element)
void	setElementAt (Object obj, int index)
void	setSize (int newSize)
int	size ()
Object []	toArray ()
String	toString ()
void	trimToSize ()

## DefaultListModel 类

```

java.lang.Object
|
+-- javax.swing.DefaultListModel
public class DefaultListModel
    extends Object
    implements ListSelectionModel, Cloneable, Serializable

```

这个类表示由列表组件缺省使用的选中模式。所有与选中相关的功能都存储在这个类中。如果必须修改选中过程的话，则可以使用这个类，并生成修改过了的子类，再用 setSelectionModel () 方法将此子类对象赋予列表组件。

### Fields

protected boolean	leadAnchorNotificationEnabled
protected EventListenerList	listenerList

### Constructor

DefaultListModel ()

### Methods

void	addListSelectionListener (ListSelectionListener l)
void	addSelectionInterval (int index0, int index1)
void	clearSelection ()
Object	clone ()
protected void	fireValueChanged (boolean isAdjusting)
protected void	fireValueChanged (int firstIndex, int lastIndex)
protected void	fireValueChanged (int firstIndex, int lastIndex, boolean isAdjusting)
int	getAnchorSelectionIndex ()
int	getLeadSelectionIndex ()
int	getMaxSelectionIndex ()
int	getMinSelectionIndex ()
int	getSelectionMode ()
boolean	getValueIsAdjusting ()
void	insertIndexInterval (int index, int length, boolean before)
boolean	isLeadAnchorNotificationEnabled ()
boolean	isSelectedIndex (int index)

```

boolean      isEmpty () 
void         removeIndexInterval ( int index0, int index1 )
void         removeListSelectionListener
            (ListSelectionListener l)
void         removeSelectionInterval ( int index0, int index1 )
void         setAnchorSelectionIndex ( int anchorIndex )
void         setLeadAnchorNotificationEnabled ( boolean flag )
void         setLeadSelectionIndex ( int leadIndex )
void         setSelectionInterval ( int index0, int index1 )
void         setSelectionMode ( int selectionMode )
void         setValuesAdjusting ( boolean isAdjusting )
String       toString ()

```

## DefaultSingleSelectionModel 类

```

java.lang.Object
|
+-- javax.swing.DefaultSingleSelectionModel
public class DefaultSingleSelectionModel
    extends Object
    implements SingleSelectionModel, Serializable

```

这个类表示一个选择模式，它缺省的作用是每次只选择一项，例如，在列表组件里。

### Fields

protected ChangeEvent	changeEvent
protected EventListenerList	listenerList

### Constructor

```
DefaultSingleSelectionModel ()
```

### Methods

void	addChangeListener (ChangeListener l)
void	clearSelection ()
protected void	fireStateChanged ()
int	getSelectedIndex ()
boolean	isSelected ()
void	removeChangeListener (ChangeListener l)
void	setSelectedIndex (int index)

## FocusManager 类

```

java.lang.Object
|
+-- javax.swing.FocusManager
public abstract class FocusManager
    extends Object

```

这个类表示 Swing 焦点管理器，它是一个抽象类。Swing 中使用的缺省焦点管理器扩展

了这个类。

### Field

```
static String FOCUS_MANAGER_CLASS_PROPERTY
```

### Constructor

```
FocusManager ()
```

### Methods

static void	disableSwingFocusManager ()
abstract void	focusNextComponent (Component aComponent)
abstract void	focusPreviousComponent (Component aComponent)
static FocusManager	getCurrentManager ()
static boolean	isFocusManagerEnabled ()
abstract void	processKeyEvent (Component focusedComponent, KeyEvent anEvent)
static void	setCurrentManager (FocusManager aFocusManager)

## GrayFilter 类

```
java.lang.Object
|
+-- java.awt.image.ImageFilter
|
+-- java.awt.image.RGBImageFilter
|
+-- javax.swing.GrayFilter
public class GrayFilter
    extends RGBImageFilter
```

这个类表示图像处理过滤器，它使被提交的图像变灰。变灰了的图像一般显示在禁用的组件上。

### Constructor

```
GrayFilter (boolean b, int p)
```

### Methods

static Image	createDisabledImage (Image i)
int	filterRGB (int x, int y, int rgb)

## ImageIcon 类

```
java.lang.Object
|
+-- javax.swing.ImageIcon
public class ImageIcon
    extends Object
    implements Icon, Serializable
```

这个类表示接口 Icon 的特定实现。图标是用 GIF 和 JPEG 源文件画出的。媒体跟踪器

也可以监视图像的装载状态。

### Fields

protected static Component	component
protected static MediaTracker	tracker

### Constructors

```
ImageIcon ()
ImageIcon (byte [] imageData)
ImageIcon (byte [] imageData, String description)
ImageIcon (Image image)
ImageIcon (Image image, String description)
ImageIcon (String filename)
ImageIcon (String filename, String description)
ImageIcon (URL location)
ImageIcon (URL location, String description)
```

### Methods

String	getDescription ()
int	getIconHeight ()
int	getIconWidth ()
Image	getImage ()
int	getImageLoadStatus ()
ImageObserver	getImageObserver ()
protected void	loadImage (Image image)
void	paintIcon (Component c, Graphics g, int x, int y)
void	setDescription (String description)
void	setImage (Image image)
void	setImageObserver (ImageObserver observer)

## JApplet 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- java.awt.Panel
|
+-- java.applet.Applet
|
+-- javax.swing.JApplet

public class JApplet
    extends Applet
    implements Accessible, RootPaneContainer
```

这个类表示 Swing 小程序，它具有 AWT 小程序上扩展了的功能。Swing 小程序支持菜

单条、能够产生深度感的框架层 (Z-序)、允许在已有组件上进行绘制的玻璃窗格，等等。

### Inner Class

protected class	JApplet.AccessibleJApplet
-----------------	---------------------------

### Fields

protected AccessibleContext	accessibleContext
protected JRootPane	rootPane
protected boolean	rootPaneCheckingEnabled

### Constructor

JApplet ()
------------

### Methods

protected void	addImpl (Component comp, Object constraints, int index)
protected JRootPane	createRootPane ()
AccessibleContext	getAccessibleContext ()
Container	getContentPane ()
Component	getGlassPane ()
JMenuBar	getJMenuBar ()
JLayeredPane	getLayeredPane ()
JRootPane	getRootPane ()
protected boolean	isRootPaneCheckingEnabled ()
protected String	paramString ()
protected void	processKeyEvent (KeyEvent e)
void	setContentPane (Container contentPane)
void	setGlassPane (Component glassPane)
void	setJMenuBar (JMenuBar menuBar)
void	setLayeredPane (JLayeredPane layeredPane)
protected void	setLayout (LayoutManager manager)
protected void	setRootPane (JRootPane root)
void	setRootPaneCheckingEnabled (boolean enabled)
	update (Graphics g)

## JButton 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JButton

public class JButton
    extends AbstractButton
    implements Accessible

```

这个类表示常规的“按压式”(push-type)的Swing按钮。这种按钮是轻量级的，而且可以拥有一个图标和指定了位置及方向的文本。

### Inner Class

```
protected class JButton.AccessibleJButton
```

### Constructors

```
JButton()
JButton(Icon icon)
JButton(String text)
JButton(String text, Icon icon)
```

### Methods

AccessibleContext	getAccessibleContext()
String	getUIClassID()
boolean	isDefaultButton()
boolean	isDefaultCapable()
protected String	paramString()
void	setDefaultCapable(boolean defaultCapable)
void	updateUI()

## JCheckBox类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JToggleButton
|
+-- javax.swing.JCheckBox

public class JCheckBox
    extends JToggleButton
    implements Accessible
```

这个类表示Swing中轻量级的复选框。

### Inner Class

```
protected class JCheckBox.AccessibleJCheckBox
```

### Constructors

```
JCheckBox()
JCheckBox(Icon icon)
JCheckBox(Icon icon, boolean selected)
```

```
JCheckBox (String text)
JCheckBox (String text, boolean selected)
JCheckBox (String text, Icon icon)
JCheckBox (String text, Icon icon, boolean selected)
```

**Methods**

AccessibleContext	getAccessibleContext ()
String	getUIClassID ()
protected String	paramString ()
void	updateUI ()

**JCheckBoxMenuItem 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JMenuItem
|
+-- javax.swing.JCheckBoxMenuItem

public class JCheckBoxMenuItem
    extends JMenuItem
    implements SwingConstants, Accessible
```

这个类表示一个复选框菜单项。复选框菜单项在菜单里使用，它的外观类似于复选框。  
当需要选择多重选项时，使用复选框菜单项是合适的。

**Inner Class**

```
protected class JCheckBoxMenuItem.AccessibleJCheckBoxMenuItem
```

**Constructors**

```
JCheckBoxMenuItem ()
JCheckBoxMenuItem (Icon icon)
JCheckBoxMenuItem (String text)
JCheckBoxMenuItem (String text, boolean b)
JCheckBoxMenuItem (String text, Icon icon)
JCheckBoxMenuItem (String text, Icon icon, boolean b)
```

**Methods**

AccessibleContext	getAccessibleContext ()
Object []	getSelectedObjects ()
boolean	getState ()
String	getUIClassID ()

```

protected void           init (String text, Icon icon)
protected String        paramString ()
void                   requestFocus ()
void                   setState (boolean b)
void                   updateUI ()

```

## JColorChooser 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JColorChooser
public class JColorChooser
    extends JComponent
    implements Accessible

```

这个类表示颜色选择器窗格。可以单击标签，在 Swatch、RGB 和 HSB 窗格之间切换，进而选择任何一种颜色。

### Inner Class

```
protected class JColorChooser.AccessibleJColorChooser
```

### Fields

protected AccessibleContext	accessibleContext
static String	CHOOSEN_PANELS_PROPERTY
static String	PREVIEW_PANEL_PROPERTY
static String	SELECTION_MODEL_PROPERTY

### Constructors

```

JColorChooser ()
JColorChooser (Color initialColor)
JColorChooser (ColorSelectionModel model)

```

### Methods

void	addChooserPanel (AbstractColorChooserPanel panel)
static JDialog	createDialog (Component c, String title, boolean modal, JColorChooser chooserPane, ActionListener okListener, ActionListener cancelListener)
AccessibleContext	getAccessibleContext ()
AbstractColorChooserPanel []	getChooserPanels ()
Color	getColor ()

```

JComponent           getPreviewPanel ()
ColorSelectionModel getSelectionModel ()
ColorChooserUI      getUI ()
String              getUIClassID ()
protected String     paramString ()
AbstractColorChooserPanel removeChooserPanel
                           (AbstractColorChooserPanel panel)
void                setChooserPanels
                           (AbstractColorChooserPanel []
panels)
void                setColor (Color color)
void                setColor (int c)
void                setColor (int r, int g, int b)
void                setPreviewPanel (JComponent preview)
void                setSelectionModel
                           (ColorSelectionModel newModel)
void                setUI (ColorChooserUI ui)
static Color        showDialog (Component component,
                           String title, Color initialColor)
void                updateUI ()

```

## JComboBox 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JComboBox

public class JComboBox
    extends JComponent
    implements ItemSelectable, ListDataListener, ActionListener,
    Accessible

```

这个类表示 Swing 组合框。该组合框是一个文本字段和一个下拉列表的组合。可以从这个下拉列表中选择一项，被选择的项显示在文本字段里。

### Inner Classes

protected class	JComboBox.AccessibleJComboBox
static interface	JComboBox.KeySelectionManager

### Fields

protected String	actionCommand
protected ComboBoxModel	dataModel
protected ComboBoxEditor	editor
protected boolean	isEditable

```

protected JComboBox.KeySelectionManager keySelectionManager
protected boolean lightWeightPopupEnabled
protected int maximumRowCount
protected ListCellRenderer renderer
protected Object selectedItemReminder

```

## Constructors

```

JComboBox ()
JComboBox (ComboBoxModel aModel)
JComboBox (Object [] items)
JComboBox (Vector items)

```

## Methods

```

void actionPerformed (ActionEvent e)
void addActionListener (ActionListener l)
void addItem (Object anObject)
void addItemListener
    (ItemListener
     aListener)
void configureEditor
    (ComboBoxEditor
     anEditor, Object
     anItem)
void contentsChanged
    (ListDataEvent e)
protected JComboBox.KeySelectionManager createDefaultKey
    SelectionManager ()
protected void fireActionPerformed ()
protected void fireItemStateChanged
    (ItemEvent e)
AccessibleContext getAccessibleContext ()
String getActionCommand ()
Object getItemAt (int index)
int getItemCount ()
JComboBox.KeySelectionManager getKeySelection
    Manager ()
int getMaximumRowCount ()
ComboBoxModel getModel ()
ListCellRenderer getRenderer ()
int getSelectedIndex ()
Object getSelectedItem ()
Object [] getSelectedObjects ()
ComboBoxUI getUI ()
String getUIClassID ()
void hidePopup ()
void insertItemAt (Object

```

```
protected void           anObject, int index)
void                   installAncestor
void                   Listener()
void                   intervalAdded
                      (ListDataEvent e)
void                   intervalRemoved
                      (ListDataEvent e)
boolean                isEditable()
boolean                isFocusTraversable()
boolean                isLightWeightPopup
                      Enabled()
boolean                isPopupVisible()
protected String        paramString()
void                   processKeyEvent
                      (KeyEvent e)
void                   removeActionListener
                      (ActionListener l)
void                   removeAllItems()
void                   removeItem (Object
                      anObject)
void                   removeItemAt (int
                      anIndex)
void                   removeItemListener
                      (ItemListener
                      aListener)
protected void          selectedItemChanged()
boolean                selectWithKeyChar (char
                      keyChar)
void                   setActionCommand (String
                      aCommand)
void                   setEditable (boolean
                      aFlag)
void                   setEditor
                      (ComboBoxEditor
                      anEditor)
void                   setEnabled (boolean b)
void                   setKeySelectionManager
                      (JComboBox.
                      KeySelectionManager
                      aManager)
void                   setLightWeight
                      PopupEnabled (boolean
                      aFlag)
void                   setMaximumRowCount (int
                      count)
```

```

void                      setModel (ComboBoxModel
                               aModel)
void                      setPopupVisible (boolean v)
void                      setRenderer
                           (ListCellRenderer
                            aRenderer)
void                      setSelectedIndex (int anIndex)
void                      setSelectedItem (Object anObject)
void                      setUI (ComboBoxUI ui)
void                      showPopup ()
void                      updateUI ()

```

## JComponent 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
public abstract class JComponent
    extends Container
    implements Serializable

```

这个类表示 Swing 组件的父类（基类），其中给出了大量为所有组件共有的功能。

### Inner Class

```
class      JComponent.AccessibleJComponent
```

### Fields

protected AccessibleContext	accessibleContext
protected EventListenerList	listenerList
static String	TOOL_TIP_TEXT_KEY
protected ComponentUI	ui
static int	UNDEFINED_CONDITION
static int	WHEN_ANCESTOR_OF_FOCUSED_COMPONENT
static int	WHEN_FOCUSED
static int	WHEN_IN_FOCUSED_WINDOW

### Constructor

```
JComponent ()
```

### Methods

void	addAncestorListener (AncestorListener listener)
void	addNotify ()
void	addPropertyChangeListener                            (PropertyChangeListener listener)
void	addVetoableChangeListener

```
(VetoableChangeListener listener)
void computeVisibleRect (Rectangle visibleRect)
boolean contains (int x, int y)
JToolTip createToolTip ()
void firePropertyChange (String propertyName,
    boolean oldValue, boolean newValue)
void firePropertyChange (String propertyName, byte
    oldValue, byte newValue)
void firePropertyChange (String propertyName, char
    oldValue, char newValue)
void firePropertyChange (String propertyName,
    double oldValue, double newValue)
void firePropertyChange (String propertyName,
    float oldValue, float newValue)
void firePropertyChange (String propertyName, int
    oldValue, int newValue)
void firePropertyChange (String propertyName, long
    oldValue, long newValue)
protected void firePropertyChange (String propertyName,
    Object oldValue, Object newValue)
void firePropertyChange (String propertyName,
    short oldValue, short newValue)
protected void fireVetoableChange (String propertyName,
    Object oldValue, Object newValue)
AccessibleContext getAccessibleContext ()
ActionListener getActionForKeyStroke (KeyStroke aKeyStroke)
float getAlignmentX ()
float getAlignmentY ()
boolean getAutoscrolls ()
Border getBorder ()
Rectangle getBounds (Rectangle rv)
Object getClientProperty (Object key)
protected Graphics getComponentGraphics (Graphics g)
int getConditionForKeyStroke (KeyStroke aKeyStroke)
int getDebugGraphicsOptions ()
Graphics getGraphics ()
int getHeight ()
Insets getInsets ()
Insets getInsets (Insets insets)
Point getLocation (Point rv)
Dimension getMaximumSize ()
Dimension getMinimumSize ()
Component getNextFocusableComponent ()
Dimension getPreferredSize ()
KeyStroke [] getRegisteredKeyStrokes ()
```

```
JRootPane      getRootPane ()
Dimension     getSize (Dimension rv)
Point         getToolTipLocation (MouseEvent event)
String         getToolTipText ()
String         getToolTipText (MouseEvent event)
Container     getTopLevelAncestor ()
String         getUIClassID ()
Rectangle    getVisibleRect ()
int           getWidth ()
int           getX ()
int           getY ()
void          grabFocus ()
boolean        hasFocus ()
boolean        isDoubleBuffered ()
boolean        isFocusCycleRoot ()
boolean        isFocusTraversable ()
static boolean isLightweightComponent (Component c)
boolean        isManagingFocus ()
boolean        isOpaque ()
boolean        isOptimizedDrawingEnabled ()
boolean        isPaintingTile ()
boolean        isRequestFocusEnabled ()
boolean        isValidateRoot ()
void          paint (Graphics g)
protected void paintBorder (Graphics g)
protected void paintChildren (Graphics g)
protected void paintComponent (Graphics g)
void          paintImmediately (int x, int y, int w, int h)
void          paintImmediately (Rectangle r)
protected String paramString ()
protected void processComponentKeyEvent (KeyEvent e)
protected void processFocusEvent (FocusEvent e)
protected void processKeyEvent (KeyEvent e)
protected void processMouseMotionEvent (MouseEvent e)
void          putClientProperty (Object key, Object value)
void          registerKeyboardAction (ActionListener
anAction, KeyStroke aKeyStroke, int
aCondition)
void          registerKeyboardAction (ActionListener
anAction, String aCommand, KeyStroke
aKeyStroke, int aCondition)
void          removeAncestorListener (AncestorListener listener)
void          removeNotify ()
void          removePropertyChangeListener
(PropertyChangeListener listener)
```

```

void           removeVetoableChangeListener
               (VetoableChangeListener listener)
void           repaint (long tm, int x, int y, int width, int height)
void           repaint (Rectangle r)
boolean        requestDefaultFocus ()
void           requestFocus ()
void           resetKeyboardActions ()
void           reshape (int x, int y, int w, int h)
void           revalidate ()
void           scrollRectToVisible (Rectangle aRect)
void           setAlignmentX (float alignmentX)
void           setAlignmentY (float alignmentY)
void           setAutoscrolls (boolean autoscrolls)
void           setBackground (Color bg)
void           setBorder (Border border)
void           setDebugGraphicsOptions (int debugOptions)
void           setDoubleBuffered (boolean aFlag)
void           setEnabled (boolean enabled)
void           setFont (Font font)
void           setForeground (Color fg)
void           setMaximumSize (Dimension maximumSize)
void           setMinimumSize (Dimension minimumSize)
void           setNextFocusableComponent (Component aComponent)
void           setOpaque (boolean isOpaque)
void           setPreferredSize (Dimension preferredSize)
void           setRequestFocusEnabled (boolean aFlag)
void           setToolTipText (String text)
protected void setUI (ComponentUI newUI)
void           setVisible (boolean aFlag)
void           unregisterKeyboardAction (KeyStroke aKeyStroke)
void           update (Graphics g)
void           updateUI ()

```

## JDesktopPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLayeredPane
|
+-- javax.swing.JDesktopPane

public class JDesktopPane
    extends JLayeredPane
    implements Accessible

```

这个类表示一个桌面窗格，它的父类是分层窗格。这种桌面窗格提供了一个虚拟的桌面。内部框架添加到了桌面窗格的层上。桌面窗格还为当前外观提供了一个对桌面管理器的引用。

### Inner Class

```
protected class JDesktopPane.AccessibleJDesktopPane
```

### Constructor

```
JDesktopPane ()
```

### Methods

AccessibleContext	getAccessibleContext ()
JInternalFrame []	getAllFrames ()
JInternalFrame []	getAllFramesInLayer (int layer)
DesktopManager	getDesktopManager ()
DesktopPaneUI	getUI ()
String	getUIClassID ()
boolean	isOpaque ()
protected String	paramString ()
void	setDesktopManager (DesktopManager d)
void	setUI (DesktopPaneUI ui)
void	updateUI ()

## JDialog 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- java.awt.Window
|
+-- java.awt.Dialog
|
+-- javax.swing.JDialog

public class JDialog
    extends Dialog
    implements WindowConstants, Accessible, RootPaneContainer
```

这个类表示重量级的 Swing 对话框。可以用这个类创建定制的对话框。至于标准对话框，则可使用它所支持的选项窗格。JDialog 对象支持诸如窗格的分层、玻璃窗格、菜单条等等特性。必须将组件添加到对话框的内容窗格上。

### Inner Class

```
protected class JDialog.AccessibleJDialog
```

### Fields

protected AccessibleContext	accessibleContext
protected JRootPane	rootPane

protected boolean rootPaneCheckingEnabled

### Constructors

JDialog ()  
 JDialog (Frame owner)  
 JDialog (Frame owner, boolean modal)  
 JDialog (Frame owner, String title)  
 JDialog (Frame owner, String title, boolean modal)

### Methods

protected void	addImpl (Component comp, Object constraints, int index)
protected JRootPane	createRootPane ()
protected void	dialogInit ()
AccessibleContext	getAccessibleContext ()
Container	getContentPane ()
int	getDefaultCloseOperation ()
Component	getGlassPane ()
JMenuBar	getJMenuBar ()
JLayeredPane	getLayeredPane ()
JRootPane	getRootPane ()
protected boolean	isRootPaneCheckingEnabled ()
protected String	paramString ()
protected void	processWindowEvent (WindowEvent e)
void	setContentPane (Container contentPane)
void	setDefaultCloseOperation (int operation)
void	setGlassPane (Component glassPane)
void	setJMenuBar (JMenuBar menu)
void	setLayeredPane (JLayeredPane layeredPane)
void	setLayout (LayoutManager manager)
void	setLocationRelativeTo (Component c)
protected void	setRootPane (JRootPane root)
protected void	setRootPaneCheckingEnabled (boolean enabled)
void	update (Graphicsg)

## JEditorPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
|
+-- javax.swing.JEditorPane

public class JEditorPane
  extends JTextComponent

```

这个类表示 Swing 编辑器窗格。编辑器工具箱会根据 URL 所指定内容的类型与这个组件建立关联。该编辑器窗格具有自动配置其自身来显示内容的能力。

### Inner Classes

protected class	JEditorPane.AccessibleJEditorPane
protected class	JEditorPane.AccessibleJEditorPanelHTML
protected class	JEditorPane.JEditorPaneAccessibleHypertextSupport

### Constructors

JEditorPane ()
JEditorPane (String url)
JEditorPane (String type, String text)
JEditorPane (URL initialPage)

### Methods

void	addHyperlinkListener (HyperlinkListener listener)
protected EditorKit	createDefaultEditorKit ()
static EditorKit	createEditorKitForContentType (String type)
void	fireHyperlinkUpdate (HyperlinkEvent e)
AccessibleContext	getAccessibleContext ()
String	getContentType ()
EditorKit	getEditorKit ()
EditorKit	getEditorKitForContentType (String type)
URL	getPage ()
Dimension	getPreferredSize ()
boolean	getScalableTracksViewportHeight ()
boolean	getScalableTracksViewportWidth ()
protected InputStream	getStream (URL page)
String	getText ()
String	getUIClassID ()
boolean	isManagingFocus ()
protected String	paramString ()
protected void	processComponentKeyEvent (KeyEvent e)
void	read (InputStream in, Object desc)
static void	registerEditorKitForContentType (String type, String classname)
static void	registerEditorKitForContentType (String type, String classname, ClassLoader loader)
void	removeHyperlinkListener (HyperlinkListener listener)
void	replaceSelection (String content)
protected void	scrollToReference (String reference)
void	setContentType (String type)
void	setEditorKit (EditorKit kit)

```

void           setEditorKitForContentType (String type, EditorKit k)
void           setPage (String url)
void           setPage (URI page)
void           setText (String t)

```

## JFileChooser 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JFileChooser

```

```

public class JFileChooser
    extends JComponent
    implements Accessible

```

这个类表示 Swing 中的文件选择器。这个类的对象就是允许用户打开或保存文件的对话框。

### Inner Class

```
protected class     JFileChooser.AccessibleJFileChooser
```

### Fields

protected AccessibleContext	accessibleContext
static String	ACCESSORY_CHANGED_PROPERTY
static String	APPROVE_BUTTON_MNEMONIC
	_CHANGED_PROPERTY
static String	APPROVE_BUTTON_TEXT
	_CHANGED_PROPERTY
static String	APPROVE_BUTTON_TOOL_TIP_TEXT
	_CHANGED_PROPERTY
static int	APPROVE_OPTION
static String	APPROVE_SELECTION
static int	CANCEL_OPTION
static String	CANCEL_SELECTION
static String	CHOOSEABLE_FILE_FILTER
	_CHANGED_PROPERTY
static int	CUSTOM_DIALOG
static String	DIALOG_TITLE_CHANGED_PROPERTY
static String	DIALOG_TYPE_CHANGED_PROPERTY
static int	DIRECTORIES_ONLY
static String	DIRECTORY_CHANGED_PROPERTY
static int	ERROR_OPTION

static String	FILE_FILTER_CHANGED_PROPERTY
static String	FILE HIDING_CHANGED_PROPERTY
static String	FILE_SELECTION_MODE
	_CHANGED_PROPERTY
static String	FILE_SYSTEM_VIEW_CHANGED_PROPERTY
static String	FILE_VIEW_CHANGED_PROPERTY
static int	FILES_AND_DIRECTORIES
static int	FILES_ONLY
static String	MULTI_SELECTION_ENABLED
CHANGED_PROPERTY	
static int	OPEN_DIALOG
static int	SAVE_DIALOG
static String	SELECTED_FILE_CHANGED_PROPERTY
static String	SELECTED_FILES_CHANGED_PROPERTY

### Constructors

```
JFileChooser()
JFileChooser(File currentDirectory)
JFileChooser(File currentDirectory, FileSystemView fsv)
JFileChooser(FileSystemView fsv)
JFileChooser(String currentDirectoryPath)
JFileChooser(String currentDirectoryPath, FileSystemView fsv)
```

## JFrame 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- java.awt.Window
|
+-- java.awt.Frame
|
+-- javax.swing.JFrame

public class JFrame
    extends Frame
    implements WindowConstants, Accessible, RootPaneContainer
```

这个类表示具有 AWT 框架功能扩展的 Swing 框架。可以看到它对组件分层 (Z-序)、玻璃窗格、菜单条和内容窗格的支持。需要把子组件添加到框架对象的内容面板上。

### Inner Class

```
protected class JFrame.AccessibleJFrame
```

### Fields

```
protected AccessibleContext accessibleContext
```

```
protected JRootPane      rootPane
protected boolean        rootPaneCheckingEnabled
```

**Constructors**

```
JFrame ()
JFrame (String title)
```

**Methods**

```
protected void            addImpl (Component comp, Object constraints, int index)
protected JRootPane       createRootPane ()
protected void            getAccessibleContext ()
Container               getContentPane ()
int                     getDefaultCloseOperation ()
Component               getGlassPane ()
JMenuBar                getJMenuBar ()
JLayeredPane            getLayeredPane ()
JRootPane               getRootPane ()
protected boolean         isRootPaneCheckingEnabled ()
protected String          paramString ()
protected void            processKeyEvent (KeyEvent e)
protected void            processWindowEvent (WindowEvent e)
void                    setContentPane (Container contentPane)
void                    setDefaultCloseOperation (int operation)
void                    setGlassPane (Component glassPane)
void                    setJMenuBar (JMenuBar menubar)
void                    setLayeredPane (JLayeredPane layeredPane)
void                    setLayout (LayoutManager manager)
protected void            setRootPane (JRootPane root)
protected void            setRootPaneCheckingEnabled (boolean enabled)
void                    update (Graphics g)
```

**JInternalFrame 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JInternalFrame

public class JInternalFrame
    extends JComponent
    implements Accessible, WindowConstants, RootPaneContainer
```

这个类表示轻量级的 Swing 内部框架。可以创建这个类的一个实例并将它加到桌面窗格上。

**Inner Classes**

protected class	JInternalFrame.AccessibleJInternalFrame
static class	JInternalFrame.JDesktopIcon

**Fields**

protected boolean	closable
static String	CONTENT_PANE_PROPERTY
protected JInternalFrame.JDesktopIcon	desktopIcon
static String	FRAME_ICON_PROPERTY
protected Icon	frameIcon
static String	GLASS_PANE_PROPERTY
protected boolean	iconable
static String	IS_CLOSED_PROPERTY
static String	IS_ICON_PROPERTY
static String	IS_MAXIMUM_PROPERTY
static String	IS_SELECTED_PROPERTY
protected boolean	isClosed
protected boolean	isIcon
protected boolean	isMaximum
protected boolean	isSelected
static String	LAYERED_PANE_PROPERTY
protected boolean	maximizable
static String	MENU_BAR_PROPERTY
protected boolean	resizable
static String	ROOT_PANE_PROPERTY
protected JRootPane	rootPane
protected boolean	rootPaneCheckingEnabled
protected String	title
static String	TITLE_PROPERTY

**Constructors**

JInternalFrame ()	
JInternalFrame (String title)	
JInternalFrame (String title, boolean resizable)	
JInternalFrame (String title, boolean resizable, boolean closable)	
JInternalFrame (String title, boolean resizable, boolean closable, boolean maximizable)	
JInternalFrame (String title, boolean resizable, boolean closable, boolean maximizable, boolean iconifiable)	

**Methods**

protected void	addImpl (Component comp, Object constraints, int index)
void	addInternalFrameListener

```
(InternalFrameListener 1)
protected JRootPane           createRootPane ()
void                         dispose ()
protected void                fireInternalFrameEvent (int id)
AccessibleContext             getAccessibleContext ()
Color                          getBackground ()
Container                     getContentPane ()
int                           getDefaultCloseOperation ()
JInternalFrame.JDesktopIcon  getDesktopIcon ()
JDesktopPane                  getDesktopPane ()
Color                          getForeground ()
Icon                           getFrameIcon ()
Component                     getGlassPane ()
JMenuBar                      getJMenuBar ()
int                           getLayer ()
JLayeredPane                 getLayeredPane ()
JMenuBar                      getMenuBar ()
// Deprecated. Use the method
getJMenuBar () .
JRootPane                     getRootPane ()
String                         getTitle ()
InternalFrameUI               getUI ()
String                         getUIClassID ()
String                         getWarningString ()
boolean                        isClosable ()
boolean                        isClosed ()
boolean                        isIcon ()
boolean                        isIconifiable ()
boolean                        isMaximizable ()
boolean                        isMaximum ()
boolean                        isResizable ()
protected boolean              isRootPaneCheckingEnabled ()
boolean                        isSelected ()
void                           moveToFront ()
void                           pack ()
protected String              paramString ()
void                           removeInternalFrameListener
                               (InternalFrameListener 1)
void                           reshape (int x, int y, int width, int height)
void                           setBackground (Color c)
void                           setClosable (boolean b)
void                           setClosed (boolean b)
```

```

void           setContentPane (Container c)
void           setDefaultCloseOperation (int operation)
void           setDesktopIcon
              (JInternalFrame.JDesktopIcon d)
void           setForeground (Color c)
void           setFrameIcon (Icon icon)
void           setGlassPane (Component glass)
void           setIcon (boolean b)
void           setIconifiable (boolean b)
void           setJMenuBar (JMenuBar m)
void           setLayer (Integer layer)
void           setLayeredPane (JLayeredPane layered)
void           setLayout (LayoutManager manager)
void           setMaximizable (boolean b)
void           setMaximum (boolean b)
void           setMenuBar (JMenuBar m)
              // Deprecated. Use the method
              setJMenuBar (JMenuBar m) .
void           setResizable (boolean b)
protected void setRootPane (JRootPane root)
protected void setRootPaneCheckingEnabled (boolean enabled)
void           setSelected (boolean selected)
void           setTitle (String title)
void           setUI (InternalFrameUI ui)
void           setVisible (boolean b)
void           show ()
void           toBack ()
void           toFront ()
void           updateUI ()

```

### JInternalFrame.JDesktopIcon 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JInternalFrame.JDesktopIcon
public static class JInternalFrame.JDesktopIcon
    extends JComponent
    implements Accessible

```

这个类表示内部框架的桌面图标。

**Inner Class**

```
protected class JInternalFrame.JDesktopIcon.AccessibleJDesktopIcon
```

**Constructor**

```
JInternalFrame.JDesktopIcon (JInternalFrame f)
```

**Methods**

AccessibleContext	getAccessibleContext ()
JDesktopPane	getDesktopPane ()
JInternalFrame	getInternalFrame ()
DesktopIconUI	getUI ()
String	getUIClassID ()
void	setInternalFrame (JInternalFrame f)
void	setUI (DesktopIconUI ui)
void	updateUI ()

**JLabel 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel

public class JLabel
    extends JComponent
    implements SwingConstants, Accessible
```

这个类表示轻量级的 Swing 标签，它选择性地显示文本字符串或图像。

**Inner Class**

```
protected class JLabel.AccessibleJLabel
Field
protected Component labelFor
```

**Constructors**

```
JLabel ()
JLabel (Icon image)
JLabel (Icon image, int horizontalAlignment)
JLabel (String text)
JLabel (String text, Icon icon, int horizontalAlignment)
JLabel (String text, int horizontalAlignment)
```

**Methods**

```
protected int checkHorizontalKey (int key, String message)
```

```

protected int      checkVerticalKey ( int key, String message)
AccessibleContext getAccessibleContext ()
Icon             getDisabledIcon ()
int              getDisplayedMnemonic ()
int              getHorizontalAlignment ()
int              getHorizontalTextPosition ()
Icon             getIcon ()
int              getIconTextGap ()
Component        getLabelFor ()
String           getText ()
LabelUI          getUI ()
String           getUIClassID ()
int              getVerticalAlignment ()
int              getVerticalTextPosition ()
protected String paramString ()
void             setDisabledIcon (Icon disabledIcon)
void             setDisplayedMnemonic (char aChar)
void             setDisplayedMnemonic (int key)
void             setHorizontalAlignment (int alignment)
void             setHorizontalTextPosition (int textPosition)
void             setIcon (Icon icon)
void             setIconTextGap (int iconTextGap)
void             setLabelFor (Component c)
void             setText (String text)
void             setUI (LabelUI ui)
void             setVerticalAlignment (int alignment)
void             setVerticalTextPosition (int textPosition)
void             updateUI ()

```

## JLayeredPane 类

```

java.lang.Object
|
+
+ -- java.awt.Component
|
+
+ -- java.awt.Container
|
+
+ -- javax.swing.JComponent
|
+
+ -- javax.swing.JLayeredPane

public class JLayeredPane
    extends JComponent
    implements Accessible

```

这个类表示 Swing 的分层窗格。分层窗格能够产生深度感。通过从 Swing 小程序、框架、对话框、窗口等组件中调用 `getLayerPane ()` 方法，可以分别得到对容器下的各个分层

窗格的引用。

### Inner Class

protected class	JLayeredPane.AccessibleJLayeredPane
-----------------	-------------------------------------

### Fields

static Integer	DEFAULT_LAYER
static Integer	DRAG_LAYER
static Integer	FRAME_CONTENT_LAYER
static String	LAYER_PROPERTY
static Integer	MODAL_LAYER
static Integer	PALETTE_LAYER
static Integer	POPUP_LAYER

### Constructor

JLayeredPane ()	
-----------------	--

### Methods

protected void	addImpl (Component comp, Object constraints, int index)
AccessibleContext	getAccessibleContext ()
int	getComponentCountInLayer (int layer)
Component []	getComponentsInLayer (int layer)
protected Hashtable	getComponentToLayer ()
int	getIndexOf (Component c)
int	getLayer (Component c)
static int	getLayer (JComponent c)
static JLayeredPane	getLayeredPaneAbove (Component c)
protected Integer	getObjectForLayer (int layer)
int	getPosition (Component c)
int	highestLayer ()
protected int	insertIndexForLayer (int layer, int position)
boolean	isOptimizedDrawingEnabled ()
int	lowestLayer ()
void	moveToBack (Component c)
void	moveToFront (Component c)
void	paint (Graphics g)
protected String	paramString ()
static void	putLayer (JComponent c, int layer)
void	remove (int index)
void	setLayer (Component c, int layer)
void	setLayer (Component c, int layer, int position)
void	setPosition (Component c, int position)

## JList 类

```

java.lang.Object
|
+
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JList

```

```

public class JList
    extends JComponent
    implements Scrollable, Accessible

```

这个类表示轻量级的 Swing 列表。列表就是一列可选择的条目。

### Inner Class

```
protected class JList.AccessibleJList
```

### Constructors

```

JList ()
JList (ListModel dataModel)
JList (Object [] listData)
JList (Vector listData)

```

### Methods

void	addListSelectionListener (ListSelectionListener listener)
void	addSelectionInterval (int anchor, int lead)
void	clearSelection ()
protected ListSelectionModel	createSelectionModel ()
void	ensureIndexIsVisible (int index)
protected void	fireSelectionValueChanged (int firstIndex, int lastIndex, boolean isAdjusting)
AccessibleContext	getAccessibleContext ()
int	getAnchorSelectionIndex ()
Rectangle	getCellBounds (int index1, int index2)
ListCellRenderer	getCellRenderer ()
int	getFirstVisibleIndex ()
int	getFixedCellHeight ()
int	getFixedCellWidth ()
int	getLastVisibleIndex ()
int	getLeadSelectionIndex ()
int	getMaxSelectionIndex ()
int	getMinSelectionIndex ()

ListModel	getModel ()
Dimension	getPreferredScrollable
	ViewportSize ()
Object	getPrototypeCellValue ()
int	getScrollableBlockIncrement (Rectangle visibleRect, orientation, int direction)
boolean	getScrollableTracksViewport Height ()
boolean	getScrollableTracksViewport Width ()
int	getScrollableUnitIncrement (Rectangle visibleRect, orientation, int direction)
int	getSelectedIndex ()
int []	getSelectedIndices ()
Object	getSelectedValue ()
Object	getSelectedValues ()
Color	getSelectionBackground ()
Color	getSelectionForeground ()
int	getSelectionMode ()
ListSelectionModel	getSelectionModel ()
ListUI	getUI ()
String	getUIClassID ()
boolean	getValueIsAdjusting ()
int	getVisibleRowCount ()
Point	indexToLocation (int index)
boolean	isSelectedIndex (int index)
boolean	isSelectionEmpty ()
int	locationToIndex (Point location)
protected String	paramString ()
void	removeListSelectionListener (ListSelectionListener listener)
void	removeSelectionInterval (int index0, int index1)
void	setCellRenderer (ListCellRenderer cellRenderer)
void	setFixedCellHeight (int height)
void	setFixedCellWidth (int width)
void	setListData (Object [] listData)
void	setListData (Vector listData)
void	setModel (ListModel model)
void	setPrototypeCellValue (Object

```

        prototypeCellValue)
void      setSelectedIndex (int index)
void      setSelectedIndices (int [] indices)
void      setSelectedValue (Object anObject,
                           boolean shouldScroll)
void      setSelectionBackground (Color
                                  selectionBackground)
void      setSelectionForeground (Color
                                  selectionForeground)
void      setSelectionInterval (int anchor, int lead)
void      setSelectionMode (int selectionMode)
void      setSelectionModel
          (ListSelectionModel
           selectionModel)
void      setUI (ListUI ui)
void      setValueIsAdjusting (boolean b)
void      setVisibleRowCount (int visibleRowCount)
void      updateUI ()

```

## JMenu 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JMenuItem
|
+-- javax.swing.JMenu

public class JMenu
    extends JMenuItem
    implements Accessible, MenuElement

```

这个类表示常规的 Swing 菜单。菜单中包括可选择的菜单项和菜单项分隔线。

### Inner Classes

```

protected class JMenu.AccessibleJMenu
protected class JMenu.WinListener

```

### Field

```
protected JMenu.WinListener popupListener
```

### Constructors

```
JMenu ()
```

JMenu (String s)  
 JMenu (String s, boolean b)

**Methods**

JMenuItem	add (Action a)
Component	add (Component c)
JMenuItem	add (JMenuItem menuItem)
JMenuItem	add (String s)
void	addMenuListener (MenuListener l)
void	addSeparator ()
protected PropertyChangeListener	createActionChangeListener (JMenuItem b)
protected JMenu.WinListener	createWinListener (JPopupMenu p)
void	doClick (int pressTime)
protected void	fireMenuCanceled ()
protected void	fireMenuDeselected ()
protected void	fireMenuSelected ()
AccessibleContext	getAccessibleContext ()
Component	getComponent ()
int	getDelay ()
JMenuItem	getItem (int pos)
int	getItemCount ()
Component	getMenuComponent (int n)
int	getMenuComponentCount ()
Component []	getMenuComponents ()
JPopupMenu	getPopupMenu ()
MenuElement []	getSubElements ()
String	getUIClassID ()
JMenuItem	insert (Action a, int pos)
JMenuItem	insert (JMenuItem mi, int pos)
void	insert (String s, int pos)
void	insertSeparator (int index)
boolean	isMenuComponent (Component c)
boolean	isPopupMenuVisible ()
boolean	isSelected ()
boolean	isTearOff ()
boolean	isTopLevelMenu ()
void	menuSelectionChanged (boolean isIncluded)
protected String	paramString ()
protected void	processKeyEvent (KeyEvent e)
void	remove (Component c)
void	remove (int pos)
void	remove (JMenuItem item)

```

void                     removeAll ()
void                     removeMenuListener
                        (MenuListener l)
void                     setAccelerator (KeyStroke keyStroke)
void                     setDelay (int d)
void                     setMenuLocation (int x, int y)
void                     setModel (ButtonModel newModel)
void                     setPopupMenuVisible (boolean b)
void                     setSelected (boolean b)
void                     updateUI ()

```

## JMenuBar 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JMenuBar

```

```

public class JMenuBar
    extends JComponent
    implements Accessible, MenuElement

```

这个类表示 Swing 菜单条。菜单条上带有菜单。对 Swing 小程序、框架、对话框和窗口来说，通过调用 getJMenuBar () 方法能够获得菜单条的句柄。

### Inner Classes

```
protected class JMenuBar.AccessibleJMenuBar
```

### Constructors

```
JMenuBar ()
```

### Methods

JMenu	add (JMenu c)
void	addNotify ()
AccessibleContext	getAccessibleContext ()
Component	getComponent ()
Component	getComponentAtIndex (int i)
int	getComponentIndex (Component c)
JMenu	getHelpMenu ()
Insets	getMargin ()
JMenu	getMenu (int index)
int	getMenuCount ()
SingleSelectionModel	getSelectionModel ()

```
MenuBarUI getUI ()  
String getUIClassID ()  
boolean isBorderPainted ()  
boolean isManagingFocus ()  
boolean isSelected ()  
void menuSelectionChanged (boolean isIncluded)  
protected void paintBorder (Graphics g)  
protected String paramString ()  
void processKeyEvent (KeyEvent e, MenuElement [] path, MenuSelectionManager manager)  
void processMouseEvent (MouseEvent event, MenuElement [] path, MenuSelectionManager manager)  
void removeNotify ()  
void setBorderPainted (boolean s)  
void setHelpMenu (JMenu menu)  
void setMargin (Insets margin)  
void setSelected (Component sel)  
void setSelectionModel (SingleSelectionModel model)  
void setUI (MenuBarUI ui)  
void updateUI ()
```

### JMenuItem 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JMenuItem
```

这个类表示 Swing 菜单项。菜单项的功能类似于按钮，它们也会触发动作事件。

### Inner Class

**protected class** JMenuItem.AccessibleMenuItem

## Constructors

```
JMenuItem ()
JMenuItem (Icon icon)
JMenuItem (String text)
JMenuItem (String text, Icon icon)
JMenuItem (String text, int mnemonic)
```

## Methods

void	addMenuDragMouseListener (MenuDragMouseListener l)
void	addMenuKeyListener (MenuKeyListener l)
protected void	fireMenuDragMouseDragged (MenuDragMouseEvent event)
protected void	fireMenuDragMouseEntered (MenuDragMouseEvent event)
protected void	fireMenuDragMouseExited (MenuDragMouseEvent event)
protected void	fireMenuDragMouseReleased (MenuDragMouseEvent event)
protected void	fireMenuKeyPressed (MenuKeyEvent event)
protected void	fireMenuKeyReleased (MenuKeyEvent event)
protected void	fireMenuKeyTyped (MenuKeyEvent event)
KeyStroke	getAccelerator ()
AccessibleContext	getAccessibleContext ()
Component	getComponent ()
MenuElement []	getSubElements ()
String	getUIClassID ()
protected void	init (String text, Icon icon)
boolean	isArmed ()
void	menuSelectionChanged (boolean isIncluded)
protected String	paramString ()
void	processKeyEvent (KeyEvent e, MenuElement [] path, MenuSelectionManager manager)
void	processMenuDragMouseEvent (MenuDragMouseEvent e)
void	processMenuKeyEvent (MenuKeyEvent e)
void	processMouseEvent (MouseEvent e, MenuElement [] path, MenuSelectionManager manager)
void	removeMenuDragMouseListener (MenuDragMouseListener l)
void	removeMenuKeyListener (MenuKeyListener l)
void	setAccelerator (KeyStroke keyStroke)
void	setArmed (boolean b)
void	setEnabled (boolean b)
void	setUI (MenuItemUI ui)
void	updateUI ()

## JOptionPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JOptionPane
public class JOptionPane
    extends JComponent
    implements Accessible

```

这个类表示 Swing 选项窗格。选项窗格是一种中间类，它通过调用相关的静态方法来简化标准对话框的使用。

### Inner Class

```
protected class JOptionPane.AccessibleJOptionPane
```

### Fields

static int	CANCEL_OPTION
static int	CLOSED_OPTION
static int	DEFAULT_OPTION
static int	ERROR_MESSAGE
protected Icon	icon
static String	ICON_PROPERTY
static int	INFORMATION_MESSAGE
static String	INITIAL_SELECTION_VALUE_PROPERTY
static String	INITIAL_VALUE_PROPERTY
protected Object	initialSelectionValue
protected Object	initialValue
static String	INPUT_VALUE_PROPERTY
protected Object	inputValue
protected Object	message
static String	MESSAGE_PROPERTY
static String	MESSAGE_TYPE_PROPERTY
protected int	messageType
static int	NO_OPTION
static int	OK_CANCEL_OPTION
static int	OK_OPTION
static String	OPTION_TYPE_PROPERTY
protected Object []	options
static String	OPTIONS_PROPERTY
protected int	optionType

static int	PLAIN_MESSAGE
static int	QUESTION_MESSAGE
static String	SELECTION_VALUES_PROPERTY
protected Object []	selectionValues
static Object	UNINITIALIZED_VALUE
protected Object	value
static String	VALUE_PROPERTY
static String	WANTS_INPUT_PROPERTY
protected boolean	wantsInput
static int	WARNING_MESSAGE
static int	YES_NO_CANCEL_OPTION
static int	YES_NO_OPTION
static int	YES_OPTION

## Constructors

```
JOptionPane ()
JOptionPane (Object message)
JOptionPane (Object message, int messageType)
JOptionPane (Object message, int messageType, int optionType)
JOptionPane (Object message, int messageType, int optionType, Icon icon)
JOptionPane (Object message, int messageType, int optionType, Icon icon,
Object [] options)
JOptionPane (Object message, int messageType, int optionType, Icon icon,
Object [] options, Object initialValue)
```

## Methods

JDialog	createDialog (Component parentComponent, String title)
JInternalFrame	createInternalFrame (Component parentComponent, String title)
AccessibleContext	getAccessibleContext ()
static JDesktopPane	getDesktopPaneForComponent (Component parentComponent)
static Frame	getFrameForComponent (Component parentComponent)
Icon	getIcon ()
Object	getInitialSelectionValue ()
Object	getInitialValue ()
Object	getInputValue ()
int	getMaxCharactersPerLineCount ()
Object	getMessage ()
int	getMessageType ()
Object []	getOptions ()
int	getOptionType ()
static Frame	getRootFrame ()
Object []	getSelectionValues ()
OptionPaneUI	getUI ()
String	getUIClassID ()

Object	getValue ()
boolean	getWantsInput ()
protected String	paramString ()
void	selectInitialValue ()
void	setIcon (Icon newIcon)
void	setInitialSelectionValue (Object newValue)
void	setInitialValue (Object newValue)
void	setInputValue (Object newValue)
void	setMessage (Object newMessage)
void	setMessageType (int newType)
void	setOptions (Object [] newOptions)
void	setOptionType (int newType)
static void	setRootFrame (Frame newRootFrame)
void	setSelectionValues (Object [] newValues)
void	setUI (OptionPaneUI ui)
void	setValue (Object newValue)
void	setWantsInput (boolean newValue)
static int	showConfirmDialog (Component parentComponent, Object message)
static int	showConfirmDialog (Component parentComponent, Object message, String title, int optionType)
static int	showConfirmDialog (Component parentComponent, Object message, String title, int optionType, int messageType)
static int	showConfirmDialog (Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)
static String	showInputDialog (Component parentComponent, Object message)
static String	showInputDialog (Component parentComponent, Object message, String title, int messageType)
static Object	showInputDialog (Component parentComponent, Object message, String title, int messageType, Icon icon, Object [] selectionValues, Object initialValue)
static String	showInputDialog (Object message)
static int	showInternalConfirmDialog (Component parentComponent, Object message)
static int	showInternalConfirmDialog (Component parentComponent, Object message, String title, int optionType)

```
static int      showInternalConfirmDialog (Component parentComponent, Object message, String title, int optionType, int messageType)
static int      showInternalConfirmDialog (Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)
static String   showInternalInputDialog (Component parentComponent, Object message)
static String   showInternalInputDialog (Component parentComponent, Object message, String title, int messageType)
static Object   showInternalInputDialog (Component parentComponent, Object message, String title, int messageType, Icon icon, Object [] selectionValues, Object initialValue)
static void     showInternalMessageDialog (Component parentComponent, Object message)
static void     showInternalMessageDialog (Component parentComponent, Object message, String title, int messageType)
static void     showInternalMessageDialog (Component parentComponent, Object message, String title, int messageType, Icon icon)
static int      showInternalOptionDialog (Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object [] options, Object initialValue)
static void     showMessageDialog (Component parentComponent, Object message)
static void     showMessageDialog (Component parentComponent, Object message, String title, int messageType)
static void     showMessageDialog (Component parentComponent, Object message, String title, int messageType, Icon icon)
static int      showOptionDialog (Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object [] options, Object initialValue)
void          updateUI ()
```

## JPanel 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JPanel

public class JPanel
    extends JComponent
    implements Accessible

```

这个类表示轻量级的 Swing 窗格。窗格可用于组合子组件。Swing 窗格能够有选择地支持双缓冲特性。

### Inner Class

```
protected class JPanel.AccessibleJPanel
```

### Constructors

```

JPanel ()
JPanel (boolean isDoubleBuffered)
JPanel (LayoutManager layout)
JPanel (LayoutManager layout, boolean isDoubleBuffered)

```

### Methods

AccessibleContext	getAccessibleContext ()
String	getUIClassID ()
protected String	paramString ()
void	updateUI ()

## JPasswordField 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
|
+-- javax.swing.JTextField
|
+-- javax.swing.JPasswordField

public class JPasswordField
    extends JTextField

```

这个类表示 Swing 支持的口令字段。该组件类似于文本字段，只是它不显示键入的字符，取而代之的是，对每一个键入的字符都显示缺省字符 \*（它是可以改变的）。

### Inner Class

```
protected class JPasswordField.AccessibleJPasswordField
```

### Constructors

```
JPasswordField()
JPasswordField(Document doc, String txt, int columns)
JPasswordField(int columns)
JPasswordField(String text)
JPasswordField(String text, int columns)
```

### Methods

void	copy ()
void	cut ()
boolean	echoCharIsSet ()
AccessibleContext	getAccessibleContext ()
char	getEchoChar ()
char []	getPassword ()
String	getText ()
String	getText (int offs, int len)
String	getUIClassID ()
protected String	paramString ()
void	setEchoChar (char c)

## JPopupMenu 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JPopupMenu

public class JPopupMenu
    extends JComponent
    implements Accessible, MenuElement
```

这个类表示 Swing 的弹出式菜单，菜单项可以添加在这个菜单上。

### Inner Classes

```
protected class JPopupMenu.AccessibleJPopupMenu
static class JPopupMenu.Separator
```

**Constructors**

JPopupMenu ()  
JPopupMenu (String label)

**Methods**

JMenuItem	add (Action a)
JMenuItem	add (JMenuItem menuItem)
JMenuItem	add (String s)
void	addPopupMenuListener (PopupMenuListener l)
void	addSeparator ()
protected PropertyChangeListener	createActionChangeListener (MenuItem b)
protected void	firePopupMenuCanceled ()
protected void	firePopupMenuWillBecome Invisible ()
protected void	firePopupMenuWillBecome Visible ()
AccessibleContext	getAccessibleContext ()
Component	getComponent ()
Component	getComponentAtIndex (int i)
int	getComponentIndex (Component c)
static boolean	getDefaultLightWeightPopup Enabled ()
Component	getInvoker ()
String	getLabel ()
Insets	getMargin ()
SingleSelectionModel	getSelectionModel ()
MenuElement []	getSubElements ()
PopupMenuUI	getUI ()
String	getUIClassID ()
void	insert (Action a, int index)
void	insert (Component component, int index)
boolean	isBorderPainted ()
boolean	isLightWeightPopupEnabled ()
boolean	isVisible ()
void	menuSelectionChanged (boolean isIncluded)
void	pack ()
protected void	paintBorder (Graphics g)
protected String	paramString ()
void	processKeyEvent (KeyEvent e, MenuElement [] path, MenuSelectionManager manager)

```

void           processMouseEvent (MouseEvent
event, MenuElement [] path,
MenuSelectionManager manager)
void           remove (Component comp)
void           removePopupMenuListener
(PopupMenuItemListener l)
void           setBorderPainted (boolean b)
static void    setDefaultLightWeightPopup
Enabled (boolean aFlag)
void           setInvoker (Component invoker)
void           setLabel (String label)
void           setLightWeightPopupEnabled
(boolean aFlag)
void           setLocation (int x, int y)
void           setPopupSize (Dimension d)
void           setPopupSize (int width, int height)
void           setSelected (Component sel)
void           setSelectionModel
(SingleSelectionModel model)
void           setUI (PopupMenuUI ui)
void           setVisible (boolean b)
void           show (Component invoker, int x, int y)
void           updateUI ()

```

## JPopupMenu.Separator 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JSeparator
|
+-- javax.swing.JPopupMenu.Separator
public static class JPopupMenu.Separator
    extends JSeparator

```

这个类表示分隔线，分隔线用来在弹出式菜单中逻辑地组合菜单项。

### Constructors

`JPopupMenu.Separator ()`

### Methods

`String getUIClassID ()`

## JProgressBar 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JProgressBar
public class JProgressBar
    extends JComponent
    implements SwingConstants, Accessible

```

这个类表示进度条，进度条通过显示完成的百分比来指示执行的进度。

### Inner Class

```
protected class JProgressBar.AccessibleJProgressBar
```

### Fields

protected ChangeEvent	changeEvent
protected ChangeListener	changeListener
protected BoundedRangeModel	model
protected int	orientation
protected boolean	paintBorder
protected boolean	paintString
protected String	progressString

### Constructors

```

JProgressBar ()
JProgressBar (BoundedRangeModel newModel)
JProgressBar (int orient)
JProgressBar (int min, int max)
JProgressBar (int orient, int min, int max)

```

### Methods

void	addChangeListener (ChangeListener l)
protected ChangeListener	createChangeListener ()
protected void	fireStateChanged ()
AccessibleContext	getAccessibleContext ()
int	getMaximum ()
int	getMinimum ()
BoundedRangeModel	getModel ()
int	getOrientation ()
double	getPercentComplete ()
String	getString ()

```

ProgressBarUI           getUI()
String                 getUIClassID()
int                   getValue()
boolean                isBorderPainted()
boolean                isStringPainted()
protected void          paintBorder (Graphics g)
protected String         paramString()
void                  removeChangeListener (ChangeListener l)
void                  setBorderPainted (boolean b)
void                  setMaximum (int n)
void                  setMinimum (int n)
void                  setModel (BoundedRangeModel newModel)
void                  setOrientation (int newOrientation)
void                  setString (String s)
void                  setStringPainted (boolean b)
void                  setUI (ProgressBarUI ui)
void                  setValue (int n)
void                  updateUI()

```

## JRadioButton 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JToggleButton
|
+-- javax.swing.JRadioButton

public class JRadioButton
    extends JToggleButton
    implements Accessible

```

这个类表示 Swing 单选按钮。为了组合单选按钮，需要创建一个 **ButtonGroup** 类型的对象，并加上所有单选按钮。从这种按钮组中每次只能选择一个按钮。

### Inner Class

```
protected class JRadioButton.AccessibleJRadioButton
```

### Constructors

```

JRadioButton ()
JRadioButton (Icon icon)
JRadioButton (Icon icon, boolean selected)

```

```
JRadioButton (String text)
JRadioButton (String text, boolean selected)
JRadioButton (String text, Icon icon)
JRadioButton (String text, Icon icon, boolean selected)
```

### Methods

AccessibleContext	getAccessibleContext ()
String	getUIClassID ()
protected String	paramString ()
void	updateUI ()

## JRadioButtonMenuItem 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JMenuItem
|
+-- javax.swing.JRadioButtonMenuItem

public class JRadioButtonMenuItem
    extends JMenuItem
    implements Accessible
```

这个类表示加到 Swing 菜单上的单选按钮菜单项。单选按钮菜单项看起来类似于单选按钮，并且必须用 ButtonGroup 对象来组合它们。这些菜单项每次只能选择一个。

### Inner Class

```
protected class JRadioButtonMenuItem.AccessibleRadioButtonMenuItem
```

### Constructors

```
JRadioButtonMenuItem ()
JRadioButtonMenuItem (Icon icon)
JRadioButtonMenuItem (Icon icon, boolean selected)
JRadioButtonMenuItem (String text)
JRadioButtonMenuItem (String text, boolean b)
JRadioButtonMenuItem (String text, Icon icon)
JRadioButtonMenuItem (String text, Icon icon, boolean selected)
```

### Methods

AccessibleContext	getAccessibleContext ()
String	getUIClassID ()
protected void	init (String text, Icon icon)

```

protected String paramString()
void requestFocus()
void updateUI()

```

## JRootPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JRootPane
public class JRootPane
    extends JComponent
    implements Accessible

```

这个类表示 Swing 根窗格。根窗格基本上就是一个内容窗格、玻璃窗格、分层窗格或菜单条的打包类。给予根窗格的任何职责都能授权给一个下层的类。根窗格存在于 Swing 小程序、框架、对话框和窗口中。

### Inner Classes

```

protected class JRootPane.AccessibleJRootPane
protected class JRootPane.RootLayout

```

### Fields

protected Container	contentPane
protected JButton	defaultButton
protected javax.swing.JRootPane.DefaultAction	defaultPressAction
protected javax.swing.JRootPane.DefaultAction	defaultReleaseAction
protected Component	glassPane
protected JLayeredPane	layeredPane
protected JMenuBar	menuBar

### Constructor

```
JRootPane()
```

### Methods

protected void	addImpl(Component comp, Object constraints, int index)
void	addNotify()
protected Container	createContentPane()
protected Component	createGlassPane()
protected JLayeredPane	createLayeredPane()
protected LayoutManager	createRootLayout()
AccessibleContext	getAccessibleContext()
Container	getContentPane()

```

JButton           getDefaultButton ()
Component        getGlassPane ()
JMenuBar         getJMenuBar ()
JLayeredPane     getLayeredPane ()
JMenuBar         getMenuBar ()
boolean          isFocusCycleRoot ()
boolean          isValidateRoot ()
protected String paramString ()
void             removeNotify ()
void             setContentPane (Container content)
void             setDefaultButton (JButton defaultButton)
void             setGlassPane (Component glass)
void             setJMenuBar (JMenuBar menu)
void             setLayeredPane (JLayeredPane layered)
void             setMenuBar (JMenuBar menu)
// Deprecated. Use the method
setJMenuBar (JMenuBar menu) .

```

## JScrollBar 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JScrollBar
public class JScrollBar
    extends JComponent
    implements Adjustable, Accessible

```

这个类表示 Swing 滚动条。不过，经常用到的是滚动窗格，而不是滚动条。

### Inner Class

```
protected class JScrollBar.AccessibleJScrollBar
```

### Fields

protected int	blockIncrement
protected BoundedRangeModel	model
protected int	orientation
protected int	unitIncrement

### Constructors

```
JScrollBar ()
JScrollBar (int orientation)
JScrollBar (int orientation, int value, int extent, int min, int max)
```

## Methods

```

void           addAdjustmentListener (AdjustmentListener l)
protected void fireAdjustmentValueChanged (int id, int type, int value)
AccessibleContext getAccessibleContext ()
int            getBlockIncrement ()
int            getBlockIncrement (int direction)
int            getMaximum ()
Dimension      getMaximumSize ()
int            getMinimum ()
Dimension      getMinimumSize ()
BoundedRangeModel getModel ()
int            getOrientation ()
ScrollBarUI    getUI ()
String          getUIClassID ()
int            getUnitIncrement ()
int            getUnitIncrement (int direction)
int            getValue ()
boolean         getValueIsAdjusting ()
int            getVisibleAmount ()
protected String paramString ()
void           removeAdjustmentListener (AdjustmentListener l)
void           setBlockIncrement (int blockIncrement)
void           setEnabled (boolean x)
void           setMaximum (int maximum)
void           setMinimum (int minimum)
void           setModel (BoundedRangeModel newModel)
void           setOrientation (int orientation)
void           setUnitIncrement (int unitIncrement)
void           setValue (int value)
void           setValueIsAdjusting (boolean b)
void           setValues (int newValue, int newExtent, int
                           newMin, int newMax)
void           setVisibleAmount (int extent)
void           updateUI ()

```

## JScrollPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JScrollPane

public class JScrollPane
    extends JComponent
    implements ScrollPaneConstants, Accessible

```

这个类表示 Swing 滚动窗格。滚动窗格是一个现成的容器，它支持可选的水平方向或垂直方向上的滚动条、视口、列标题和行标题以及其他位于滚动窗格四个角上的组件。

### Inner Classes

```
protected class JScrollPane.AccessibleJScrollPane
protected class JScrollPane.ScrollBar
```

### Fields

```
protected JViewport columnHeader
protected JScrollBar horizontalScrollBar
protected int horizontalScrollBarPolicy
protected Component lowerLeft
protected Component lowerRight
protected JViewport rowHeader
protected Component upperLeft
protected Component upperRight
protected JScrollBar verticalScrollBar
protected int verticalScrollBarPolicy
protected JViewport viewport
```

### Constructors

```
JScrollPane ()
JScrollPane (Component view)
JScrollPane (Component view, int vsbPolicy, int hsbPolicy)
JScrollPane (int vsbPolicy, int hsbPolicy)
```

### Methods

```
JScrollBar createHorizontalScrollBar ()
JScrollBar createVerticalScrollBar ()
protected JViewport createViewport ()
AccessibleContext getAccessibleContext ()
JViewport getColumnHeader ()
Component getCorner (String key)
JScrollBar getHorizontalScrollBar ()
int getHorizontalScrollBarPolicy ()
JViewport getRowHeader ()
ScrollPaneUI getUI ()
String getUIClassID ()
JScrollBar getVerticalScrollBar ()
int getVerticalScrollBarPolicy ()
JViewport getViewport ()
Border getViewportBorder ()
Rectangle getViewportBorderBounds ()
boolean isOpaque ()
```

```

boolean           isValidRoot ()
protected String paramString ()
void              setColumnHeader (JViewport columnHeader)
void              setColumnHeaderView (Component view)
void              setCorner (String key, Component corner)
void              setHorizontalScrollBar (JScrollBar horizontalScrollBar)
void              setHorizontalScrollBarPolicy (int policy)
void              setLayout (LayoutManager layout)
void              setRowHeader (JViewport rowHeader)
void              setRowHeaderView (Component view)
void              setUI (ScrollPaneUI ui)
void              setVerticalScrollBar (JScrollBar verticalScrollBar)
void              setVerticalScrollBarPolicy (int policy)
void              setViewport (JViewport viewport)
void              setViewportBorder (Border viewportBorder)
void              setViewportView (Component view)
void              updateUI ()

```

## JSeparator 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JSeparator

```

```

public class JSeparator
    extends JComponent
    implements SwingConstants, Accessible

```

这个类表示用于 Swing 菜单的分隔线。分隔线用来逻辑地组合菜单中的菜单项。

### Inner Class

```
protected class JSeparator.AccessibleJSeparator
```

### Constructors

```

JSeparator ()
JSeparator (int orientation)

```

### Methods

```

AccessibleContext getAccessibleContext ()
int             getOrientation ()
SeparatorUI      getUI ()

```

---

```

String           getUIClassID ()
boolean          isFocusTraversable ()
protected String paramString ()
void             setOrientation (int orientation)
void             setUI (SeparatorUI ui)
void             updateUI ()

```

## JSlider 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JSlider

public class JSlider
    extends JComponent
    implements SwingConstants, Accessible

```

这个类表示 Swing 滑动块。滑动块允许用户在最小值和最大值范围内选择一个特定的值。

### Inner Classes

```
protected class JSlider.AccessibleJSlider
```

### Fields

protected ChangeEvent	changeEvent
protected ChangeListener	changeListener
protected int	majorTickSpacing
protected int	minorTickSpacing
protected int	orientation
protected BoundedRangeModel	sliderModel
protected boolean	snapToTicks

### Constructors

```

JSlider ()
JSlider (BoundedRangeModel brm)
JSlider (int orientation)
JSlider (int min, int max)
JSlider (int min, int max, int value)
JSlider (int orientation, int min, int max, int value)

```

### Methods

void	addChangeListener (ChangeListener l)
------	--------------------------------------

```
protected ChangeListener    createChangeListener ()
Hashtable                 createStandardLabels (int increment)
Hashtable                 createStandardLabels (int increment, int start)
protected void              fireStateChanged ()
AccessibleContext          getAccessibleContext ()
int                       getExtent ()
boolean                   getInverted ()
Dictionary                getLabelTable ()
int                       getMajorTickSpacing ()
int                       getMaximum ()
int                       getMinimum ()
int                       getMinorTickSpacing ()
BoundedRangeModel         getModel ()
int                       getOrientation ()
boolean                   getPaintLabels ()
boolean                   getPaintTicks ()
boolean                   getPaintTrack ()
boolean                   getSnapToTicks ()
SliderUI                  getUI ()
String                     getUIClassID ()
int                       getValue ()
boolean                   getValueIsAdjusting ()
protected String           paramString ()
void                      removeChangeListener (ChangeListener l)
void                      setExtent (int extent)
void                      setInverted (boolean b)
void                      setLabelTable (Dictionary labels)
void                      setMajorTickSpacing (int n)
void                      setMaximum (int maximum)
void                      setMinimum (int minimum)
void                      setMinorTickSpacing (int n)
void                      setModel (BoundedRangeModel newModel)
void                      setOrientation (int orientation)
void                      setPaintLabels (boolean b)
void                      setPaintTicks (boolean b)
void                      setPaintTrack (boolean b)
void                      setSnapToTicks (boolean b)
void                      setUI (SliderUI ui)
void                      setValue (int n)
void                      setValueIsAdjusting (boolean b)
protected void             updateLabelUIs ()
void                      updateUI ()
```

## JSplitPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JSplitPane

public class JSplitPane
    extends JComponent
    implements Accessible

```

这个类表示 Swing 分割窗格。分割窗格用绘图方式来分隔一个容器中的两个子组件。

### Inner Class

```
protected class JSplitPane.AccessibleJSplitPane
```

### Fields

static String	BOTTOM
static String	CONTINUOUS_LAYOUT_PROPERTY
protected boolean	continuousLayout
static String	DIVIDER
static String	DIVIDER_SIZE_PROPERTY
protected int	dividerSize
static int	HORIZONTAL_SPLIT
static String	LAST_DIVIDER_LOCATION_PROPERTY
protected int	lastDividerLocation
static String	LEFT
protected Component	leftComponent
static String	ONE_TOUCH_EXPANDABLE_PROPERTY
protected boolean	oneTouchExpandable
protected int	orientation
static String	ORIENTATION_PROPERTY
static String	RIGHT
protected Component	rightComponent
static String	TOP
static int	VERTICAL_SPLIT

### Constructors

```

JSplitPane ()
JSplitPane (int newOrientation)
JSplitPane (int newOrientation, boolean newContinuousLayout)
JSplitPane (int newOrientation, boolean newContinuousLayout,

```

```
Component newLeftComponent, Component newRightComponent)
JSplitPane (int newOrientation, Component newLeftComponent,
Component newRightComponent)
```

## Methods

protected void	addImpl (Component comp, Object constraints, int index)
AccessibleContext	getAccessibleContext ()
Component	getBottomComponent ()
int	getDividerLocation ()
int	getDividerSize ()
int	getLastDividerLocation ()
Component	getLeftComponent ()
int	getMaximumDividerLocation ()
int	getMinimumDividerLocation ()
int	getOrientation ()
Component	getRightComponent ()
Component	getTopComponent ()
SplitPaneUI	getUI ()
String	getUIClassID ()
boolean	isContinuousLayout ()
boolean	isOneTouchExpandable ()
protected void	paintChildren (Graphics g)
protected String	paramString ()
void	remove (Component component)
void	remove (int index)
void	removeAll ()
void	resetToPreferredSizes ()
void	setBottomComponent (Component comp)
void	setContinuousLayout (boolean newContinuousLayout)
void	setDividerLocation (double proportionalLocation)
void	setDividerLocation (int location)
void	setDividerSize (int newSize)
void	setLastDividerLocation (int newLastLocation)
void	setLeftComponent (Component comp)
void	setOneTouchExpandable (boolean newValue)
void	setOrientation (int orientation)
void	setRightComponent (Component comp)
void	setTopComponent (Component comp)
void	setUI (SplitPaneUI ui)
void	updateUI ()

## JTabbedPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JTabbedPane

public class JTabbedPane
    extends JComponent
    implements Serializable, Accessible, SwingConstants

```

这个类表示 Swing 的标签窗格。通过单击标签，可以同一显示区域内显示出一个的新窗格。每个窗格都可以包含自己的子组件集合。

### Inner Classes

```

protected class JTabbedPane.AccessibleJTabbedPane
protected class JTabbedPane.ModelListener

```

### Fields

```

protected ChangeEvent changeEvent
protected ChangeListener changeListener
protected SingleSelectionModel model
protected int tabPlacement

```

### Constructors

```

JTabbedPane()
JTabbedPane(int tabPlacement)

```

### Methods

Component	add (Component component)
Component	add (Component component, int index)
void	add (Component component, Object constraints)
void	add (Component component, Object constraints, int index)
Component	add (String title, Component component)
void	addChangeListener (ChangeListener l)
void	addTab (String title, Component component)
void	addTab (String title, Icon icon, Component component)
void	addTab (String title, Icon icon, Component component, String tip)
protected ChangeListener	createChangeListener ()
protected void	fireStateChanged ()

```
AccessibleContext      getAccessibleContext()
Color                 getBackgroundAt (int index)
Rectangle            getBoundsAt (int index)
Component            getComponentAt (int index)
Icon                 getDisabledIconAt (int index)
Color                getForegroundAt (int index)
Icon                 getIconAt (int index)
SingleSelectionModel getModel ()
Component            getSelectedComponent ()
int                  getSelectedIndex ()
int                  getTabCount ()
int                  getTabPlacement ()
int                  getTabRunCount ()
String               getTitleAt (int index)
String               getToolTipText (MouseEvent event)
TabbedPaneUI         getUI ()
String               getUIClassID ()
int                  indexOfComponent (Component component)
int                  indexOfTab (Icon icon)
int                  indexOfTab (String title)
void                insertTab (String title, Icon icon,
                           Component component, String tip,
                           int index)
boolean              isEnabledAt (int index)
protected String     paramString ()
void                remove (Component component)
void                removeAll ()
void                removeChangeListener (ChangeListener l)
void                removeTabAt (int index)
void                setBackgroundAt (int index, Color background)
void                setComponentAt (int index, Component component)
void                setDisabledIconAt (int index, Icon disabledIcon)
void                setEnabledAt (int index, boolean enabled)
void                setForegroundAt (int index, Color foreground)
void                setIconAt (int index, Icon icon)
void                setModel (SingleSelectionModel model)
void                setSelectedComponent (Component c)
void                setSelectedIndex (int index)
void                setTabPlacement (int tabPlacement)
void                setTitleAt (int index, String title)
void                setUI (TabbedPaneUI ui)
void                updateUI ()
```

## JTable 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JTable
public class JTable
    extends JComponent
    implements TableModelListener, Scrollable,
    TableColumnModelListener, ListSelectionListener,
    CellEditorListener, Accessible

```

这个类表示 Swing 表格。表格可以看作是二维的列表。

### Inner Class

```
protected class JTable.AccessibleJTable
```

### Fields

static int	AJTC_RESIZE_ALL_COLUMNS
static int	AJTC_RESIZE_LAST_COLUMN
static int	AJTC_RESIZE_NEXT_COLUMN
static int	AJTC_RESIZE_OFF
static int	AJTC_RESIZE_SUBSEQUENT_COLUMNS
protected boolean	autoCreateColumnsFromModel
protected int	autoResizeMode
protected TableCellEditor	cellEditor
protected boolean	cellSelectionEnabled
protected TableColumnModel	columnModel
protected TableModel	dataModel
protected Hashtable	defaultEditorsByColumnClass
protected Hashtable	defaultRenderersByColumnClass
protected int	editingColumn
protected int	editingRow
protected Component	editorComp
protected Color	gridColor
protected Dimension	preferredViewportSize
protected int	rowHeight
protected int	rowMargin
protected boolean	rowSelectionAllowed
protected Color	selectionBackground
protected Color	selectionForeground
protected ListSelectionModel	selectionModel

protected boolean	showHorizontalLines
protected boolean	showVerticalLines
protected JTableHeader	tableHeader

## Constructors

```
JTable ()
JTable (int numRows, int numColumns)
JTable (Object [] [] rowData, Object [] columnNames)
JTable (TableModel dm)
JTable (TableModel dm, TableColumnModel cm)
JTable (TableModel dm, TableColumnModel cm, ListSelectionModel sm)
JTable (Vector rowData, Vector columnNames)
```

## Methods

void	addColumn (TableColumn aColumn)
void	addColumnSelectionInterval (int index0, int index1)
void	addNotify ()
void	addRowSelectionInterval (int index0, int index1)
void	clearSelection ()
void	columnAdded (TableColumnModelEvent e)
int	columnAtPoint (Point point)
void	columnMarginChanged (ChangeEvent e)
void	columnMoved (TableColumnModelEvent e)
void	columnRemoved (TableColumnModelEvent e)
void	columnSelectionChanged (ListSelectionEvent e)
protected void	configureEnclosingScrollPane ()
int	convertColumnIndexToModel (int viewColumnIndex)
int	convertColumnIndexToView (int modelColumnIndex)
protected TableColumnModel	createDefaultColumnModel ()
void	createDefaultColumnsFromModel ()
protected TableModel	createDefaultDataModel ()
protected void	createDefaultEditors ()
protected void	createDefaultRenderers ()
protected ListSelectionModel	createDefaultSelectionModel ()
protected JTableHeader	createDefaultTableHeader ()
static JScrollPane	createScrollPaneForTable (JTable aTable) // Deprecated. Use the method JScrollPane (aTable) .
boolean	editCellAt (int row, int column)
boolean	editCellAt (int row, int column, EventObject e)
void	editingCanceled (ChangeEvent e)
void	editingStopped (ChangeEvent e)
AccessibleContext	getAccessibleContext ()

boolean	getAutoCreateColumnsFromModel ()
int	getAutoResizeMode ()
TableCellEditor	getCellEditor ()
TableCellEditor	getCellEditor (int row, int column)
Rectangle	getCellRect (int row, int column, boolean includeSpacing)
TableCellRenderer	getCellRenderer (int row, int column)
boolean	getCellSelectionEnabled ()
TableColumn	getColumn (Object identifier)
Class	getColumnClass (int column)
int	getColumnCount ()
TableColumnModel	getColumnModel ()
String	getColumnName (int column)
boolean	getColumnSelectionAllowed ()
TableCellEditor	getDefaultEditor (Class columnClass)
TableCellRenderer	getDefaultRenderer (Class columnClass)
int	getEditingColumn ()
int	getEditingRow ()
Component	getEditorComponent ()
Color	getGridColor ()
Dimension	getInterCellSpacing ()
TableModel	getModel ()
Dimension	getPreferredScrollableViewportSize ()
int	getRowCount ()
int	getRowHeight ()
int	getRowMargin ()
boolean	getRowSelectionAllowed ()
int	getScrollableBlockIncrement (Rectangle visibleRect, int orientation, int direction)
boolean	getScrollableTracksViewportHeight ()
boolean	getScrollableTracksViewportWidth ()
int	getScrollableUnitIncrement (Rectangle visibleRect, int orientation, int direction)
int	getSelectedColumn ()
int	getSelectedColumnCount ()
int []	getSelectedColumns ()
int	getSelectedRow ()
int	getSelectedRowCount ()
int []	getSelectedRows ()
Color	getSelectionBackground ()
Color	getSelectionForeground ()
ListSelectionModel	getSelectionModel ()
boolean	getShowHorizontalLines ()

```
boolean           getShowVerticalLines ()
JTableHeader      getTableHeader ()
String            getToolTipText (MouseEvent event)
TableUI           getUIT ()
String            getClassID ()
Object            getValueAt (int row, int column)
protected void    initializeLocalVars ()
boolean           isCellEditable (int row, int column)
boolean           isCellSelected (int row, int column)
boolean           isColumnSelected (int column)
boolean           isEditing ()
boolean           isManagingFocus ()
boolean           isRowSelected (int row)
void              moveColumn (int column, int targetColumn)
protected String  paramString ()
Component         prepareEditor (TableCellEditor
                           editor, int row, int column)
Component         prepareRenderer (TableCellRenderer
                           renderer, int row, int column)
void              removeColumn (TableColumn aColumn)
void              removeColumnSelectionInterval (int
                           index0, int index1)
void              removeEditor ()
void              removeRowSelectionInterval (int
                           index0, int index1)
void              reshape (int x, int y, int width, int height)
protected void    resizeAndRepaint ()
int               rowAtPoint (Point point)
void              selectAll ()
void              setAutoCreateColumnsFromModel
                  (boolean createColumns)
void              setAutoResizeMode (int mode)
void              setCellEditor (TableCellEditor anEditor)
void              setCellSelectionEnabled (boolean flag)
void              setColumnModel (TableColumnModel newModel)
void              setColumnSelectionAllowed (boolean flag)
void              setColumnSelectionInterval (int
                           index0, int index1)
void              setDefaultEditor (Class
                           columnClass, TableCellEditor
                           editor)
void              setDefaultRenderer (Class
                           columnClass, TableCellRenderer
                           renderer)
void              setEditingColumn (int aColumn)
```

```

void setEditingRow (int aRow)
void setGridColor (Color newColor)
void setIntercellSpacing (Dimension newSpacing)
void setModel (TableModel newModel)
void setPreferredScrollableViewportSize
(Dimension size)
void setRowHeight (int newHeight)
void setRowMargin (int rowMargin)
void setRowSelectionAllowed (boolean flag)
void setRowSelectionInterval (int index0, int index1)
void setSelectionBackground
(Color selectionBackground)
void setSelectionForeground
(Color selectionForeground)
void setSelectionMode (int selectionMode)
void setSelectionModel
(ListSelectionModel newModel)
void setShowGrid (boolean b)
void setShowHorizontalLines (boolean b)
void setShowVerticalLines (boolean b)
void setTableHeader (TableHeader newHeader)
void setUI (TableUI ui)
void setValueAt (Object aValue, int row, int column)
void sizeColumnsToFit (boolean lastColumnOnly)
// Deprecated. Use the method
void sizeColumnsToFit (int).
void sizeColumnsToFit (int resizingColumn)
void tableChanged (TableModelEvent e)
void updateUI ()
void valueChanged (ListSelectionEvent e)

```

## JTextArea 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
|
+-- javax.swing.JTextArea
public class JTextArea
    extends JTextComponent

```

这个类表示 Swing 文本区域。文本区域支持多行纯文本 (plain text) 的显示和编辑。

### Inner Class

protected class JTextArea.AccessibleJTextArea

### Constructors

```
JTextArea ()  
JTextArea (Document doc)  
JTextArea (Document doc, String text, int rows, int columns)  
JTextArea (int rows, int columns)  
JTextArea (String text)  
JTextArea (String text, int rows, int columns)
```

### Methods

void	append (String str)
protected Document	createDefaultModel ()
AccessibleContext	getAccessibleContext ()
int	getColumns ()
protected int	getColumnWidth ()
int	getLineCount ()
int	getLineEndOffset (int line)
int	getLineOffset (int offset)
int	getLineStartOffset (int line)
boolean	getLineWrap ()
Dimension	getPreferredSize (ViewPort viewPort)
Dimension	getPreferredSize ()
protected int	getRowHeight ()
int	getRows ()
boolean	getScrollableTracksViewportWidth ()
int	getScrollableUnitIncrement (Rectangle visibleRect, int orientation, int direction)
int	getTabSize ()
String	getUIClassID ()
boolean	getWrapStyleWord ()
void	insert (String str, int pos)
boolean	isManagingFocus ()
protected String	paramString ()
protected void	processComponentKeyEvent (KeyEvent e)
void	replaceRange (String str, int start, int end)
void	setColumns (int columns)
void	setFont (Font f)
void	setLineWrap (boolean wrap)
void	setRows (int rows)
void	setTabSize (int size)

```
void setWrapStyleWord (boolean word)
```

## JTextField 类

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
|
+-- javax.swing.JTextField

public class JTextField
    extends JTextComponent
    implements SwingConstants
```

这个类表示 Swing 支持的轻量级文本字段。该组件支持单行纯文本的显示和编辑。

### Inner Class

```
protected class JTextField.AccessibleJTextField
```

### Fields

```
static String notifyAction
```

### Constructors

```
JTextField ()
JTextField (Document doc, String text, int columns)
JTextField (int columns)
JTextField (String text)
JTextField (String text, int columns)
```

### Methods

void	addActionListener (ActionListener l)
protected Document	createDefaultModel ()
protected void	fireActionPerformed ()
AccessibleContext	getAccessibleContext ()
Action []	getActions ()
int	getColumns ()
protected int	getColumnWidth ()
int	getHorizontalAlignment ()
BoundedRangeModel	getHorizontalVisibility ()
Dimension	getPreferredSize ()
int	getScrollOffset ()
String	getUIClassID ()
boolean	isValidRoot ()
protected String	paramString ()

```

void           postActionEvent ()
void           removeActionListener (ActionListener l)
void           scrollRectToVisible (Rectangle r)
void           setActionCommand (String command)
void           setColumns (int columns)
void           setFont (Font f)
void           setHorizontalAlignment (int alignment)
void           setScrollOffset (int scrollOffset)

```

## JTextPane 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
|
+-- javax.swing.JEditorPane
|
+-- javax.swing.JTextPane

public class JTextPane
    extends JEditorPane

```

这个类表示 Swing 支持的文本窗格。文本窗格支持多行并具有特定样式文本的显示和编辑。

### Constructors

```

JTextPane ()
JTextPane (StyledDocument doc)

```

### Methods

Style	addStyle (String nm, Style parent)
protected EditorKit	createDefaultEditorKit ()
AttributeSet	getCharacterAttributes ()
MutableAttributeSet	getInputAttributes ()
Style	getLogicalStyle ()
AttributeSet	getParagraphAttributes ()
boolean	getScrollableTracksViewportWidth ()
Style	getStyle (String nm)
StyledDocument	getStyledDocument ()
protected StyledEditorKit	getStyledEditorKit ()
String	getUIClassID ()
void	insertComponent (Component c)

---

```

void           insertIcon (Icon g)
protected String paramString ()
void           removeStyle (String nm)
void           replaceSelection (String content)
void           setCharacterAttributes (AttributeSet attr, boolean
replace)
void           setDocument (Document doc)
void           setEditorKit (EditorKit kit)
void           setLogicalStyle (Style s)
void           setParagraphAttributes (AttributeSet attr, boolean
replace)
void           setStyledDocument (StyledDocument doc)

```

## JToggleButton 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.AbstractButton
|
+-- javax.swing.JToggleButton
public class JToggleButton
    extends AbstractButton
    implements Accessible

```

这个类表示 Swing 支持的切换按钮。当单击切换按钮时，它会保持被按下的状态；当再次单击它时，它就会弹出到原来的位置。

### Inner Class

```

protected class   JToggleButton.AccessibleJToggleButton
static class     JToggleButton.ToggleButtonModel

```

### Constructors

```

JToggleButton ()
JToggleButton (Icon icon)
JToggleButton (Icon icon, boolean selected)
JToggleButton (String text)
JToggleButton (String text, boolean selected)
JToggleButton (String text, Icon icon)
JToggleButton (String text, Icon icon, boolean selected)

```

### Methods

```

AccessibleContext  getAccessibleContext ()

```

```

String           getUIClassID ()
protected String paramString ()
void            updateUI ()

```

## JToggleButton.ToggleButtonModel 类

```

java.lang.Object
|
+-- javax.swing.DefaultButtonModel
|
+-- javax.swing.JToggleButton.ToggleButtonModel
public static class JToggleButton.ToggleButtonModel
    extends DefaultButtonModel

```

这个类表示 Swing 切换按钮所使用的按钮模式，该按钮模式包含按钮的状态。

### Constructor

```
JToggleButton.ToggleButtonModel ()
```

### Methods

```

boolean   isSelected ()
void      setPressed (boolean b)
void      setSelected (boolean b)

```

## JToolBar 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JToolBar
public class JToolBar
    extends JComponent
    implements SwingConstants, Accessible

```

这个类表示 Swing 中使用的工具条。工具条包括诸如按钮、组合框等组件，从而可以很容易地访问频繁使用的控件。

### Inner Class

```

protected class JToolBar.AccessibleJToolBar
static class JToolBar.Separator

```

### Constructors

```

JToolBar ()
JToolBar (int orientation)

```

**Methods**

JButton	add (Action a)
protected void	addImpl (Component comp, Object constraints, int index)
void	addSeparator ()
void	addSeparator (Dimension size)
protected PropertyChangeListener	createActionChange Listener (JButton b)
AccessibleContext	getAccessibleContext ()
Component	getComponentAtIndex (int i)
int	getComponentIndex (Component c)
Insets	getMargin ()
int	getOrientation ()
ToolBarUI	getUI ()
String	getUIClassID ()
boolean	isBorderPainted ()
boolean	isFloatable ()
protected void	paintBorder (Graphics g)
protected String	paramString ()
void	remove (Component comp)
void	setBorderPainted (boolean b)
void	setFloatable (boolean b)
void	setMargin (Insets m)
void	setOrientation (int o)
void	setUI (ToolBarUI ui)
void	updateUI ()

**JToolBar.Separator 类**

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JSeparator
|
+-- javax.swing.JToolBar.Separator
public static class JToolBar.Separator
    extends JSeparator

```

这个类表示特定工具条的分隔线，此工具条组件可以按逻辑组来分别显示。

## Constructors

```
JToolBar.Separator()
JToolBar.Separator(Dimension size)
```

## Methods

Dimension	getMaximumSize()
Dimension	getMinimumSize()
Dimension	getPreferredSize()
Dimension	getSeparatorSize()
String	getUIClassID()
void	setSeparatorSize(Dimension size)

## JToolTip 类

```
java.lang.Object
  +-- java.awt.Component
    +-- java.awt.Container
      +-- javax.swing.JComponent
        +-- javax.swing.JToolTip
public class JToolTip
  extends JComponent
  implements Accessible
```

这个类表示 Swing 中的工具提示。一般情况下，可以使用 JComponent 类中的 setToolTipText() 方法。不过，这个类对创建定制的工具提示很有用。

## Inner Class

```
protected class JToolTip.AccessibleJToolTip
```

## Constructor

```
JToolTip()
```

## Methods

AccessibleContext	getAccessibleContext()
JComponent	getComponent()
String	getTipText()
ToolTipUI	getUI()
String	getUIClassID()
protected String	paramString()
void	setComponent(JComponent c)
void	setTipText(String tipText)
void	updateUI()

## JTree 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JTree
public class JTree
extends JComponent
implements Scrollable, Accessible

```

这个类表示 Swing 树。树对象通常用来显示层次结构数据，如计算机系统的目录。

### Inner Classes

protected class	JTree.AccessibleJTree
static class	JTree.DynamicUtilTreeNode
protected static class	JTree.EmptySelectionModel
protected class	JTree.TreeModelHandler
protected class	JTree.TreeSelectionRedirector

### Fields

static String	CELL_EDITOR_PROPERTY
static String	CELL_RENDERER_PROPERTY
protected TreeCellEditor	cellEditor
protected TreeCellRenderer	cellRenderer
protected boolean	editable
static String	EDITABLE_PROPERTY
static String	INVOKES_STOP_CELL
	EDITING_PROPERTY
protected boolean	invokesStopCellEditing
static String	LARGE_MODEL_PROPERTY
protected boolean	largeModel
static String	ROOT_VISIBLE_PROPERTY
protected boolean	rootVisible
static String	ROW_HEIGHT_PROPERTY
protected int	rowHeight
static String	SCROLLS_ON_EXPAND_PROPERTY
protected boolean	scrollsOnExpand
static String	SELECTION_MODEL_PROPERTY
protected TreeSelectionModel	selectionModel

```

protected JTree.TreeSelectionRedirector selectionRedirector
static String SHOWS_ROOT_HANDLES
    PROPERTY

protected boolean showsRootHandles
protected int toggleClickCount
static String TREE_MODEL_PROPERTY
protected TreeModel treeModel
protected TreeModelListener treeModelListener
static String VISIBLE_ROW_COUNT
    PROPERTY

protected int visibleRowCount

```

### Constructors

```

JTree ()
JTree (Hashtable value)
JTree (Object [] value)
JTree (TreeModel newModel)
JTree (TreeNode root)
JTree (TreeNode root, boolean asksAllowsChildren)
JTree (Vector value)

```

### Methods

```

void addSelectionInterval (int index0, int index1)
void addSelectionPath (TreePath path)
void addSelectionPaths (TreePath [] paths)
void addSelectionRow (int row)
void addSelectionRows (int [] rows)
void addTreeExpansionListener
    (TreeExpansionListener tel)
void addTreeSelectionListener
    (TreeSelectionListener ts1)
void addTreeWillExpandListener
    (TreeWillExpandListener tel)
void cancelEditing ()
void clearSelection ()
protected void clearToggledPaths ()
void collapsePath (TreePath path)
void collapseRow (int row)
String convertValueToText (Object value,
    boolean selected, boolean
    expanded, boolean leaf, int row,
    boolean hasFocus)
protected static TreeModel createTreeModel (Object value)
protected TreeModelListener createTreeModelListener ()

```

```
void expandPath (TreePath path)
void expandRow (int row)
void fireTreeCollapsed (TreePath path)
void fireTreeExpanded (TreePath path)
void fireTreeWillCollapse (TreePath path)
void fireTreeWillExpand (TreePath path)
protected void fireValueChanged
    (TreeSelectionEvent e)
AccessibleContext getAccessibleContext ()
TreeCellEditor getCellEditor ()
TreeCellRenderer getCellRenderer ()
TreePath getClosestPathForLocation (int x, int y)
int getClosestRowForLocation (int x, int y)
protected static TreeModel getDefaultTreeModel ()
protected Enumeration getDescendantToggledPaths (TreePath parent)
TreePath getEditingPath ()
Enumeration getExpandedDescendants (TreePath parent)
boolean getInvokesStopCellEditing ()
Object getLastSelectedPathComponent ()
TreePath getLeadSelectionPath ()
int getLeadSelectionRow ()
int getMaxSelectionRow ()
int getMinSelectionRow ()
TreeModel getModel ()
protected TreePath []
Rectangle getPathBounds (TreePath path)
TreePath getPathForLocation (int x, int y)
TreePath getPathForRow (int row)
Dimension getPreferredScrollableViewport
    Size ()
Rectangle getRowBounds (int row)
int getCount ()
int getRowForLocation (int x, int y)
int getRowForPath (TreePath path)
int getRowHeight ()
int getScrollableBlockIncrement
    (Rectangle visibleRect, int
     orientation, int direction)
boolean getScrollableTracksViewport
    Height ()
boolean getScrollableTracksViewportWidth ()
int getScrollableUnitIncrement
    (Rectangle visibleRect,
```

```
        orientation, int direction)
boolean           getScrollsOnExpand ()
int               getSelectionCount ()
TreeSelectionModel getSelectionModel ()
TreePath          getSelectionPath ()
TreePath []       getSelectionPaths ()
int []            getSelectionRows ()
boolean           getShowsRootHandles ()
String            getToolTipText (MouseEvent event)
TreeUI            getUI ()
String            getUIClassID ()
int               getVisibleRowCount ()
boolean           hasBeenExpanded (TreePath path)
boolean           isCollapsed (int row)
boolean           isCollapsed (TreePath path)
boolean           isEditable ()
boolean           isEditing ()
boolean           isExpanded (int row)
boolean           isExpanded (TreePath path)
boolean           isFixedRowHeight ()
boolean           isLargeModel ()
boolean           isPathEditable (TreePath path)
boolean           isPathSelected (TreePath path)
boolean           isRootVisible ()
boolean           isRowSelected (int row)
boolean           isSelectionEmpty ()
boolean           isVisible (TreePath path)
void              makeVisible (TreePath path)
protected String  paramString ()
protected void    removeDescendantToggledPaths
                  (Enumeration toRemove)
void              removeSelectionInterval (int
                                         index0, int index1)
void              removeSelectionPath (TreePath path)
void              removeSelectionPaths (TreePath [] paths)
void              removeSelectionRow (int row)
void              removeSelectionRows (int [] rows)
void              removeTreeExpansionListener
                  (TreeExpansionListener tel)
void              removeTreeSelectionListener
                  (TreeSelectionListener tsl)
void              removeTreeWillExpandListener
                  (TreeWillExpandListener tel)
```

```

void scrollPathToVisible (TreePath path)
void scrollRowToVisible (int row)
void setCellEditor (TreeCellEditor cellEditor)
void setCellRenderer (TreeCellRenderer x)
void setEditable (boolean flag)
protected void setExpandedState (TreePath path, boolean state)
void setInvokesStopCellEditing (boolean newValue)
void setLargeModel (boolean newValue)
void setModel (TreeModel newModel)
void setRootVisible (boolean rootVisible)
void setRowHeight (int rowHeight)
void setScrollsOnExpand (boolean newValue)
void setSelectionInterval (int index0, int index1)
void setSelectionModel
    (TreeSelectionModel
selectionModel)
void setSelectionPath (TreePath path)
void setSelectionPaths (TreePath [] paths)
void setSelectionRow (int row)
void setSelectionRows (int [] rows)
void setShowsRootHandles (boolean newValue)
void setUI (TreeUI ui)
void setVisibleRowCount (int newCount)
void startEditingAtPath (TreePath path)
boolean stopEditing ()
void treeDidChange ()
void updateUI ()

```

## JTree.DynamicUtilTreeNode 类

```

java.lang.Object
|
+-- javax.swing.tree.DefaultMutableTreeNode
|
+-- javax.swing.JTree.DynamicUtilTreeNode
public static class JTree.DynamicUtilTreeNode
    extends DefaultMutableTreeNode

```

这个类表示一个实用工具，它能够动态地创建通过向量、数组、字符串或哈希表等数据结构所指定的树节点。

### Fields

protected Object	childValue
protected boolean	hasChildren
protected boolean	loadedChildren

**Constructor**

```
JTree.DynamicUtilTreeNode (Object value, Object children)
```

**Methods**

Enumeration	children ()
static void	createChildren (DefaultMutableTreeNode parent, Object children)
TreeNode	getChildAt (int index)
int	getChildCount ()
boolean	isLeaf ()
protected void	loadChildren ()

**JTree.EmptySelectionModel 类**

```
java.lang.Object
|
+-- javax.swing.tree.DefaultTreeSelectionModel
|
+-- javax.swing.JTree.EmptySelectionModel
protected static class JTree.EmptySelectionModel
    extends DefaultTreeSelectionModel
```

这个类表示一种 TreeSelectionModel，它不允许选择节点。

**Field**

```
protected static JTree.EmptySelectionModel sharedInstance
```

**Constructor**

```
protected JTree.EmptySelectionModel ()
```

**Methods**

void	addSelectionPaths (TreePath [] paths)
void	removeSelectionPaths (TreePath [] paths)
void	setSelectionPaths (TreePath [] paths)
static JTree.EmptySelectionModel	sharedInstance ()

**JViewport 类**

```
java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JViewport
public class JViewport
    extends JComponent
    implements Accessible
```

这个类表示视口，它可以用在滚动窗格里。视口是用来观看下面内容的显示窗口。通过使用滚动条使视口上下滚动来观察原来看不见的内容。

### Inner Classes

protected class	JViewport.AccessibleJViewport
protected class	JViewport.ViewListener

### Fields

protected boolean	backingStore
protected Image	backingStoreImage
protected boolean	isViewSizeSet
protected Point	lastPaintPosition
protected boolean	scrollUnderway

### Constructor

JViewport ()

### Methods

void	addChangeListener (ChangeListener l)
protected void	addImpl (Component child, Object constraints, int index)
protected boolean	computeBlit (int dx, int dy, Point blitFrom, Point blitTo, Dimension blitSize, Rectangle blitPaint)
protected LayoutManager	createLayoutManager ()
protected JViewport.ViewListener	createViewListener ()
protected void	fireStateChanged ()
AccessibleContext	getAccessibleContext ()
Dimension	getExtensionSize ()
Insets	getInsets ()
Insets	getInsets (Insets insets)
Component	getView ()
Point	getViewPosition ()
Rectangle	getViewRect ()
Dimension	getViewSize ()
boolean	isBackingStoreEnabled ()
boolean	isOptimizedDrawingEnabled ()
void	paint (Graphics g)
protected String	paramString ()
void	remove (Component child)
void	removeChangeListener (ChangeListener l)

```

void           repaint (long tm, int x, int
                      y, int w, int h)
void           reshape (int x, int y, int w, int h)
void           scrollRectToVisible (Rectangle contentRect)
void           setBackingStoreEnabled
              (boolean x)
void           setBorder (Border border)
void           setExtentSize (Dimension newExtent)
void           setView (Component view)
void           setViewPosition (Point p)
void           setViewSize (Dimension newSize)
Dimension     toViewCoordinates (Dimension size)
Point          toViewCoordinates (Point p)

```

## JWindow 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- java.awt.Window
|
+-- javax.swing.JWindow

public class JWindow
    extends Window
    implements Accessible, RootPaneContainer

```

这个类表示 Swing 窗口。Swing 窗口类似于 AWT 窗口，而且没有标题条、边界或关闭按钮和放大按钮。窗口支持内容窗格、玻璃窗格、分层窗格和菜单条。需要把子组件加在窗口下的内容窗格上，而不是直接加在窗口上。

### Inner Class

```
protected class JWindow.AccessibleJWindow
```

### Fields

protected AccessibleContext	accessibleContext
protected JRootPane	rootPane
protected boolean	rootPaneCheckingEnabled

### Constructors

```
JWindow ()
JWindow (Frame owner)
```

### Methods

protected void	addImpl (Component comp, Object constraints, int index)
JRootPane	createRootPane ()

---

AccessibleContext	getAccessibleContext ()
Container	getContentPane ()
Component	getGlassPane ()
JLayeredPane	getLayeredPane ()
JRootPane	getRootPane ()
protected boolean	isRootPaneCheckingEnabled ()
protected String	paramString ()
void	setContentPane (Container contentPane)
void	setGlassPane (Component glassPane)
void	setLayeredPane (JLayeredPane layeredPane)
void	setLayout (LayoutManager manager)
protected void	setRootPane (JRootPane root)
protected void	setRootPaneCheckingEnabled (boolean enabled)
protected void	windowInit ()

## KeyStroke 类

```
java.lang.Object
|
+-- javax.swing.KeyStroke
public class KeyStroke
    extends Object
    implements Serializable
```

这个类表示发生在键盘上的击键动作。击键对象包含必要的信息，如字符键、扫描码、功能键以及修饰键。

### Methods

boolean	equals (Object anObject)
char	getKeyChar ()
int	getKeyCode ()
static KeyStroke	getKeyStroke (char keyChar)
static KeyStroke	getKeyStroke (char keyChar, boolean onKeyRelease) //Deprecated. Use the method getKeyStroke (char)
static KeyStroke	getKeyStroke (int keyCode, int modifiers)
static KeyStroke	getKeyStroke (int keyCode, int modifiers, boolean onKeyRelease)
static KeyStroke	getKeyStroke (String representation)
static KeyStroke	getKeyStrokeForEvent (KeyEvent anEvent)
int	getModifiers ()
int	hashCode ()
boolean	isOnKeyRelease ()
String	toString ()

## LookAndFeel 类

```
java.lang.Object
|
+-- javax.swing.LookAndFeel
public abstract class LookAndFeel
    extends Object
```

这是一个抽象类，它表示组件的外观类型。这个类包含安装或卸载外观属性的功能，这些属性包括边界、颜色、字体等等。

### Constructor

```
LookAndFeel ()
```

### Methods

UIDefaults	getDefaults ()
abstract String	getDescription ()
abstract String	getID ()
abstract String	getName ()
void	initialize ()
static void	installBorder (JComponent c, String defaultBorderName)
static void	installColors (JComponent c, String defaultBgName, String defaultFgName)
static void	installColorsAndFont (JComponent c, String defaultBgName, String defaultFgName, String defaultFontName)
abstract boolean	isNativeLookAndFeel ()
abstract boolean	isSupportedLookAndFeel ()
static Object	makeIcon (Class baseClass, String gifFile)
static JTextComponent.KeyBinding []	makeKeyBindings (Object [] keyBindingList)
String	toString ()
void	uninitialize ()
static void	uninstallBorder (JComponent c)

## MenuSelectionManager 类

```
java.lang.Object
|
+-- javax.swing.MenuSelectionManager
public class MenuSelectionManager
    extends Object
```

这个类表示菜单选择管理器，该管理器负责管理在菜单层次结构中所做的选择。

### Fields

```
protected ChangeEvent      changeEvent
protected EventListenerList  listenerList
```

### Constructor

```
MenuSelectionManager ()
```

### Methods

```
void          addChangeListener (ChangeListener l)
void          clearSelectedPath ()
Component    componentForPoint (Component
                           source, Point sourcePoint)
static MenuSelectionManager defaultManager ()
protected void fireStateChanged ()
MenuItem []   getSelectedPath ()
boolean       isComponentPartOfCurrentMenu
              (Component c)
void          processKeyEvent (KeyEvent e)
void          processMouseEvent (MouseEvent event)
void          removeChangeListener
              (ChangeListener l)
void          setSelectedPath (MenuItem [] path)
```

## OverlayLayout 类

```
java.lang.Object
|
+-- javax.swing.OverlayLayout
public class OverlayLayout
    extends Object
    implements LayoutManager2, Serializable
```

这个类表示一个布局管理器，该布局管理器使用户能给组件分层。该布局由 Swing 组件自动使用。一般情况下，无需处理这个布局管理器。

### Constructor

```
OverlayLayout (Container target)
```

### Methods

```
void          addLayoutComponent (Component comp, Object constraints)
void          addLayoutComponent (String name, Component comp)
float         getLayoutAlignmentX (Container target)
float         getLayoutAlignmentY (Container target)
void          invalidateLayout (Container target)
void          layoutContainer (Container target)
```

```

Dimension    maximumLayoutSize (Container target)
Dimension    minimumLayoutSize (Container target)
Dimension    preferredLayoutSize (Container target)
void         removeLayoutComponent (Component comp)

```

## ProgressMonitor 类

```

java.lang.Object
|
+-- javax.swing.ProgressMonitor
public class ProgressMonitor
    extends Object

```

这个类表示进度监视器。进度监视器用来观察程序执行的进展情况。

### Constructors

```
ProgressMonitor (Component parentComponent, Object message, String note,
int min, int max)
```

### Methods

```

void        close ()
int         getMaximum ()
int         getMillisToDecideToPopup ()
int         getMillisToPopup ()
int         getMinimum ()
String      getNote ()
boolean     isCanceled ()
void         setMaximum (int m)
void         setMillisToDecideToPopup (int millisToDecideToPopup)
void         setMillisToPopup (int millisToPopup)
void         setMinimum (int m)
void         setNote (String note)
void         setProgress (int nv)

```

## ProgressMonitorInputStream 类

```

java.lang.Object
|
+-- java.io.InputStream
|
+-- java.io.FilterInputStream
|
+-- javax.swing.ProgressMonitorInputStream
public class ProgressMonitorInputStream
    extends FilterInputStream

```

这个类表示一个监视器，它专门用来观察输入流中数据的读出情况。

**Constructor**

```
ProgressMonitorInputStream (Component parentComponent, Object message,
InputStream in)
```

**Methods**

void	close ()
ProgressMonitor	getProgressMonitor ()
int	read ()
int	read (byte [] b)
int	read (byte [] b, int off, int len)
void	reset ()
long	skip (long n)

**RepaintManager 类**

```
java.lang.Object
|
+-- javax.swing.RepaintManager
public class RepaintManager
    extends Object
```

通过把多个请求分解为单一绘画操作，该类优化了重绘动作调用的次数，例如，在重绘树节点的情况下就可以实现这种优化。

**Constructors**

```
RepaintManager ()
```

**Methods**

void	addDirtyRegion (JComponent c, int x, int y, int w, int h)
void	addInvalidComponent (JComponent invalidComponent)
static RepaintManager	currentManager (Component c)
static RepaintManager	currentManager (JComponent c)
Rectangle	getDirtyRegion (JComponent aComponent)
Dimension	getDoubleBufferMaximumSize ()
Image	getOffscreenBuffer (Component c, int proposedWidth, int proposedHeight)
boolean	isCompletelyDirty (JComponent aComponent)
boolean	isDoubleBufferingEnabled ()
void	markCompletelyClean (JComponent aComponent)
void	markCompletelyDirty (JComponent aComponent)
void	paintDirtyRegions ()
void	removeInvalidComponent (JComponent component)
static void	setCurrentManager (RepaintManager aRepaintManager)
void	setDoubleBufferingEnabled (boolean aFlag)
void	setDoubleBufferMaximumSize (Dimension d)

String	toString ()
void	validateInvalidComponents ()

## ScrollPaneLayout 类

```
java.lang.Object
|
+-- javax.swing.ScrollPaneLayout
public class ScrollPaneLayout
    extends Object
    implements LayoutManager, ScrollPaneConstants, Serializable
```

这个类表示由 JScrollPane 自动使用的布局管理器。通常，不必直接处理这种布局。

### Inner Class

```
static class ScrollPaneLayout.UIResource
```

### Fields

protected JViewport	colHead
protected JScrollBar	hsb
protected int	hsbPolicy
protected Component	lowerLeft
protected Component	lowerRight
protected JViewport	rowHead
protected Component	upperLeft
protected Component	upperRight
protected JViewport	viewport
protected JScrollBar	vsb
protected int	vsbPolicy

### Constructor

```
ScrollPaneLayout ()
```

### Methods

void	addLayoutComponent (String s, Component c)
protected Component	addSingletonComponent (Component oldC, Component newC)
JViewport	getColumnHeader ()
Component	getCorner (String key)
JScrollBar	getHorizontalScrollBar ()
int	getHorizontalScrollBarPolicy ()
JViewport	getRowHeader ()
JScrollBar	getVerticalScrollBar ()
int	getVerticalScrollBarPolicy ()
JViewport	getViewport ()
Rectangle	getViewportBorderBounds (JScrollPane scrollpane)
	// Deprecated. Use the method
	JScrollPane.getViewportBorderBounds () .

```

void           layoutContainer (Container parent)
Dimension    minimumLayoutSize (Container parent)
Dimension    preferredLayoutSize (Container parent)
void          removeLayoutComponent (Component c)
void          setHorizontalScrollBarPolicy (int x)
void          setVerticalScrollBarPolicy (int x)
void          syncWithScrollPane (JScrollPane sp)

```

## ScrollPaneLayout, UIResource 类

```

java.lang.Object
|
+-- javax.swing.ScrollPaneLayout
|
+-- javax.swing.ScrollPaneLayout.UIResource
public static class ScrollPaneLayout.UIResource
    extends ScrollPaneLayout
    implements UIResource

```

这个类表示 ScrollPaneLayout 的 UI 资源。

### Constructor

```
ScrollPaneLayout.UIResource ()
```

## SizeRequirements 类

```

java.lang.Object
|
+-- javax.swing.SizeRequirements
public class SizeRequirements
    extends Object
    implements Serializable

```

这个类包含计算组件大小和位置的功能。

### Fields

float	alignment
int	maximum
int	minimum
int	preferred

### Constructors

```

SizeRequirements ()
SizeRequirements (int min, int pref, int max, float a)

```

### Methods

static int []	adjustSizes (int delta, SizeRequirements [] children)
static void	calculateAlignedPositions (int allocated, SizeRequirements total,

```

        SizeRequirements [] children,
        int [] offsets, int [] spans)
static void calculateTiledPositions (int allocated,
        SizeRequirements total,
        SizeRequirements [] children,
        int [] offsets, int [] spans)
static SizeRequirements getAlignedSizeRequirements
        (SizeRequirements [] children)
static SizeRequirements getTiledSizeRequirements
        (SizeRequirements [] children)
String toString ()

```

## SwingUtilities 类

```

java.lang.Object
|
+-- javax.swing.SwingUtilities
public class SwingUtilities
    extends Object
    implements SwingConstants

```

这个类包含了一些工具方法，它们在使用 Swing 组件的应用程序中是有用的。

### Methodss

static Rectangle []	computeDifference (Rectangle rectA, Rectangle rectB)
static Rectangle	computeIntersection (int x, int y int width, int height, Rectangle dest)
static int	computeStringWidth (FontMetrics fm, String str)
static Rectangle	computeUnion (int x, int y, int width, int height, Rectangle dest)
static MouseEvent	convertMouseEvent (Component source, MouseEvent sourceEvent, destination)
static Point	convertPoint (Component source, int x, int y, Component destination)
static Point	convertPoint (Component source, Point aPoint, Component destination)
static void	convertPointFromScreen (Point p, Component c)
static void	convertPointToScreen (Point p, Component c)
static Rectangle	convertRectangle (Component source, Rectangle aRectangle, Component destination)
static Component	findFocusOwner (Component c)
static Accessible	getAccessibleAt (Component c, Point p)

static Accessible	getAccessibleChild (Component c, int i)
static int	getAccessibleChildrenCount (Component c)
static int	getAccessibleIndexInParent (Component c)
static AccessibleStateSet	getAccessibleStateSet (Component c)
static Container	getAncestorNamed (String name, Component comp)
static Container	getAncestorOfClass (Class c, Component comp)
static Component	getDeepestComponentAt (Component parent, int x, int y)
static Rectangle	getLocalBounds (Component aComponent)
static Component	getRoot (Component c)
static JRootPane	getRootPane (Component c)
static void	invokeAndWait (Runnable doRun)
static void	invokeLater (Runnable doRun)
static boolean	isDescendingFrom (Component a, Component b)
static boolean	isEventDispatchThread ()
static boolean	isLeftMouseButton (MouseEvent anEvent)
static boolean	isMiddleMouseButton (MouseEvent anEvent)
static boolean	isRectangleContainingRectangle (Rectangle a, Rectangle b)
static boolean	isRightMouseButton (MouseEvent anEvent)
static String	layoutCompoundLabel (FontMetrics fm, String text, Icon icon, int verticalAlignment, int horizontalAlignment, int verticalTextPosition, int horizontalTextPosition, Rectangle viewR, Rectangle iconR, Rectangle textR, int textIconGap)
static String	layoutCompoundLabel (JComponent c, FontMetrics fm, String text, Icon icon, int verticalAlignment, int horizontalAlignment, int verticalTextPosition, int horizontalTextPosition, Rectangle viewR, Rectangle iconR, Rectangle textR, int textIconGap)
static void	paintComponent (Graphics g, Component c, Container p, int x, int y, int w, int h)
static void	paintComponent (Graphics g, Component c, Container p, Rectangle r)
static void	updateComponentTreeUI (Component c)
static Window	windowForComponent (Component aComponent)

## Timer 类

```
java.lang.Object
|
+-- javax.swing.Timer
public class Timer
    extends Object
    implements Serializable
```

这个类表示定时器，它控制按指定速率反复进行的动作。

### Field

```
protected EventListenerList listenerList
```

### Constructor

```
Timer (int delay, ActionListener listener)
```

### Methods

void	addActionListener (ActionListener listener)
protected void	fireActionPerformed (ActionEvent e)
int	getDelay ()
int	getInitialDelay ()
static boolean	getLogTimers ()
boolean	isCoalesce ()
boolean	isRepeats ()
boolean	isRunning ()
void	removeActionListener (ActionListener listener)
void	restart ()
void	setCoalesce (boolean flag)
void	setDelay (int delay)
void	setInitialDelay (int initialDelay)
static void	setLogTimers (boolean flag)
void	setRepeats (boolean flag)
void	start ()
void	stop ()

## ToolTipManager 类

```
java.lang.Object
|
+-- java.awt.event.MouseAdapter
    |
    +-- javax.swing.ToolTipManager
public class ToolTipManager
    extends MouseAdapter
    implements MouseMotionListener
```

这个类有助于管理应用程序中各种 UI 组件的 ToolTips（工具提示）。

**Inner Classes**

protected class	ToolTipManager.insideTimerAction
protected class	ToolTipManager.outsideTimerAction
protected class	ToolTipManager.stillInsideTimerAction

**Fields**

protected boolean	heavyWeightPopupEnabled
protected boolean l	lightWeightPopupEnabled

**Methods**

int	getDismissDelay ()
int	getInitialDelay ()
int	getReshowDelay ()
boolean	isEnabled ()
boolean	isLightWeightPopupEnabled ()
void	mouseDragged (MouseEvent event)
void	mouseEntered (MouseEvent event)
void	mouseExited (MouseEvent event)
void	mouseMoved (MouseEvent event)
void	mousePressed (MouseEvent event)
void	registerComponent (JComponent component)
void	setDismissDelay (int microSeconds)
void	setEnabled (boolean flag)
void	setInitialDelay (int microSeconds)
void	setLightWeightPopupEnabled (boolean aFlag)
	// Deprecated. Use the method
void	setToolTipWindowUsePolicy (int)
void	setReshowDelay (int microSeconds)
static ToolTipManager	sharedInstance ()
void	unregisterComponent (JComponent component)

**UIDefaults 类**

```

java.lang.Object
|
+-- java.util.Dictionary
|
+-- java.util.Hashtable
|
+-- javax.swing.UIDefaults

public class UIDefaults
    extends Hashtable

```

这个类表示具有 Swing 组件外观类型缺省值的表格。这些值由应用程序通过 UI 管理器来访问。

**Inner Classes**

static interface	UIDefaults.ActiveValue
static interface	UIDefaults.LazyValue

**Constructors**

```
UIDefaults ()  
UIDefaults (Object [] keyValueList)
```

**Methods**

void	addPropertyChangeListener (PropertyChangeListener listener)
protected void	firePropertyChange (String propertyName, Object oldValue, Object newValue)
Object	get (Object key)
Border	getBorder (Object key)
Color	getColor (Object key)
Dimension	getDimension (Object key)
Font	getFont (Object key)
Icon	getIcon (Object key)
Insets	getInsets (Object key)
int	getInt (Object key)
String	getString (Object key)
ComponentUI	getUI (JComponent target)
Class	getUIClass (String uiClassID)
Class	getUIClass (String uiClassID, ClassLoader uiClassLoader)
protected void	getUIError (String msg)
Object	put (Object key, Object value)
void	putDefaults (Object [] keyValueList)
void	removePropertyChangeListener (PropertyChangeListener listener)

**UIManager 类**

```
java.lang.Object  
|  
+-- javax.swing.UIManager  
public class UIManager  
    extends Object  
    implements Serializable
```

这个类表示 UI 管理器，该 UI 管理器负责监视当前外观和它的缺省外观。当需要给界面赋以新的外观时，就要用到这个类。

**Inner Classes**

```
static class UIManager.LookAndFeelInfo
```

**Constructor**`UIManager ()`**Methods**

<code>static void</code>	<code>addAuxiliaryLookAndFeel (LookAndFeel laf)</code>
<code>static void</code>	<code>addPropertyChangeListener (PropertyChangeListener listener)</code>
<code>static Object</code>	<code>get (Object key)</code>
<code>static LookAndFeel []</code>	<code>getAuxiliaryLookAndFeels ()</code>
<code>static Border</code>	<code>getBorder (Object key)</code>
<code>static Color</code>	<code>getColor (Object key)</code>
<code>static String</code>	<code>getCrossPlatformLookAndFeelClassName ()</code>
<code>static UIDefaults</code>	<code>getDefaults ()</code>
<code>static Dimension</code>	<code>getDimension (Object key)</code>
<code>static Font</code>	<code>getFont (Object key)</code>
<code>static Icon</code>	<code>getIcon (Object key)</code>
<code>static Insets</code>	<code>getInsets (Object key)</code>
<code>static UIManager.LookAndFeelInfo []</code>	<code>getInstalledLookAndFeels ()</code>
<code>static int</code>	<code>getInt (Object key)</code>
<code>static LookAndFeel</code>	<code>getLookAndFeel ()</code>
<code>static UIDefaults</code>	<code>getLookAndFeelDefaults ()</code>
<code>static String</code>	<code>getString (Object key)</code>
<code>static String</code>	<code>getSystemLookAndFeelClassName ()</code>
<code>static ComponentUI</code>	<code>getUI (JComponent target)</code>
<code>static void</code>	<code>installLookAndFeel (String name, String className)</code>
<code>static void</code>	<code>installLookAndFeel (UIManager.LookAndFeelInfo info)</code>
<code>static Object</code>	<code>put (Object key, Object value)</code>
<code>static boolean</code>	<code>removeAuxiliaryLookAndFeel (LookAndFeel laf)</code>
<code>static void</code>	<code>removePropertyChangeListener (PropertyChangeListener listener)</code>
<code>static void</code>	<code>setInstalledLookAndFeels (UIManager.LookAndFeelInfo [] infos)</code>
<code>static void</code>	<code>setLookAndFeel (LookAndFeel newLookAndFeel)</code>

```
static void setLookAndFeel (String className)
```

## UIManager.LookAndFeelInfo 类

```
java.lang.Object
|
+-- javax.swing.UIManager.LookAndFeelInfo
public static class UIManager.LookAndFeelInfo
    extends Object
```

这个类有助于菜单提供外观信息来配置应用程序启动等操作。

### Constructor

```
UIManager.LookAndFeelInfo (String name, String className)
```

### Methods

String	getClassName ()
String	getName ()
String	toString ()

## ViewportLayout 类

```
java.lang.Object
|
+-- javax.swing.ViewportLayout
public class ViewportLayout
    extends Object
    implements LayoutManager, Serializable
```

这个类表示布局管理器，它由对象自动使用。一般情况下，没有必要直接使用这个类。

### Constructor

```
ViewportLayout ()
```

### Methods

void	addLayoutComponent (String name, Component c)
void	layoutContainer (Container parent)
Dimension	minimumLayoutSize (Container parent)
Dimension	preferredLayoutSize (Container parent)
void	removeLayoutComponent (Component c)

## UnsupportedLookAndFeelException 类

```
java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- javax.swing.UnsupportedLookAndFeelException
public class UnsupportedLookAndFeelException
    extends Exception
```

这个类表示一个异常，当用户的计算机系统里没有指定的外观类型时，就会抛出这个异常。

### Constructor

```
UnsupportedLookAndFeelException (String s)
```

## javax.swing.border

### Border 接口

```
public abstract interface Border
```

这是一个设计级接口，表示围绕组件边缘的边界。

### Methods

Insets	getBorderInsets (Component c)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## AbstractBorder 类

```
java.lang.Object  
+ -- javax.swing.border.AbstractBorder  
public abstract class AbstractBorder  
    extends Object  
    implements Border, Serializable
```

这个类表示抽象边界，它是各种边界类的基类。

### Constructor

```
AbstractBorder ()
```

### Methods

Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
static Rectangle	getInteriorRectangle (Component c, Border b, int x, int y, int width, int height)
Rectangle	getInteriorRectangle (Component c, int x, int y, int width, int height) . . .
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## BevelBorder 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.BevelBorder
public class BevelBorder
    extends AbstractBorder

```

这个类表示斜角边界，它要么是凸起的，要么是凹进的。

### Fields

protected int	bevelType
protected Color	highlightInner
protected Color	highlightOuter
static int	LOWERED
static int	RAISED
protected Color	shadowInner
protected Color	shadowOuter

### Constructors

```

BevelBorder (int bevelType)
BevelBorder (int bevelType, Color highlight, Color shadow)
BevelBorder (int bevelType, Color highlightOuter, Color highlightInner,
Color shadowOuter, Color shadowInner)

```

### Methods

int	getBevelType ()
Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
Color	getHighlightInnerColor (Component c)
Color	getHighlightOuterColor (Component c)
Color	getShadowInnerColor (Component c)
Color	getShadowOuterColor (Component c)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)
protected void	paintLoweredBevel (Component c, Graphics g, int x, int y, int width, int height)
protected void	paintRaisedBevel (Component c, Graphics g, int x, int y, int width, int height)

## CompoundBorder 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.CompoundBorder
public class CompoundBorder
    extends AbstractBorder

```

这个类表示复合边界。所谓复合边界，就是通过将一个内部边界嵌套在一个外部边界里，从而将两个边界对象合并成一个边界。

### Fields

protected Border	insideBorder
protected Border	outsideBorder

### Constructors

CompoundBorder ()
CompoundBorder (Border outsideBorder, Border insideBorder)

### Methods

Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
Border	getInsideBorder ()
Border	getOutsideBorder ()
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## EmptyBorder 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EmptyBorder
public class EmptyBorder
    extends AbstractBorder
    implements Serializable

```

这个类表示空边界，它具有指定的镶边。

### Fields

protected int	bottom
protected int	left
protected int	right
protected int	top

## Constructors

```
EmptyBorder (Insets insets)
EmptyBorder (int top, int left, int bottom, int right)
```

## Methods

Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## EtchedBorder 类

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EtchedBorder
public class EtchedBorder
    extends AbstractBorder
```

这个类表示浮雕边界，它可以是凹进的，也可以是凸起的。

## Fields

protected int	etchType
protected Color	highlight
static int	LOWERED
static int	RAISED
protected Color	shadow

## Constructors

```
EtchedBorder ()
EtchedBorder (Color highlight, Color shadow)
EtchedBorder (int etchType)
EtchedBorder (int etchType, Color highlight, Color shadow)
```

## Methods

Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
int	getEtchType ()
Color	getHighlightColor (Component c)
Color	getShadowColor (Component c)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## MatteBorder 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EmptyBorder
|
+-- javax.swing.border.MatteBorder
public class MatteBorder
    extends EmptyBorder

```

这个类表示粗糙边界，它被画上纯色或图标。

### Fields

protected Color	color
protected Icon	tileIcon

### Constructors

MatteBorder (Icon tileIcon)
MatteBorder (int top, int left, int bottom, int right, Color color)
MatteBorder (int top, int left, int bottom, int right, Icon tileIcon)

### Methods

Insets	getBorderInsets (Component c)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## SoftBevelBorder 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.BevelBorder
|
+-- javax.swing.border.SoftBevelBorder
public class SoftBevelBorder
    extends BevelBorder

```

这个类表示有平滑拐角的斜角边界，它是 BevelBorder 的子类。

### Constructors

SoftBevelBorder (int bevelType)
SoftBevelBorder (int bevelType, Color highlight, Color shadow)
SoftBevelBorder (int bevelType, Color highlightOuter, Color highlightInner, Color shadowOuter, Color shadowInner)

## Methods

Insets	getBorderInsets (Component c)
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## TitledBorder 类

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.TitledBorder
public class TitledBorder
    extends AbstractBorder
```

这个类表示带标题的边界。利用其所支持的字段，可以控制标题的位置。

## Fields

static int	ABOVE_BOTTOM
static int	ABOVE_TOP
static int	BELOW_BOTTOM
static int	BELOW_TOP
protected Border	border
static int	BOTTOM
static int	CENTER
static int	DEFAULT_JUSTIFICATION
static int	DEFAULT_POSITION
protected static int	EDGE_SPACING
static int	LEFT
static int	RIGHT
protected static int	TEXT_INSET_H
protected static int	TEXT_SPACING
protected String	title
protected Color	titleColor
protected Font	titleFont
protected int	titleJustification
protected int	titlePosition
static int	TOP

## Constructors

```
TitledBorder (Border border)
TitledBorder (Border border, String title)
TitledBorder (Border border, String title, int titleJustification,
int titlePosition)
TitledBorder (Border border, String title, int titleJustification,
```

```

int titlePosition, Font titleFont)
TitledBorder (Border border, String title, int titleJustification,
int titlePosition, Font titleFont, Color titleColor)
TitledBorder (String title)

```

**Methods**

Border	getBorder ()
Insets	getBorderInsets (Component c)
Insets	getBorderInsets (Component c, Insets insets)
protected Font	getFont (Component c)
Dimension	getMinimumSize (Component c)
String	getTitle ()
Color	getTitleColor ()
Font	getTitleFont ()
int	getTitleJustification ()
int	getTitlePosition ()
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)
void	setBorder (Border border)
void	setTitle (String title)
void	setTitleColor (Color titleColor)
void	setTitleFont (Font titleFont)
void	setTitleJustification (int titleJustification)
void	setTitlePosition (int titlePosition)

**javax.swing.colorchooser****ColorSelectionModel 接口**

```
public abstract interface ColorSelectionModel
```

本接口表示在 Swing 中颜色选择器的选择方式。

**Methods**

void	addChangeListener(ChangeListener listener)
	//Adds listener as a listener to changes in the model.
Color	getSelectedColor()
void	removeChangeListener(ChangeListener listener)
	//Removes listener as a listener to changes in the model.
void	setSelectedColor(Color color)
	//Sets the model's selected Color to color.

## AbstractColorChooserPanel 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JPanel
|
+-- javax.swing.colorchooser.AbstractColorChooserPanel
public abstract class AbstractColorChooserPanel
    extends JPanel

```

这个抽象类表示 Swing 颜色选择器的超类。要增加另外的颜色选择窗格，需通过此类扩展用户生成的类。

### Constructor

```
AbstractColorChooserPanel()
```

### Methods

protected abstract void	buildChooser()
protected Color	getColorFrameModel()
ColorSelectionModel	getColorSelectionModel()
abstract String	getDisplayName()
abstract Icon	getLargeDisplayIcon()
abstract Icon	getSmallDisplayIcon()
Void	installChooserPanel(JColorChooser enclosingChooser)         //This get called when the panel is added         to the chooser.
void	paint(Graphics g)
void	uninstallChooserPanel(JColorChooser         enclosingChooser)         //This get called when the panel is removed         from the chooser.
abstract void	updateChooser()         //Override this method to update your ChooserPanel.         This method will be automatically called when the         model's state changes.

## ColorChooserComponentFactory 类

```

java.lang.Object
|
+-- javax.swing.colorchooser.ColorChooserComponentFactory
public class ColorChooserComponentFactory
    extends Object

```

此类表示一个附属类的类工厂，它可以插入到 Swing 颜色选择器中。

### Methods

static AbstractColorChooserPanel []	getDefauletChoosersPanels ()
static JComponent	getPreviewPanel ()

### DefaultColorSelectionModel 类

```
java.lang.Object
+
+ -- javax.swing.colorchooser.DefaultColorSelectionModel
public class DefaultColorSelectionModel
    extends Object
    implements ColorSelectionModel, Serializable
```

这个类表示 ColorSelectionModel 接口的缺省实现形式。

### Fields

protected ChangeEvent	changeEvent
	//Only one ChangeEvent is needed per model
	instance since the event's only (read-only)
	state is the source property.
protected EventListenerList	listenerList

### Constructors

```
DefaultColorSelectionModel ()
//Default constructor.

DefaultColorSelectionModel (Color color)
//Initializes selected Color to color
```

### Methods

void	addChangeListener(ChangeListener l)
	//Adds a ChangeListener to the model.
protected void	fireStateChanged()
	//Run each ChangeListener's stateChanged() method.
Color	getSelectedColor()
void	removeChangeListener(ChangeListener l)
	//Removes a ChangeListener from the model.
void	setSelectedColor(Color color)

## javax.swing.event

### AncestorListener 接口

```
public abstract interface AncestorListener
    extends EventListener
```

这个接口表示祖先监听类 (ancestor listener)。当一个组件或它的祖先发生变化时，就会

通知此实现类。这种变化可能是组件移动、组件可见性的改变，等等。

### Methods

```
void ancestorAdded (AncestorEvent event)  
void ancestorMoved (AncestorEvent event)  
void ancestorRemoved (AncestorEvent event)
```

## CaretListener 接口

```
public abstract interface CaretListener  
    extends EventListener
```

这个接口表示光标监听类，当文本组件中的光标位置改变时，就会通知此实现类。

### Method

```
void caretUpdate (CaretEvent e)
```

## CellEditorListener 接口

```
public abstract interface CellEditorListener  
    extends EventListener
```

这个接口表示单元编辑器监听类。当单元中有确认了的新编辑操作时，就会通知此实现类。

### Methods

```
void editingCanceled (ChangeEvent e)  
void editingStopped (ChangeEvent e)
```

## ChangeListener 接口

```
public abstract interface ChangeListener  
    extends EventListener
```

这个接口表示变化监听类。当组件选择发生变化时，就会通过 ChangeEvent 对象通知此实现类。

### Method

```
void stateChanged (ChangeEvent e)
```

## DocumentEvent 接口

```
public abstract interface DocumentEvent
```

这个接口表示文档事件。当正在修改文档时，该接口的实现就会保存对文档所做的修改。

### Inner Classes

```
static interface DocumentEvent.ElementChange  
static class DocumentEvent.EventType
```

**Methods**

DocumentEvent.ElementChange	getChange (Element elem)
Document	getDocument ()
int	getLength ()
int	getOffset ()
DocumentEvent.EventType	getType ()

**DocumentEvent.ElementChange 接口**

```
public abstract static interface DocumentEvent.ElementChange
```

这是一个附属接口，它表示对文档元素的修改。

**Methods**

Element []	getChildrenAdded ()
Element []	getChildrenRemoved ()
Element	getElement ()
int	getIndex ()

**DocumentListener 接口**

```
public abstract interface DocumentListener
    extends EventListener
```

这个接口表示文档监听类。当文档的内容发生改变时，就会通知此实现类。

**Methods**

void	changedUpdate (DocumentEvent e)
void	insertUpdate (DocumentEvent e)
void	removeUpdate (DocumentEvent e)

**HyperlinkListener 接口**

```
public abstract interface HyperlinkListener
    extends EventListener
```

这个接口表示超链接监听类。当超文本接收一个动作（比如鼠标单击）时，该实现类用于通知 HyperLinkEvent 类型的事件。

**Method**

void	hyperlinkUpdate (HyperlinkEvent e)
------	------------------------------------

**InternalFrameListener 接口**

```
public abstract interface InternalFrameListener
    extends EventListener
```

这个接口表示内部框架监听类。当 InternalFrameEvent 类型的事件由内部框架触出时，就会通知此实现类。这个接口的功能类似于 AWT 中使用的 WindowListener。

**Methods**

```

void internalFrameActivated (InternalFrameEvent e)
void internalFrameClosed (InternalFrameEvent e)
void internalFrameClosing (InternalFrameEvent e)
void internalFrameDeactivated (InternalFrameEvent e)
void internalFrameDeiconified (InternalFrameEvent e)
void internalFrameIconified (InternalFrameEvent e)
void internalFrameOpened (InternalFrameEvent e)

```

**ListDataListener 接口**

```

public abstract interface ListDataListener
    extends EventListener

```

这个接口表示列表数据监听类。当列表组件的数据发生改变时，就会通知此实现类。

**Methods**

```

void contentsChanged (ListDataEvent e)
void intervalAdded (ListDataEvent e)
void intervalRemoved (ListDataEvent e)

```

**ListSelectionListener 接口**

```

public abstract interface ListSelectionListener
    extends EventListener

```

这个接口表示列表选择监听类。当列表组件的选择值发生改变时，就会通知此实现类。

**Method**

```
void valueChanged (ListSelectionEvent e)
```

**MenuDragMouseListener 接口**

```

public abstract interface MenuDragMouseListener
    extends EventListener

```

这个接口表示菜单鼠标拖动监听类。当一个拖动的鼠标进入、离开菜单或在菜单的显示区内释放时，就会通知此实现类。

**Methods**

```

void menuDragMouseDragged (MenuDragMouseEvent e)
void menuDragMouseEntered (MenuDragMouseEvent e)
void menuDragMouseExited (MenuDragMouseEvent e)
void menuDragMouseReleased (MenuDragMouseEvent e)

```

**MenuKeyListener 接口**

```

public abstract interface MenuKeyListener
    extends EventListener

```

该接口表示菜单按键监听类。当键被按下、释放或在有输入焦点的情况下发生按键动作时，就会通知此实现类。

#### Methods

```
void menuKeyPressed (MenuKeyEvent e)  
void menuKeyReleased (MenuKeyEvent e)  
void menuKeyTyped (MenuKeyEvent e)
```

### MenuListener 接口

```
public abstract interface MenuListener  
extends EventListener
```

这个接口表示菜单事件的监听类。当选定、取消选定或者取消某一菜单时，就会通知此实现类。

#### Methods

```
void menuCanceled (MenuEvent e)  
void menuDeselected (MenuEvent e)  
void menuSelected (MenuEvent e)
```

### MouseListener 接口

```
public abstract interface MouseInputListener  
extends MouseListener, MouseMotionListener
```

这个接口表示鼠标输入监听类。当鼠标进入或离开组件，或者当鼠标在组件上被按下、释放或单击时，或者在组件上移动或拖动鼠标时，都会通知此实现类。

### PopupMenuListener 接口

```
public abstract interface PopupMenuItemListener  
extends EventListener
```

这个接口表示弹出式菜单监听类。当弹出式菜单变为可见的、不可见的或者被取消时，就会通知此实现类。

#### Methods

```
void popupMenuCanceled (PopupMenuEvent e)  
void popupMenuWillBecomeInvisible (PopupMenuEvent e)  
void popupMenuWillBecomeVisible (PopupMenuEvent e)
```

### TableColumnModelListener 接口

```
public abstract interface TableColumnModelListener  
extends EventListener
```

这个接口表示表格栏模式监听类。当表格栏模式的数据发生改变时，就会通知此实现类。

**Methods**

```
void    columnAdded (TableColumnModelEvent e)
void    columnMarginChanged (ChangeEvent e)
void    columnMoved (TableColumnModelEvent e)
void    columnRemoved (TableColumnModelEvent e)
void    columnSelectionChanged (ListSelectionEvent e)
```

**TableModelListener 接口**

```
public abstract interface TableModelListener
extends EventListener
```

这个接口表示表格模式监听类。当表格模式的数据发生改变时，就会通知此实现类。

**Method**

```
void    tableChanged (TableModelEvent e)
```

**TreeExpansionListener 接口**

```
public abstract interface TreeExpansionListener
extends EventListener
```

这个接口表示树扩展监听类。当一棵树沿着一条路径扩展或压缩节点时，就会通知此实现类。

**Methods**

```
void    treeCollapsed (TreeExpansionEvent event)
void    treeExpanded (TreeExpansionEvent event)
Interface TreeModelListener
public abstract interface TreeModelListener
extends EventListener
```

这个接口表示树改变监听类。当树的节点数据发生改变时，就会通知此实现类。

**Methods**

```
void    treeNodesChanged (TreeModelEvent e)
void    treeNodesInserted (TreeModelEvent e)
void    treeNodesRemoved (TreeModelEvent e)
void    treeStructureChanged (TreeModelEvent e)
```

**TreeSelectionListener 接口**

```
public abstract interface TreeSelectionListener
extends EventListener
```

这个接口表示树选择监听类。当树的选择值发生改变时，就会通知此类。

**Method**

```
void    valueChanged (TreeSelectionEvent e)
```

## TreeWillExpandListener 接口

```
public abstract interface TreeWillExpandListener
    extends EventListener
```

这个接口表示树扩展监听类。当树扩展或压缩节点时，就会通知此类。

### Methods

```
void treeWillCollapse (TreeExpansionEvent event)
void treeWillExpand (TreeExpansionEvent event)
```

## UndoableEditListener 接口

```
public abstract interface UndoableEditListener
    extends EventListener
```

这个接口表示可撤消编辑监听类。当在文本组件中执行了可撤消编辑动作时，就会通知此类。

### Method

```
void undoableEditHappened (UndoableEditEvent e)
```

## AncestorEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- javax.swing.event.AncestorEvent
public class AncestorEvent
    extends AWTEvent
```

这个类表示一个祖先事件 (ancestor event)。当组件或它的祖先发生改变时，就会发出该事件。

### Fields

```
static int ANCESTOR_ADDED
static int ANCESTOR_MOVED
static int ANCESTOR_REMOVED
```

### Constructor

```
AncestorEvent (JComponent source, int id, Container ancestor,
Container ancestorParent)
```

### Methods

```
Container getAncestor ()
Container getAncestorParent ()
JComponent getComponent ()
```

## CaretEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.CaretEvent
public abstract class CaretEvent
    extends EventObject
```

这个类表示插入光标事件 (caret event), 用来通知文本组件中的插入光标位置已经改变。

### Constructor

`CaretEvent (Object source)` Creates a new CaretEvent object.

### Methods

```
abstract int     getDot ()
abstract int     getMark ()
```

## ChangeEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.ChangeEvent
public class ChangeEvent
    extends EventObject
```

这个类表示改变事件。触发改变事件是用来通知事件源中的选择状态已经发生改变。

### Constructor

`ChangeEvent (Object source)`

## DocumentEvent.EventType 类

```
java.lang.Object
|
+-- javax.swing.event.DocumentEvent.EventType
public static final class DocumentEvent.EventType
    extends Object
```

这个类是针对文档事件类型的类型安全枚举 (typesafe enumeration) 的支持类。

### Fields

static DocumentEvent.EventType	CHANGE
static DocumentEvent.EventType	INSERT
static DocumentEvent.EventType	REMOVE

### Method

`String toString ()`

## EventListenerList 类

```
java.lang.Object
|
+-- javax.swing.event.EventListenerList
public class EventListenerList
    extends Object
    implements Serializable
```

这个类通过使用数组来表示事件监听类列表。

### Field

```
protected Object [] listenerList
```

### Constructor

```
EventListenerList ()
```

### Methods

void	add (Class t, EventListener l)
int	getListenerCount ()
int	getListenerCount (Class t)
Object []	getListenerList ()
void	remove (Class t, EventListener l)
String	toString ()

## HyperlinkEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.HyperlinkEvent
public class HyperlinkEvent
    extends EventObject
```

这个类表示超链接事件。当在 HTML 文档里的超链接上发生某种动作时，就会触发超链接事件。

### Inner Class

```
static class HyperlinkEvent.EventType
```

### Constructors

```
HyperlinkEvent (Object source, HyperlinkEvent.EventType type, URL u)
HyperlinkEvent (Object source, HyperlinkEvent.EventType type, URL u, String desc)
```

### Methods

String	getDescription ()
HyperlinkEvent.EventType	getEventType ()
URL	getURL ()

## HyperlinkEvent.EventType 类

```
java.lang.Object
|
+-- javax.swing.event.HyperlinkEvent.EventType
public static final class HyperlinkEvent.EventType
    extends Object
```

这是一个附属的内部类，它定义的静态事件类型，如 ENTERED、EXITED 和 ACTIVATED，用来指定超链接事件。

### Fields

static HyperlinkEvent.EventType	ACTIVATED
static HyperlinkEvent.EventType	ENTERED
static HyperlinkEvent.EventType	EXITED

### Method

```
String    toString ()
```

## InternalFrameAdapter 类

```
java.lang.Object
|
+-- javax.swing.event.InternalFrameAdapter
public abstract class InternalFrameAdapter
    extends Object
    implements InternalFrameListener
```

这个类表示抽象适配器，可以扩展它来创建监听程序类，以接收内部框架事件。

### Constructor

```
InternalFrameAdapter ()
```

### Methods

```
void    internalFrameActivated (InternalFrameEvent e)
void    internalFrameClosed (InternalFrameEvent e)
void    internalFrameClosing (InternalFrameEvent e)
void    internalFrameDeactivated (InternalFrameEvent e)
void    internalFrameDeiconified (InternalFrameEvent e)
void    internalFrameIconified (InternalFrameEvent e)
void    internalFrameOpened (InternalFrameEvent e)
```

## InternalFrameEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- javax.swing.event.InternalFrameEvent
public class InternalFrameEvent
    extends AWTEvent
```

这个类表示内部框架事件。当内部框架上要执行诸如打开或关闭、激活或取消激活、图标化或取消图标化等操作时，就会触发该事件。

### Fields

```
static int INTERNAL_FRAME_ACTIVATED  
static int INTERNAL_FRAME_CLOSED  
static int INTERNAL_FRAME_CLOSING  
static int INTERNAL_FRAME_DEACTIVATED  
static int INTERNAL_FRAME_DEICONIFIED  
static int INTERNAL_FRAME_FIRST  
static int INTERNAL_FRAME_ICONIFIED  
static int INTERNAL_FRAME_LAST  
static int INTERNAL_FRAME_OPENED
```

### Constructor

```
InternalFrameEvent (JInternalFrame source, int id)
```

### Method

```
String paramString ()
```

## ListDataEvent 类

```
java.lang.Object  
|  
+-- java.util.EventObject  
|  
+-- javax.swing.event.ListDataEvent  
public class ListDataEvent  
    extends EventObject
```

这个类表示列表数据事件；当列表组件模式中的数据发生改变时，就会触发该事件。

### Fields

```
static int CONTENTS_CHANGED  
static int INTERVAL_ADDED  
static int INTERVAL_REMOVED
```

### Constructor

```
ListDataEvent (Object source, int type, int index0, int index1)
```

### Methods

```
int getIndex0 ()  
int getIndex1 ()  
int getType ()
```

## ListSelectionEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.ListSelectionEvent
public class ListSelectionEvent
    extends EventObject

```

这个类表示列表选择事件。当列表项的选择发生改变时，就会触发该事件。

### Constructor

```
ListSelectionEvent (Object source, int firstIndex, int lastIndex, boolean isAdjusting)
```

### Methods

int	getFirstIndex ()
int	getLastIndex ()
boolean	getValueIsAdjusting ()
String	toString ()

## MenuDragMouseEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.InputEvent
|
+-- java.awt.event.MouseEvent
|
+-- javax.swing.event.MenuDragMouseEvent
public class MenuDragMouseEvent
    extends MouseEvent

```

这个类表示在菜单上的鼠标拖动事件。当在菜单上拖动鼠标时，就会触发该事件。

### Constructor

```
MenuDragMouseEvent (Component source, int id, long when, int modifiers,
int x, int y, int clickCount, boolean popupTrigger, MenuElement [] p,
MenuSelectionManager m)
```

### Methods

MenuSelectionManager	getMenuSelectionManager ()
MenuElement []	getPath ()

## MenuEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.MenuEvent
public class MenuEvent
    extends EventObject
```

这个类表示菜单事件。当打开、选择被取消菜单组件时，就会触发该事件。

### Constructor

```
MenuEvent (Object source)
```

## MenuKeyEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.InputEvent
|
+-- java.awt.event.KeyEvent
|
+-- javax.swing.event.MenuKeyEvent
public class MenuKeyEvent
    extends KeyEvent
```

这个类表示按键事件，当菜单接收到一个按键事件时，用于通知已经注册的监听类。

### Constructor

```
MenuKeyEvent (Component source, int id, long when, int modifiers,
int keyCode, char keyChar, MenuElement [] p, MenuSelectionManager m)
```

### Methods

MenuSelectionManager	getMenuSelectionManager ()
MenuElement []	getPath ()

## MouseInputAdapter 类

```
java.lang.Object
|
+-- javax.swing.event.MouseInputAdapter
public abstract class MouseInputAdapter
    extends Object
    implements MouseInputListener
```

这是一个适配器类，用来创建一个鼠标输入监听类，该监听类是适配器类的扩展类。这个扩展类只包含必要的方法，这些方法重载了适配器中的方法。

### **Constructor**

```
MouseInputAdapter ()
```

### **Methods**

```
void mouseClicked (MouseEvent e)
void mouseDragged (MouseEvent e)
void mouseEntered (MouseEvent e)
void mouseExited (MouseEvent e)
void mouseMoved (MouseEvent e)
void mousePressed (MouseEvent e)
void mouseReleased (MouseEvent e)
```

## **PopupMenuEvent 类**

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.PopupMenuEvent
public class PopupMenuEvent
    extends EventObject
```

这个类表示弹出式菜单事件。当弹出式菜单为可见的、不可见的或被取消时，就会触发该事件。

### **Constructor**

```
PopupMenuEvent (Object source)
```

## **SwingPropertyChangeSupport 类**

```
java.lang.Object
|
+-- java.beans.PropertyChangeSupport
|
+-- javax.swing.event.SwingPropertyChangeSupport
public final class SwingPropertyChangeSupport
    extends PropertyChangeSupport
```

这是 Swing 中支持属性变化的实用工具类。该类扩展了 JavaBean 的 PropertyChangeSupport 类，而这个类是支持界定属性 (bounded properties) 的。

### **Constructor**

```
SwingPropertyChangeSupport (Object sourceBean)
```

### **Methods**

```
void addPropertyChangeListener (PropertyChangeListener listener)
void addPropertyChangeListener (String propertyName,
```

```

        PropertyChangeListener listener)
void      firePropertyChange (PropertyChangeEvent evt)
void      firePropertyChange (String propertyName, Object oldValue,
                           Object newValue)
boolean   hasListeners (String propertyName)
void      removePropertyChangeListener (PropertyChangeListener listener)
void      removePropertyChangeListener (String propertyName,
                                       PropertyChangeListener listener)

```

## TableColumnModelEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.TableColumnModelEvent
public class TableColumnModelEvent
extends EventObject

```

这个类表示表格栏模式事件。触发该事件是用来通知已经注册的监听类，表格栏的数据模式已经因为增加、删除或移动表格栏而发生了变化。

### Fields

```

protected int    fromIndex
protected int    toIndex

```

### Constructor

```
TableColumnModelEvent (TableColumnModel source, int from, int to)
```

### Methods

```

int     getFromIndex ()
int     getToIndex ()

```

## TableModelEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.TableModelEvent
public class TableModelEvent
extends EventObject

```

这个类表示表格模式事件。触发该事件是用来通知已注册的监听类，表格的数据模式已经改变。

### Fields

```

static int      ALL_COLUMNS
protected int    column
static int      DELETE

```

```

protected int      firstRow
static int        HEADER_ROW
static int        INSERT
protected int      lastRow
protected int      type
static int        UPDATE

```

### Constructors

```

TableModelEvent (TableModel source)
TableModelEvent (TableModel source, int row)
TableModelEvent (TableModel source, int firstRow, int lastRow)
TableModelEvent (TableModel source, int firstRow, int lastRow, int column)
TableModelEvent (TableModel source, int firstRow, int lastRow, int column, int type)

```

### Methods

```

int   getColumn ()
int   getFirstRow ()
int   getLastRow ()
int   getType ()

```

## TreeExpansionEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.TreeExpansionEvent
public class TreeExpansionEvent
    extends EventObject

```

这个类表示树扩展事件，触发该事件是用来识别树上的路径做了怎样的扩展。

### Field

```
protected TreePath path
```

### Constructor

```
TreeExpansionEvent (Object source, TreePath path)
```

### Method

```
TreePath getPath ()
```

## TreeModelEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.TreeModelEvent
public class TreeModelEvent
    extends EventObject

```

这个类表示树模式事件。当树的数据模式改变时，就会触发该事件。

### Fields

protected int []	childIndices
protected Object []	children
protected TreePath	path

### Constructors

```
TreeModelEvent (Object source, Object [] path)
TreeModelEvent (Object source, Object [] path, int [] childIndices,
Object [] children)
TreeModelEvent (Object source, TreePath path)
TreeModelEvent (Object source, TreePath path, int [] childIndices,
Object [] children)
```

### Methods

int []	getChildIndices ()
Object []	getChildren ()
Object []	getPath ()
TreePath	getTreePath ()
String	toString ()

## TreeSelectionEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.TreeSelectionEvent
public class TreeSelectionEvent
    extends EventObject
```

这个类表示树选择事件。当树选择发生改变时，就会触发该事件。

### Fields

protected boolean []	areNew
protected TreePath	newLeadSelectionPath
protected TreePath	oldLeadSelectionPath
protected TreePath []	paths

### Constructor

```
TreeSelectionEvent (Object source, TreePath [] paths, boolean [] areNew,
TreePath oldLeadSelectionPath, TreePath newLeadSelectionPath)

TreeSelectionEvent (Object source, TreePath path, boolean isNew,
TreePath oldLeadSelectionPath, TreePath newLeadSelectionPath)
```

### Methods

Object	cloneWithSource (Object newSource)
TreePath	getNewLeadSelectionPath ()

```

TreePath           getOldLeadSelectionPath()
TreePath           getPath()
TreePath []        getPaths()
boolean           isAddedPath()
boolean           isAddedPath( TreePath path)

```

## UndoableEditEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.UndoableEditEvent
public class UndoableEditEvent
    extends EventObject

```

这个类表示可撤消编辑事件。当文本组件上发生可撤消编辑操作时，就会触发该事件。

### Constructor

```
UndoableEditEvent (Object source, UndoableEdit edit)
```

### Method

```
UndoableEdit getEdit()
```

## javax.swing.filechooser

### FileFilter 类

```

java.lang.Object
|
+-- javax.swing.filechooser.FileFilter
public abstract class FileFilter
    extends Object

```

这个类抽象的过滤器。它只显示文件选择器中指定的文件。为了创建文件过滤器，可以使用一个实现子类来重载这个类的方法。

### Constructor

```
FileFilter ()
```

### Methods

abstract boolean	accept (File f)
abstract String	getDescription()

### FileSystemView 类

```

java.lang.Object
|
+-- javax.swing.filechooser.FileSystemView
public abstract class FileSystemView
    extends Object

```

这个类表示计算机文件系统的视图。

### Constructor

```
FileSystemView ()
```

### Methods

File	createFileObject (File dir, String filename)
File	createFileObject (String path)
abstract File	createNewFolder (File containingDir)
File []	getFiles (File dir, boolean useFileHiding)
static FileSystemView	getFileSystemView ()
File	getHomeDirectory ()
File	getParentDirectory (File dir)
abstract File []	getRoots ()
abstract boolean	isHiddenFile (File f)
abstract boolean	isRoot (File f)

## FileView 类

```
java.lang.Object
|
+-- javax.swing.filechooser.FileView
public abstract class FileView
    extends Object
```

这个类表示显示在文件选择器中的文件视图。可以生成该类的子类，给不同类型的文件赋以不同的图标。

### Constructor

```
FileView ()
```

### Methods

abstract String	getDescription (File f)
abstract Icon	getIcon (File f)
abstract String	getName (File f)
abstract String	getTypeDescription (File f)
abstract Boolean	isTraversable (File f)

## javax.swing.plaf

这个类在此程序包中只是简单地用做相应组件的可插入外观接口。例如，ButtonUI 类表示组件 JButton 的可插入外观接口。

### UIResource 接口

```
public abstract interface UIResource
```

这是一个设计级接口，它表示由 ComponentUI 创建的对象。可以找到 BorderUIResource

类及其实现此接口的内部类。

## BorderUIResource 类

```
java.lang.Object
|
+
+ -- javax.swing.plaf.BorderUIResource
public class BorderUIResource
    extends Object
    implements Border, UIResource, Serializable
```

这个类表示 UIResource 边界对象，该对象提供了对边界实例的包装。

### Inner Classes

static class	BorderUIResource.BevelBorderUIResource
static class	BorderUIResource.CompoundBorderUIResource
static class	BorderUIResource.EmptyBorderUIResource
static class	BorderUIResource.EtchedBorderUIResource
static class	BorderUIResource.LineBorderUIResource
static class	BorderUIResource.MatteBorderUIResource
static class	BorderUIResource.TitledBorderUIResource

### Constructor

```
BorderUIResource (Border delegate)
```

### Methods

static Border	getBlackLineBorderUIResource ()
Insets	getBorderInsets (Component c)
static Border	getEtchedBorderUIResource ()
static Border	getLoweredBevelBorderUIResource ()
static Border	getRaisedBevelBorderUIResource ()
boolean	isBorderOpaque ()
void	paintBorder (Component c, Graphics g, int x, int y, int width, int height)

## BorderUIResource.BevelBorderUIResource 类

```
java.lang.Object
|
+
+ -- javax.swing.border.AbstractBorder
|
+ -- javax.swing.border.BevelBorder
|
+ -- javax.swing.plaf.BorderUIResource.BevelBorderUIResource
public static class BorderUIResource.BevelBorderUIResource
    extends BevelBorder
    implements UIResource
```

### Constructors

```
BorderUIResource.BevelBorderUIResource (int bevelType)
```

```
BorderUIResource.BevelBorderUIResource (int bevelType, Color highlight,
Color shadow)
BorderUIResource.BevelBorderUIResource (int bevelType, Color highlightOuter,
Color highlightInner, Color shadowOuter, Color shadowInner)
```

## **BorderUIResource.CompoundBorderUIResource 类**

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.CompoundBorder
|
+-- javax.swing.plaf.BorderUIResource.CompoundBorderUIResource
public static class BorderUIResource.CompoundBorderUIResource
    extends CompoundBorder
    implements UIResource
```

### **Constructor**

```
BorderUIResource.CompoundBorderUIResource (Border outsideBorder,
Border insideBorder)
```

## **BorderUIResource.EmptyBorderUIResource 类**

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EmptyBorder
|
+-- javax.swing.plaf.BorderUIResource.EmptyBorderUIResource
public static class BorderUIResource.EmptyBorderUIResource
    extends EmptyBorder
    implements UIResource
```

### **Constructors**

```
BorderUIResource.EmptyBorderUIResource (Insets insets)
BorderUIResource.EmptyBorderUIResource (int top, int left, int bottom,
int right)
```

## **BorderUIResource.EtchedBorderUIResource 类**

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EtchedBorder
|
+-- javax.swing.plaf.BorderUIResource.EtchedBorderUIResource
public static class BorderUIResource.EtchedBorderUIResource
    extends EtchedBorder
    implements UIResource
```

## Constructors

```
BorderUIResource.EtchedBorderUIResource ()
BorderUIResource.EtchedBorderUIResource (Color highlight, Color shadow)
BorderUIResource.EtchedBorderUIResource (int etchType)
BorderUIResource.EtchedBorderUIResource (int etchType, Color highlight,
Color shadow)
```

## BorderUIResource.LineBorderUIResource 类

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.LineBorder
|
+-- javax.swing.plaf.BorderUIResource.LineBorderUIResource
public static class BorderUIResource.LineBorderUIResource
    extends LineBorder
    implements UIResource
```

## Constructors

```
BorderUIResource.LineBorderUIResource (Color color)
BorderUIResource.LineBorderUIResource (Color color, int thickness)
```

## BorderUIResource.MatteBorderUIResource 类

```
java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.EmptyBorder
|
+-- javax.swing.border.MatteBorder
|
+-- javax.swing.plaf.BorderUIResource.MatteBorderUIResource
public static class BorderUIResource.MatteBorderUIResource
    extends MatteBorder
    implements UIResource
```

## Constructors

```
BorderUIResource.MatteBorderUIResource (Icon tileIcon)
BorderUIResource.MatteBorderUIResource (int top, int left, int bottom,
int right, Color color)
BorderUIResource.MatteBorderUIResource (int top, int left, int bottom,
int right, Icon tileIcon)
```

## BorderUIResource.TitledBorderUIResource 类

```

java.lang.Object
|
+-- javax.swing.border.AbstractBorder
|
+-- javax.swing.border.TitledBorder
|
+-- javax.swing.plaf.BorderUIResource.TitledBorderUIResource
public static class BorderUIResource.TitledBorderUIResource
    extends TitledBorder
    implements UIResource

```

### Constructors

```

BorderUIResource.TitledBorderUIResource (Border border)
BorderUIResource.TitledBorderUIResource (Border border, String title)
BorderUIResource.TitledBorderUIResource (Border border, String title,
int titleJustification, int titlePosition)
BorderUIResource.TitledBorderUIResource (Border border, String title,
int titleJustification, int titlePosition, Font titleFont)
BorderUIResource.TitledBorderUIResource (Border border, String title,
int titleJustification, int titlePosition, Font titleFont, Color titleColor)
BorderUIResource.TitledBorderUIResource (String title)

```

## ButtonUI 类

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ButtonUI
public abstract class ButtonUI
    extends ComponentUI

```

### Constructor

```
ButtonUI ()
```

## ColorChooserUI 类

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ColorChooserUI
public abstract class ColorChooserUI
    extends ComponentUI

```

### Constructor

```
ColorChooserUI ()
```

## ColorUIResource 类

```
java.lang.Object
|
+-- java.awt.Color
|
+-- javax.swing.plaf.ColorUIResource
public class ColorUIResource
    extends Color
    implements UIResource
```

### Constructors

```
ColorUIResource (Color c)
ColorUIResource (float r, float g, float b)
ColorUIResource (int rgb)
ColorUIResource (int r, int g, int b)
```

## ComboBoxUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ComboBoxUI
public abstract class ComboBoxUI
    extends ComponentUI
```

### Constructor

```
ComboBoxUI ()
```

### Methods

abstract boolean	isFocusTraversable (JComboBox c)
abstract boolean	isPopupVisible (JComboBox c)
abstract void	setPopupVisible (JComboBox c, boolean v)

## ComponentUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
public abstract class ComponentUI
    extends Object
```

### Constructor

```
ComponentUI ()
```

### Methods

boolean	contains (JComponent c, int x, int y)
static ComponentUI	createUI (JComponent c)
Accessible	getAccessibleChild (JComponent c, int i)

```

int           getAccessibleChildrenCount (JComponent c)
Dimension    getMaximumSize (JComponent c)
Dimension    getMinimumSize (JComponent c)
Dimension    getPreferredSize (JComponent c)
void         installUI (JComponent c)
void         paint (Graphics g, JComponent c)
void         uninstallUI (JComponent c)
void         update (Graphics g, JComponent c)

```

## DesktopIconUI 类

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.DesktopIconUI
public abstract class DesktopIconUI
    extends ComponentUI

```

### Constructor

```
DesktopIconUI ()
```

## DesktopPaneUI 类

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.DesktopPaneUI
public abstract class DesktopPaneUI
    extends ComponentUI

```

### Constructor

```
DesktopPaneUI ()
```

## DimensionUIResource 类

```

java.lang.Object
|
+-- java.awt.geom.Dimension2D
|
+-- java.awt.Dimension
|
+-- javax.swing.plaf.DimensionUIResource
public class DimensionUIResource
    extends Dimension
    implements UIResource

```

### Constructor

```
DimensionUIResource (int width, int height)
```

## FileChooserUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.FileChooserUI
public abstract class FileChooserUI
    extends ComponentUI
```

### Constructor

```
FileChooserUI ()
```

### Methods

abstract void	ensureFileisVisible (JFileChooser fc, File f)
abstract FileFilter	getAcceptAllFileFilter (JFileChooser fc)
abstract String	getApproveButtonText (JFileChooser fc)
abstract String	getDialogTitle (JFileChooser fc)
abstract FileView	getFileView (JFileChooser fc)
abstract void	rescanCurrentDirectory (JFileChooser fc)

## FontUIResource 类

```
java.lang.Object
|
+-- java.awt.Font
|
+-- javax.swing.plaf.FontUIResource
public class FontUIResource
    extends Font
    implements UIResource
```

## IconUIResource 类

```
java.lang.Object
|
+-- javax.swing.plaf.IconUIResource
public class IconUIResource
    extends Object
    implements Icon, UIResource, Serializable
```

### Constructor

```
IconUIResource (Icon delegate)
```

### Methods

int	getIconHeight ()
int	getIconWidth ()
void	paintIcon (Component c, Graphics g, int x, int y)

## InsetsUIResource 类

```
java.lang.Object
|
+-- java.awt.Insets
|
+-- javax.swing.plaf.InsetsUIResource
public class InsetsUIResource
    extends Insets
    implements UIResource
```

### Constructor

```
InsetsUIResource (int top, int left, int bottom, int right)
```

## InternalFrameUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.InternalFrameUI
public abstract class InternalFrameUI
    extends ComponentUI
```

### Constructor

```
InternalFrameUI ()
```

## LabelUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.LabelUI
public abstract class LabelUI
    extends ComponentUI
```

### Constructor

```
LabelUI ()
```

## ListUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ListUI
public abstract class ListUI
    extends ComponentUI
```

### Constructor

```
ListUI ()
```

**Methods**

abstract Rectangle	getCellBounds (JList list, int index, int index2)
abstract Point	indexToLocation (JList list, int index)
abstract int	locationToIndex (JList list, Point location)

**MenuBarUI 类**

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.MenuBarUI
public abstract class MenuBarUI
    extends ComponentUI
```

**Constructor**

```
MenuBarUI ()
```

**MenuItemUI 类**

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ButtonUI
|
+-- javax.swing.plaf.MenuItemUI
public abstract class MenuItemUI
    extends ButtonUI
```

**Constructor**

```
MenuItemUI ()
```

**OptionPaneUI 类**

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plafOptionPaneUI
public abstract class OptionPaneUI
    extends ComponentUI
```

**Constructor**

```
OptionPaneUI ()
```

**Methods**

abstract boolean	containsCustomComponents ( JOptionPane op )
abstract void	selectInitialValue ( JOptionPane op )

## PanelUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.PanelUI
public abstract class PanelUI
    extends ComponentUI
```

### Constructor

```
PanelUI ()
```

## PopupMenuUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.PopupMenuUI
public abstract class PopupMenuUI
    extends ComponentUI
```

### Constructor

```
PopupMenuUI ()
```

## ProgressBarUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ProgressBarUI
public abstract class ProgressBarUI
    extends ComponentUI
```

### Constructor

```
ProgressBarUI ()
```

## ScrollBarUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ScrollBarUI
public abstract class ScrollBarUI
    extends ComponentUI
```

### Constructor

```
ScrollBarUI ()
```

## ScrollPaneUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ScrollPaneUI
public abstract class ScrollPaneUI
    extends ComponentUI
```

### Constructor

ScrollPaneUI ()

## SeparatorUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.SeparatorUI
public abstract class SeparatorUI
    extends ComponentUI
```

### Constructor

SeparatorUI ()

## SliderUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.SliderUI
public abstract class SliderUI
    extends ComponentUI
```

### Constructor

SliderUI ()

## ScrollPaneUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.SplitPaneUI
public abstract class SplitPaneUI
    extends ComponentUI
```

### Constructor

SplitPaneUI ()

**Methods**

```

abstract void finishedPaintingChildren (JSplitPane jc, Graphics g)
abstract int getDividerLocation (JSplitPane jc)
abstract int getMaximumDividerLocation (JSplitPane jc)
abstract int getMinimumDividerLocation (JSplitPane jc)
abstract void resetToPreferredSizes (JSplitPane jc)
abstract void setDividerLocation (JSplitPane jc, int location)

```

**TabbedPaneUI 类**

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.TabbedPaneUI
public abstract class TabbedPaneUI
    extends ComponentUI

```

**Constructor**

```
TabbedPaneUI ()
```

**Methods**

```

abstract Rectangle getTabBounds (JTabbedPane pane, int index)
abstract int getTabRunCount (JTabbedPane pane)
abstract int tabForCoordinate (JTabbedPane pane, int x, int y)

```

**TableHeaderUI 类**

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.TableHeaderUI
public abstract class TableHeaderUI
    extends ComponentUI

```

**Constructor**

```
TableHeaderUI ()
```

**TableUI 类**

```

java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.TableUI
public abstract class TableUI
    extends ComponentUI

```

**Constructor**

```
TableUI ()
```

## TextUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.TextUI
public abstract class TextUI
    extends ComponentUI
```

### Constructor

TextUI ()

### Methods

abstract void	damageRange (JTextComponent t, int p0, int p1)
abstract void	damageRange (JTextComponent t, int p0, int p1, Position.Bias firstBias, Position.Bias secondBias)
abstract EditorKit	getEditorKit (JTextComponent t)
abstract int	getNextVisualPositionFrom (JTextComponent t, nt pos, osition.Bias b, int cirection, Position.Bias [] biasRet)
abstract View	getRootView (JTextComponent t)
abstract Rectangle	modelToView (JTextComponent t, int pos)
abstract Rectangle	modelToView (JTextComponent t, int pos, Position.Bias bias)
abstract int	viewToModel (JTextComponent t, Point pt)
abstract int	viewToModel (JTextComponent t, Point pt, Position.Bias [] iasReturn)

## ToolBarUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ToolBarUI
public abstract class ToolBarUI
    extends ComponentUI
```

### Constructor

ToolBarUI ()

## ToolTipUI 类

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ToolTipUI
public abstract class ToolTipUI
    extends ComponentUI
```

**Constructor**

```
ToolTipUI ()
```

**TreeUI 类**

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.TreeUI
public abstract class TreeUI
    extends ComponentUI
```

**Constructor**

```
TreeUI ()
```

**Methods**

abstract void	cancelEditing (JTree tree)
abstract TreePath	getClosestPathForLocation (JTree tree, int x, int y)
abstract TreePath	getEditingPath (JTree tree)
abstract Rectangle	getPainBounds (JTree tree, TreePath path)
abstract TreePath	getPathForRow (JTree tree, int row)
abstract int	getRowCount (JTree tree)
abstract int	getRowForPath (JTree tree, TreePath path)
abstract boolean	isEditing (JTree tree)
abstract void	startEditingAtPath (JTree tree, TreePath path)
abstract boolean	stopEditing (JTree tree)

**ViewportUI 类**

```
java.lang.Object
|
+-- javax.swing.plaf.ComponentUI
|
+-- javax.swing.plaf.ViewportUI
public abstract class ViewportUI
    extends ComponentUI
```

**Constructor**

```
ViewportUI ()
```

**javax.swing.table****TableCellEditor 接口**

```
public abstract interface TableCellEditor
    extends CellEditor
```

这个接口描述了一个编辑器，该编辑器可以用来编辑 Swing 表格的单元。

**Method**

```
Component getTableCellEditorComponent (JTable table, Object value,
boolean isSelected, int row, int column)
```

**TableCellRenderer 接口**

```
public abstract interface TableCellRenderer
```

这个接口描述了 Swing 表格中单元的绘制程序。

**Method**

```
Component getTableCellRendererComponent (JTable table, Object value,
boolean isSelected, boolean hasFocus, int row, int column)
```

 **TableColumnModel 接口**

```
public abstract interface TableColumnModel
```

这个接口描述了表格中栏的数据模式。

**Methods**

void	addColumn (TableColumn aColumn)
void	addColumnModelListener (TableColumnModelListener x)
TableColumn	getColumn (int columnIndex)
int	getColumnCount ()
int	getColumnIndex (Object columnIdentifier)
int	getColumnIndexAtX (int xPosition)
int	getColumnMargin ()
Enumeration	getColumns ()
boolean	getColumnSelectionAllowed ()
int	getSelectedColumnCount ()
int []	getSelectedColumns ()
ListSelectionModel	getSelectionModel ()
int	getTotalColumnWidth ()
void	moveColumn (int columnIndex, int newIndex)
void	removeColumn (TableColumn column)
void	removeColumnModelListener (TableColumnModelListener x)
void	setColumnMargin (int newMargin)
void	setColumnSelectionAllowed (boolean flag)
void	setSelectionModel (ListSelectionModel newModel)

**TableModel 接口**

```
public abstract interface TableModel
```

这个接口描述了表格的数据模式。

**Methods**

```

void           addTableModelListener (TableModelListener l)
Class          getColumnClass (int columnIndex)
int            getColumnCount ()
String         getColumnName (int columnIndex)
int            getRowCount ()
Object         getValueAt (int rowIndex, int columnIndex)
boolean        isCellEditable (int rowIndex, int columnIndex)
void           removeTableModelListener (TableModelListener l)
void           setValueAt (Object aValue, int rowIndex, int columnIndex)

```

**AbstractTableModel 类**

```

java.lang.Object
|
+-- javax.swing.table.AbstractTableModel
public abstract class AbstractTableModel
    extends Object
    implements TableModel, Serializable

```

这个类通过实现表格模式来表示抽象表格模式。这个类需要实现下面的方法：

```

public int      getRowCount ();
public int      getColumnCount ();
public Object   getValueAt (int row, int column);

```

**Field**

```
protected EventListenerList           listenerList
```

**Constructor**

```
AbstractTableModel ()
```

**Methods**

```

void           addTableModelListener (TableModelListener l)
int            findColumn (String columnName)
void           fireTableCellUpdated (int row, int column)
void           fireTableChanged (TableModelEvent e)
void           fireTableDataChanged ()
void           fireTableRowsDeleted (int firstRow, int lastRow)
void           fireTableRowsInserted (int firstRow, int lastRow)
void           fireTableRowsUpdated (int firstRow, int lastRow)
void           fireTableStructureChanged ()
Class          getColumnClass (int columnIndex)
String         getColumnName (int column)
boolean        isCellEditable (int rowIndex, int columnIndex)
void           removeTableModelListener (TableModelListener l)
void           setValueAt (Object aValue, int rowIndex, int columnIndex)

```

## DefaultTableCellRenderer 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel
|
+-- javax.swing.table.DefaultTableCellRenderer
public class DefaultTableCellRenderer
    extends JLabel
    implements TableCellRenderer, Serializable

```

这个类表示缺省的单元绘制程序，用于显示表格的单元。

### Inner Class Summary

static class            DefaultTableCellRenderer.UIResource

### Field

protected static Border    noFocusBorder

### Constructor

DefaultTableCellRenderer ()

### Methods

Component	getTableCellRendererComponent (JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)
void	setBackground (Color c)
void	setForeground (Color c)
protected void	setValue (Object value)
void	updateUI ()

## DefaultTableCellRenderer.UIResource 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel
|
+-- javax.swing.table.DefaultTableCellRenderer
|
+-- javax.swing.table.DefaultTableCellRenderer.UIResource
public static class DefaultTableCellRenderer.UIResource
    extends DefaultTableCellRenderer
    implements UIResource

```

这个类通过实现 `UIResource` 来表示单元绘制程序的用户接口资源。

### **Constructor**

```
DefaultTableCellRenderer.UIResource ()
```

## **DefaultTableColumnModel 类**

```
java.lang.Object
|
+-- javax.swing.table.DefaultTableColumnModel
public class DefaultTableColumnModel
extends Object
implements TableColumnModel, PropertyChangeListener,
ListSelectionListener, Serializable
```

这个类表示 Swing 表格栏模式的标准实现。

### **Fields**

protected ChangeEvent	changeEvent
protected int	columnMargin
protected boolean	columnSelectionAllowed
protected EventListenerList	listenerList
protected ListSelectionModel	selectionModel
protected Vector	tableColumns
protected int	totalColumnWidth

### **Constructor**

```
DefaultTableColumnModel ()
```

### **Methods**

void	addColumn (TableColumn aColumn)
void	addColumnModelListener (TableColumnModelListener x)
protected ListSelectionModel	createSelectionModel ()
protected void	fireColumnAdded (TableColumnModelEvent e)
protected void	fireColumnMarginChanged ()
protected void	fireColumnMoved (TableColumnModelEvent e)
protected void	fireColumnRemoved (TableColumnModelEvent e)
protected void	fireColumnSelectionChanged (ListSelectionEvent e)
TableColumn	getColumn (int columnIndex)
int	getColumnCount ()
int	getColumnIndex (Object identifier)
int	getColumnIndexAtX (int xPosition)
int	getColumnMargin ()
Enumeration	getColumns ()
boolean	getColumnSelectionAllowed ()

```

int           getSelectedColumnCount ()
int []        getSelectedColumns ()
ListSelectionModel  getSelectionModel ()
int           getTotalColumnWidth ()
void          moveColumn (int columnIndex, int newIndex)
void          propertyChange (PropertyChangeEvent evt)
protected void  recalcWidthCache ()
void          removeColumn (TableColumn column)
void          removeColumnModelListener
              (TableColumnModelListener x)
void          setColumnMargin (int newMargin)
void          setColumnSelectionAllowed (boolean flag)
void          setSelectionModel (ListSelectionModel newModel)
void          valueChanged (ListSelectionEvent e)

```

## DefaultTableModel 类

```

java.lang.Object
|
+-- javax.swing.table.AbstractTableModel
|
+-- javax.swing.table.DefaultTableModel
public class DefaultTableModel
    extends AbstractTableModel
    implements Serializable

```

这个类表示表格数据模式的标准实现。

### Fields

protected Vector	columnIdentifiers
Vector	dataVector

### Constructors

DefaultTableModel ()	
DefaultTableModel (int numRows, int numColumns)	
DefaultTableModel (Object [][] data, Object [] columnNames)	
DefaultTableModel (Object [] columnNames, int numRows) 0	
DefaultTableModel (Vector columnNames, int numRows)	
DefaultTableModel (Vector data, Vector columnNames)	

### Methods

void	addColumn (Object columnName)
void	addColumn (Object columnName, Object [] columnData)
void	addColumn (Object columnName, Vector columnData)
void	addRow (Object [] rowData)
void	addRow (Vector rowData)
protected static Vector	convertToVector (Object [] anArray)

```

protected static Vector convertToVector (Object [][] anArray)
int getCount ()
String getColumnNames (int column)
Vector getDataVector ()
int getRowCount ()
Object getValueAt (int row, int column)
void insertRow (int row, Object [] rowData)
void insertRow (int row, Vector rowData)
boolean isCellEditable (int row, int column)
void moveRow (int startIndex, int endIndex, int toIndex)
void newDataAvailable (TableModelEvent event)
void newRowsAdded (TableModelEvent event)
void removeRow (int row)
void rowsRemoved (TableModelEvent event)
void setColumnIdentifiers (Object [] newIdentifiers)
void setColumnIdentifiers (Vector newIdentifiers)
void setDataVector (Object [] [] newData,
Object [] columnNames)
void setDataVector (Vector newData, Vector columnNames)
void setNumRows (int newSize)
void setValueAt (Object aValue, int row, int column)

```

## JTableHeader 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.table.JTableHeader

public class JTableHeader
    extends JComponent
    implements TableColumnModelListener, Accessible

```

这个类表示 Swing 表格的列标题。

### Inner Class Summary

```
protected class JTableHeader.AccessibleJTableHeader
```

### Fields

protected TableColumnModel	columnModel
protected TableColumn	draggedColumn
protected int	draggedDistance
protected boolean	reorderingAllowed

protected boolean	resizingAllowed
Protected TableColumn	resizingColumn
protected JTable	table
protected boolean	updateTableInRealTime

## Constructors

JTableHeader ()  
 JTableHeader (TableColumnModel cm)

## Methods

void	columnAdded (TableColumnModelEvent e)
int	columnAtPoint (Point point)
void	columnMarginChanged (ChangeEvent e)
void	columnMoved (TableColumnModelEvent e)
void	columnRemoved (TableColumnModelEvent e)
void	columnSelectionChanged (ListSelectionEvent e)
protected TableColumnModel	createDefaultColumnModel ()
AccessibleContext	getAccessibleContext ()
TableColumnModel	getColumnModel ()
TableColumn	getDraggedColumn ()
int	getDraggedDistance ()
Rectangle	getHeaderRect (int columnIndex)
boolean	getReorderingAllowed ()
boolean	getResizingAllowed ()
TableColumn	getResizingColumn ()
JTable	getTable ()
String	getToolTipText (MouseEvent event)
TableHeaderUI	getUI ()
String	getUIClassID ()
boolean	getUpdateTableInRealTime ()
protected void	initializeLocalVars ()
protected String	paramString ()
void	resizeAndRepaint ()
void	setColumnModel (TableColumnModel newModel)
void	setDraggedColumn (TableColumn aColumn)
void	setDraggedDistance (int distance)
void	setReorderingAllowed (boolean b)
void	setResizingAllowed (boolean b)
void	setResizingColumn (TableColumn aColumn)
void	setTable (JTable aTable)
void	setUI (TableHeaderUI ui)
void	setUpdateTableInRealTime (boolean flag)
void	updateUI ()

## TableColumn 类

```

java.lang.Object
|
+-- javax.swing.table.TableColumn
public class TableColumn
    extends Object
    implements Serializable

```

这个类表示表格的列，它具有表格列的所有属性。

### Fields

static String	CELL_RENDERER_PROPERTY
protected TableCellEditor	cellEditor
protected TableCellRenderer	cellRenderer
static String	COLUMN_WIDTH_PROPERTY
static String	HEADER_RENDERER_PROPERTY
static String	HEADER_VALUE_PROPERTY
protected TableCellRenderer	headerRenderer
protected Object	HeaderValue
protected Object	identifier
protected boolean isResizable	Resizable flag
protected int	maxWidth
protected int	minWidth
protected int	modelIndex
protected int	resizedPostponingDisableCount
protected int	width

### Constructors

TableColumn ()	
TableColumn (int modelIndex)	
TableColumn (int modelIndex, int width)	
TableColumn (int modelIndex, int width, TableCellRenderer cellRenderer,	
TableCellEditor cellEditor)	
TableColumn with modelIndex	

### Methods

void	addPropertyChangeListener (PropertyChangeListener listener)
protected TableCellRenderer	createDefaultHeaderRenderer ()
void	disableResizing ()
void	enableResizing ()
TableCellEditor	getCellEditor ()
TableCellRenderer	getCellRenderer ()
TableCellRenderer	getHeaderRenderer ()
Object	getHeaderValue ()

```

Object           getIdentifier ()
int             getMaxWidth ()
int             getMinWidth ()
int             getModelIndex ()
int             getPreferredWidth ()
boolean        getResizable ()
int             getWidth ()
void            removePropertyChangeListener
                (PropertyChangeListener listener)
void            setCellEditor (TableCellEditor aEditor)
void            setCellRenderer (TableCellRenderer aRenderer)
void            setHeaderRenderer (TableCellRenderer aRenderer)
void            setHeaderValue (Object aValue)
void            setIdentifier (Object anIdentifier)
void            setMaxWidth (int maxWidth)
void            setMinWidth (int minWidth)
void            setModelIndex (int anIndex)
void            setPreferredSize (int preferredWidth)
void            setResizable (boolean flag)
void            setWidth (int width)
void            sizeWidthToFit ()

```

## **javax.swing.text**

### **AbstractDocument.AttributeContext 接口**

```
public abstract static interface AbstractDocument.AttributeContext
```

这个接口描述了一个环境对象，它允许添加或删除易变的属性集。

#### **Methods**

AttributeSet	addAttribute (AttributeSet old, Object name, Object value)
AttributeSet	addAttributes (AttributeSet old, AttributeSet attrs)
AttributeSet	getEmptySet ()
void	reclaim (AttributeSet a)
AttributeSet	removeAttribute (AttributeSet old, Object name)
AttributeSet	removeAttributes (AttributeSet old, AttributeSet attrs)

### **AbstractDocument.Content 接口**

```
public abstract static interface AbstractDocument.Content
```

这个接口描述文档内信息的内容。

#### **Methods**

Position	createPosition (int offset)
----------	-----------------------------

void	getChars ( int where, int len, Segment txt)
String	getString ( int where, int len)
UndoableEdit	insertString ( int where, String str)
int	length ()
UndoableEdit	remove ( int where, int nitems)

## AttributeSet 接口

```
public abstract interface AttributeSet
```

这个接口表示只读属性集。属性集基本上是关键字与相关值的集合。

### Inner Classes

static interface	AttributeSet.CharacterAttribute
static interface	AttributeSet.ColorAttribute
static interface	AttributeSet.FontAttribute
static interface	AttributeSet.ParagraphAttribute

### Fields

static Object	NameAttribute
static Object	ResolveAttribute

### Methods

boolean	containsAttribute (Object name, Object value)
boolean	containsAttributes (AttributeSet attributes)
AttributeSet	copyAttributes ()
Object	getAttribute (Object key)
int	getAttributeCount ()
Enumeration	getAttributeNames ()
AttributeSet	getResolveParent ()
boolean	isDefined (Object attrName)
boolean	isEqual (AttributeSet attr)

## AttributeSet.CharacterAttribute 接口

```
public abstract static interface AttributeSet.CharacterAttribute
```

这个接口描述了类型签名，它需要与属性键相关联来提供字符内容。

## AttributeSet.ColorAttribute 接口

```
public abstract static interface AttributeSet.ColorAttribute
```

这个接口描述了类型签名，它需要与属性键相关联来提供颜色。

## AttributeSet.FontAttribute 接口

```
public abstract static interface AttributeSet.FontAttribute
```

这个接口描述了类型签名，它需要与属性键相关联来提供文本的字体。

## AttributeSet.ParagraphAttribute 接口

```
public abstract static interface AttributeSet.ParagraphAttribute
```

这个接口描述了类型签名，它需要与属性键相关联来提供一段内容。

## Caret 接口

```
public abstract interface Caret
```

这个接口描述了插入光标的特征，插入光标用于在文档视图中插入文本。

### Methods

void	addChangeListener (ChangeListener l)
void	deinstall (JTextComponent c)
int	getBlinkRate ()
int	getDot ()
Point	getMagicCaretPosition ()
int	getMark ()
void	install (JTextComponent c)
boolean	isSelectionVisible ()
boolean	isVisible ()
void	moveDot (int dot)
void	paint (Graphics g)
void	removeChangeListener (ChangeListener l)
void	setBlinkRate (int rate)
void	setDot (int dot)
void	setMagicCaretPosition (Point p)
void	setSelectionVisible (boolean v)
void	setVisible (boolean v)

## Document 接口

```
public abstract interface Document
```

这个接口描述了文档的特征。该文档是用于文本组件的内容模式。

### Fields

```
static String StreamDescriptionProperty  
static String TitleProperty
```

### Methods

void	addDocumentListener (DocumentListener listener)
void	addUndoableEditListener (UndoableEditListener listener)
Position	createPosition (int offs)
Element	getDefaultRootElement ()
Position	getEndPosition ()
int	getLength ()

object	getProperty (Object key)
Element []	getRootElement ()
Position	getStartPosition ()
String	getText (int offset, int length)
void	getText (int offset, int length, Segment txt)
void	insertString (int offset, String str, AttributeSet a)
void	putProperty (Object key, Object value)
void	remove (int offs, int len)
void	removeDocumentListener (DocumentListener listener)
void	removeUndoableEditListener (UndoableEditListener listener)
void	render (Runnable r)

## Element 接口

```
public abstract interface Element;
```

这个接口描述了文档中的一个元素。

### Methods

AttributeSet	getAttributes ()
Document	getDocument ()
Element	getElement (int index)
int	getElementCount ()
int	getElementIndex (int offset)
int	getEndOffset ()
String	getName ()
Element	getParentElement ()
int	getStartOffset ()
boolean	isLeaf ()

## Highlighter 接口

```
public abstract interface Highlighter;
```

这个接口描述了一个对象，该对象允许用指定的颜色加亮显示所做的选择。

### Inner Classes

static interface	Highlighter.Highlight
static interface	Highlighter.HighlightPainter

### Methods

Object	addHighlight (int p0, int p1, Highlighter.HighlightPainter p)
void	changeHighlight (Object tag, int p0, int p1)
void	deinstall (JTextComponent c)
Highlighter.Highlight []	getHighlights ()
void	install (JTextComponent c)

---

```

void           paint (Graphics g)
void           removeAllHighlights ()
void           removeHighlight (Object tag)

```

## **Highlighter.Highlight 接口**

```
public abstract static interface Highlighter.Highlight
```

这个接口描述的对象包含一个高亮度的指定值（参看其接口方法）。

### **Methods**

int	getEndOffset ()
Highlighter.HighlightPainter	getPainter ()
int	getStartOffset ()

## **Highlighter.HighlightPainter 接口**

```
public abstract static interface Highlighter.HighlightPainter
```

这个接口描述了用来加亮显示一定内容的绘制程序。

### **Method**

void	paint (Graphics g, int p0, int pl, Shape bounds, JTextComponent c)
------	---

## **Keymap 接口**

```
public abstract interface Keymap
```

这个接口描述了键盘映射关系（名字 - 值的配对）。

### **Methods**

void	addActionForKeyStroke (KeyStroke key, Action a)
Action	getAction (KeyStroke key)
Action []	getBoundActions ()
KeyStroke []	getBoundKeyStrokes ()
Action	getDefaultAction ()
KeyStroke []	getKeyStrokesForAction (Action a)
String	getName ()
Keymap	getResolveParent ()
boolean	isLocallyDefined (KeyStroke key)
void	removeBindings ()
void	removeKeyStrokeBinding (KeyStroke keys)
void	setDefaultAction (Action a)
void	setResolveParent (Keymap parent)

## **MutableAttributeSet 接口**

```
public abstract interface MutableAttributeSet
```

---

```
extends AttributeSet
```

这个接口表示一组可改变的唯一属性。

### Methods

void	addAttribute (Object name, Object value)
void	addAttributes (AttributeSet attributes)
void	removeAttribute (Object name)
void	removeAttributes (AttributeSet attributes)
void	removeAttributes (Enumeration names)
void	setResolveParent (AttributeSet parent)

## Position 接口

```
public abstract interface Position
```

这个接口表示文档中的位置。

### Inner Class

```
static class Position.Bias
```

### Method

```
int getOffset ()
```

## Style 接口

```
public abstract interface Style
```

```
extends MutableAttributeSet
```

这个接口描述了与文档中某个元素相关的属性集合。

### Methods

void	addChangeListener (ChangeListener l)
String	getName ()
void	removeChangeListener (ChangeListener l)

## StyledDocument 接口

```
public abstract interface StyledDocument
```

```
extends Document
```

这个接口表示具有某种样式的文档。

### Methods

Style	addStyle (String nm, Style parent)
Color	getBackground (AttributeSet attr)
Element	getCharacterElement (int pos)
Font	getFont (AttributeSet attr)
Color	getForeground (AttributeSet attr)
Style	getLogicalStyle (int p)

```

Element    getParagraphElement ( int pos )
Style      getStyle ( String nm )
void       removeStyle ( String nm )
void       setCharacterAttributes ( int offset, int length,
                                    AttributeSet s, boolean replace )
void       setLogicalStyle ( int pos, Style s )
void       setParagraphAttributes ( int offset, int length,
                                    AttributeSet s, boolean replace )

```

## TabableView 接口

```
public abstract interface TabableView
```

这个接口描述了一种视图，该视图的大小依赖于标签的大小。

### Methods

```

float     getPartialSpan ( int p0, int p1 )
float     getTabbedSpan ( float x, TabExpander e )

```

## TabExpander 接口

```
public abstract interface TabExpander
```

这个接口描述了一个对象，该对象实现了对一个标签的扩展。

### Method

```
float nextTabStop ( float x, int tabOffset )
```

## ViewFactory 接口

```
public abstract interface ViewFactory
```

这个接口表示在文档指定部分创建视图的工厂。

### Method

```
View     create ( Element elem )
```

## AbstractDocument 类

```

java.lang.Object
|
+
+ -- javax.swing.text.AbstractDocument
public abstract class AbstractDocument
    extends Object
    implements Document, Serializable

```

这个类表示接口 Document 的抽象实现。

### Inner Classes

class	AbstractDocument.AbstractElement
static interface	AbstractDocument.AttributeContext

---

class	AbstractDocument.BranchElement
static interface	AbstractDocument.Content
class	AbstractDocument.DefaultDocumentEvent
static class	AbstractDocument.ElementEdit
class	AbstractDocument.LeafElement

**Fields**

protected static String	BAD_LOCATION
static String	BidiElementName
static String	ContentElementName
static String	ElementNameAttribute
protected EventListenerList	listenerList
static String	ParagraphElementName
static String	SectionElementName

**Constructors**

protected	AbstractDocument (AbstractDocument.Content data)
protected	AbstractDocument (AbstractDocument.Content data, AbstractDocument.AttributeContext context)

**Methods**

void	addDocumentListener (DocumentListener listener)
void	addUndoableEditListener (UndoableEditListener listener)
protected Element	createBranchElement (Element parent, AttributeSet a)
protected Element	createLeafElement (Element parent, AttributeSet a, int p0, int p1)
Position	createPosition (int offs)
void	dump (PrintStream out)
protected void	fireChangedUpdate (DocumentEvent e)
protected void	fireInsertUpdate (DocumentEvent e)
protected void	fireRemoveUpdate (DocumentEvent e)
protected void	fireUndoableEditUpdate (UndoableEditEvent e)
int	getAynchronousLoadPriority ()
protected AbstractDocument.AttributeContext	getAttributeContext ()
Element	getBidiRootElement ()
protected AbstractDocument.Content	getContent ()
protected Thread	getCurrentWriter ()
abstract Element	getDefaultRootElement ()

```

Dictionary           getDocumentProperties ()
Position            getEndPosition ()
int                getLength ()
abstract Element   getParagraphElement (int pos)
Object              getProperty (Object key)
Element []          getRootElement ()
Position            getStartPosition ()
String              getText (int offset, int length)
void               getText (int offset, int length,
                           Segment txt)
void               insertString (int offs,
                                 String str, AttributeSet a)
protected void      insertUpdate
                    (AbstractDocument,
                     DefaultDocumentEvent chng,
                     AttributeSet attr)
protected void      postRemoveUpdate
                    (AbstractDocument,
                     DefaultDocumentEvent chng)
void               putProperty (Object key,
                               Object value)
void               readLock ()
void               readUnlock ()
void               remove (int offs, int len)
void               removeDocumentListener
                    (DocumentListener listener)
void               removeUndoableEditListener
                    (UndoableEditListener listener)
protected void      removeUpdate (AbstractDocument,
                                 DefaultDocumentEvent chng)
void               render (Runnable r)
void               setAsynchronousLoadPriority
                    (int p)
void               setDocumentProperties
                    (Dictionary x)
protected void      writeLock ()
protected void      writeUnlock ()

```

### AbstractDocument.ElementEdit 类

```

java.lang.Object
+
+ -- javax.swing.undo.AbstractUndoableEdit
|
+ -- javax.swing.text.AbstractDocument.ElementEdit
public static class AbstractDocument.ElementEdit
    extends AbstractUndoableEdit
    implements DocumentEvent.ElementChange

```

这个类表示对文档中某元素执行编辑操作。

### **Constructor**

```
AbstractDocument.ElementEdit (Element e, int index, Element [] removed,
Element [] added)
```

### **Methods**

Element []	getChildrenAdded ()
Element []	getChildrenRemoved ()
Element	getElement ()
int	getIndex ()
void	redo ()
void	undo ()

## **AbstractWriter 类**

```
java.lang.Object
|
+-- javax.swing.text.AbstractWriter
public abstract class AbstractWriter
    extends Object
```

这个类表示一个抽象的书写器，它实际书写了文档中元素的层次结构。

### **Field**

```
protected static char NEWLINE
```

### **Constructors**

protected	AbstractWriter (Writer w, Document doc)
protected	AbstractWriter (Writer w, Document doc, int pos, int len)
protected	AbstractWriter (Writer w, Element root)
protected	AbstractWriter (Writer w, Element root, int pos, int len)

### **Methods**

protected void	decrIndent ()
protected Document	getDocument ()
protected ElementIterator	getElementIterator ()
protected String	getText (Element elem)
protected void	incrIndent ()
protected void	indent ()
protected boolean	inRange (Element next)
protected void	setIndentSpace (int space)
protected void	setLineLength (int l)
protected void	text (Element elem)
protected abstract void	write ()
protected void	write (char ch)
protected void	write (String str)

```
protected void writeAttributes (AttributeSet attr)
```

## BoxView 类

```
java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
public class BoxView
    extends CompositeView
```

这个接口表示文本模式的视图，此文本模式将其子类放进一个方框里。

### Constructor

```
BoxView (Element elem, int axis)
```

### Methods

protected void	baselineLayout (int targetSpan, int axis, int [] offsets, int [] spans)
protected SizeRequirements	baselineRequirements (int axis, SizeRequirements r)
protected SizeRequirements	calculateMajorAxisRequirements (int axis, SizeRequirements r)
protected SizeRequirements	calculateMinorAxisRequirements (int axis, SizeRequirements r)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
protected void	childAllocation (int index, Rectangle alloc)
protected boolean	flipEastAndWestAtEnds (int position, Position.Bias bias)
float	getAlignment (int axis)
int	getHeight ()
float	getMaximumSpan (int axis)
float	getMinimumSpan (int axis)
protected int	getOffset (int axis, int childIndex)
float	getPreferredSize (int axis)
int	getResizeWeight (int axis)
protected int	getSpan (int axis, int childIndex)
protected View	getViewAtPoint (int x, int y, Rectangle alloc)
int	getWidth ()
void	insertUpdate (DocumentEvent e, Shape a, ViewFactory f)
protected boolean	isAfter (int x, int y, Rectangle innerAlloc)
protected boolean	isAllocationValid ()

```

protected boolean           isBefore (int x, int y, Rectangle innerAlloc)
protected void              layout (int width, int height)
protected void              layoutMajorAxis (int targetSpan, int axis,
                                             int [] offsets, int [] spans)
protected void              layoutMinorAxis (int targetSpan, int axis,
                                              int [] offsets, int [] spans)
Shape                      modelToView (int pos, Shape a, Position.Bias b)
void                       paint (Graphics g, Shape allocation)
protected void              paintChild (Graphics g, Rectangle alloc,
                                         int index)
void                       preferenceChanged (View child, boolean width,
                                              boolean height)
void                       removeUpdate (DocumentEvent e, Shape a,
                                         ViewFactory f)
void                       replace (int offset, int length, View [] elems)
void                       setSize (float width, float height)
int                        viewToModel (float x, float y, Shape a,
                                         Position.Bias [] bias)

```

## ComponentView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.ComponentView
public class ComponentView
    extends View

```

这个接口代表一个组件装饰器，该组件装饰器实现了视图接口。

### Constructor

```
ComponentView (Element elem)
```

### Methods

protected Component	createComponent ()
float	getAlignment (int axis)
Component	getComponent ()
float	getMaximumSpan (int axis)
float	getMinimumSpan (int axis)
float	getPreferredSpan (int axis)
Shape	modelToView (int pos, Shape a, Position.Bias b)
void	paint (Graphics g, Shape a)
void	setParent (View p)
void	setSize (float width, float height)
int	viewToModel (float x, float y, Shape a,                            Position.Bias [] bias)

## CompositeView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
public abstract class CompositeView
    extends View

```

这个接口表示文本模式的复合视图，该文本模式将其子类放进一个方框里。

### Constructor

```
CompositeView (Element elem)
```

### Methods

void	append (View v)
protected abstract void	childAllocation (int index, Rectangle a)
protected boolean	flipEastAndWestAtEnds (int position, Position.Bias bias)
protected short	getBottomInset ()
Shape	getChildAllocation (int index, Shape a)
protected Rectangle	getInsideAllocation (Shape a)
protected short	getLeftInset ()
protected int	getNextEastWestVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
protected int	getNextNorthSouthVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
int	getNextVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
protected short	getRightInset ()
protected short	getTopInset ()
View	getView (int n)
protected abstract View	getViewAtPoint (int x, int y, Rectangle alloc)
protected View	getViewAtPosition (int pos, Rectangle a)
int	getViewCount ()
protected int	getViewIndexAtPosition (int pos)
void	insert (int offs, View v)
protected abstract boolean	isAfter (int x, int y, Rectangle alloc)
protected abstract boolean	isBefore (int x, int y, Rectangle alloc)
protected void	loadChildren (ViewFactory f)
Shape	modelToView (int p0, Position.Bias b0, int p1,

```

Shape
void
void
protected void
protected void
void
int
Position.Bias bi, Shape a)
modelToView (int pos, Shape a, Position.Bias b)
removeAll ()
replace (int offset, int length, View [] views)
setInsets (short top, short left, short bottom,
short right)
setParagraphInsets (AttributeSet attr)
setParent (View parent)
viewToModel (float x, float y, Shape a,
Position.Bias [] bias)

```

## DefaultCaret 类

```

java.lang.Object
+
+ -- java.awt.geom.RectangularShape
|
+ -- java.awt.geom.Rectangle2D
|
+ -- java.awt.Rectangle
|
+ -- javax.swing.text.DefaultCaret
public class DefaultCaret
    extends Rectangle
    implements Caret, FocusListener, MouseListener,
    MouseMotionListener

```

这个类表示 Caret 的缺省实现，它实际上是一小段以指定频率闪动的垂直线段。

### Fields

protected ChangeEvent	changeEvent
protected EventListenerList	listenerList

### Constructor

DefaultCaret ()

### Methods

void	addChangeListener (ChangeListener l)
protected void	adjustVisibility (Rectangle nloc)
protected void	damage (Rectangle r)
void	deinstall (JTextComponent c)
protected void	fireStateChanged ()
void	focusGained (FocusEvent e)
void	focusLost (FocusEvent e)
int	getBlinkRate ()
protected JTextComponent	getComponent ()
int	getDot ()
Point	getMagicCaretPosition ()

```

int                           getMark ()
protected Highlighter.HighlightPainter getSelectionPainter ()
void                          install (JTextComponent c)
boolean                      isSelectionVisible ()
boolean                      isVisible ()
void                          mouseClicked (MouseEvent e)
void                          mouseDragged (MouseEvent e)
void                          mouseEntered (MouseEvent e)
void                          mouseExited (MouseEvent e)
void                          mouseMoved (MouseEvent e)
void                          mousePressed (MouseEvent e)
void                          mouseReleased (MouseEvent e)
protected void                moveCaret (MouseEvent e)
void                          moveDot (int dot)
void                          paint (Graphics g)
protected void                positionCaret (MouseEvent e)
void                          removeChangeListener
                             (ChangeListener l)
protected void                repaint ()
void                          setBlinkRate (int rate)
void                          setDot (int dot)
void                          setMagicCaretPosition (Point p)
void                          setSelectionVisible (boolean vis)
void                          setVisible (boolean e)
String                        toString ()

```

## DefaultEditorKit 类

```

java.lang.Object
|
+-- javax.swing.text.EditorKit
|
+-- javax.swing.text.DefaultEditorKit
public class DefaultEditorKit
    extends EditorKit

```

这个类表示纯文本的缺省编辑器工具箱，其中只有纯文本，可以对该文本进行剪切、复制和粘贴等操作。

### Inner Classes

static class	DefaultEditorKit.BeepAction
static class	DefaultEditorKit.CopyAction
static class	DefaultEditorKit.CutAction
static class	DefaultEditorKit.DefaultKeyTypedAction
static class	DefaultEditorKit.InsertBreakAction
static class	DefaultEditorKit.InsertContentAction
static class	DefaultEditorKit.InsertTabAction

static class DefaultEditorKit.PasteAction

### Fields

static String backwardAction  
static String beepAction  
static String beginAction  
static String beginLineAction  
static String beginParagraphAction  
static String beginWordAction  
static String copyAction  
static String cutAction  
static String defaultKeyTypedAction  
static String deleteNextCharAction  
static String deletePrevCharAction  
static String downAction  
static String endAction  
static String endLineAction  
static String EndOfStringProperty  
static String endParagraphAction  
static String endWordAction  
static String forwardAction  
static String insertBreakAction  
static String insertContentAction  
static String insertTabAction  
static String nextWordAction  
static String pageDownAction  
static String pageUpAction  
static String pasteAction  
static String previousWordAction  
static String readOnlyAction  
static String selectAllAction  
static String selectionBackwardAction  
static String selectionBeginAction  
static String selectionBeginLineAction  
static String selectionBeginParagraphAction  
static String selectionBeginWordAction  
static String selectionDownAction  
static String selectionEndAction  
static String selectionEndLineAction  
static String selectionEndParagraphAction  
static String selectionEndWordAction  
static String selectionForwardAction  
static String selectionNextWordAction  
static String selectionPreviousWordAction  
static String selectionUpAction  
static String selectLineAction  
static String selectParagraphAction  
static String selectWordAction  
static String upAction

```
static String    writableAction
```

### Constructor

```
DefaultEditorKit ()
```

### Methods

Object:	clone ()
Caret:	createCaret ()
Document:	createDefaultDocument ()
Action []:	getActions ()
String:	getContent-Type ()
ViewFactory:	getViewFactory ()
void:	read (InputStream in, Document doc, int pos)
void:	read (Reader in, Document doc, int pos)
void:	write (OutputStream out, Document doc, int pos, int len)
void:	write (Writer out, Document doc, int pos, int len)

## DefaultEditorKit.BeepAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.BeepAction
public static class DefaultEditorKit.BeepAction
    extends TextAction
```

这个类表示创建蜂鸣声的动作。

### Constructor

```
DefaultEditorKit.BeepAction ()
```

### Method

```
void actionPerformed (ActionEvent e)
```

## DefaultEditorKit.CopyAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.CopyAction
public static class DefaultEditorKit.CopyAction
    extends TextAction
```

这个类表示一个动作，它复制选定的区域并将其内容放在系统剪贴板上。

### Constructor

```
DefaultEditorKit.CopyAction ()
```

**Method**

```
void actionPerformed (ActionEvent e)
```

**DefaultEditorKit.CutAction 类**

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.CutAction
public static class DefaultEditorKit.CutAction
    extends TextAction
```

这个类表示一个动作，它剪切选定的区域并将其内容放在系统剪贴板上。

**Constructor**

```
DefaultEditorKit.CutAction ()
```

创建带有相应标识符的这个对象。

**Method**

```
void actionPerformed (ActionEvent e)
```

**DefaultEditorKit.DefaultKeyTypedAction 类**

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.DefaultKeyTypedAction
public static class DefaultEditorKit.DefaultKeyTypedAction
    extends TextAction
```

这个类表示一个动作对象；假如一个击键事件没有接收任何键盘映射输入，就缺省地执行该动作对象。

**Constructor**

```
DefaultEditorKit.DefaultKeyTypedAction ()
```

**Method**

```
void actionPerformed (ActionEvent e)
```

**DefaultEditorKit.InsertBreakAction 类**

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.InsertBreakAction
public static class DefaultEditorKit.InsertBreakAction
    extends TextAction
```

这个类表示一个对象，它将一行或一段分隔符放入文档中。如果没有任何选择的话，在加入分隔符之前，将删除该对象。

#### Constructor

```
DefaultEditorKit.InsertBreakAction ()
```

#### Method

```
void actionPerformed (ActionEvent e)
```

### DefaultEditorKit.InsertContentAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.InsertContentAction
public static class DefaultEditorKit.InsertContentAction
    extends TextAction
```

这个类表示一个动作对象，它将信息内容放在文档里。

#### Constructor

```
DefaultEditorKit.InsertContentAction ()
```

#### Method

```
void actionPerformed (ActionEvent e)
```

### DefaultEditorKit.InsertTabAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.InsertTabAction
public static class DefaultEditorKit.InsertTabAction
    extends TextAction
```

这个类表示一个动作对象，它将制表符放在文档里。

#### Constructor

```
DefaultEditorKit.InsertTabAction ()
```

#### Method

```
void actionPerformed (ActionEvent e)
```

## DefaultEditorKit.PasteAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.DefaultEditorKit.PasteAction
public static class DefaultEditorKit.PasteAction
    extends TextAction

```

这个类表示粘贴动作，它将系统剪贴板上的内容粘贴到选定的区域内。假如没有选定内容的话，就把剪贴板内容放在光标之前。

### Constructor

```
DefaultEditorKit.PasteAction ()
```

### Method

```
void actionPerformed (ActionEvent e)
```

## DefaultHighlighter 类

```

java.lang.Object
|
+-- javax.swing.text.LayeredHighlighter
|
+-- javax.swing.text.DefaultHighlighter
public class DefaultHighlighter
    extends LayeredHighlighter

```

这个类表示加亮区的缺省实现。此加亮区用纯色绘制。

### Inner Class

```
static class DefaultHighlighter.DefaultHighlightPainter
```

### Field

```
static LayeredHighlighter.LayerPainter
```

### Constructor

```
DefaultHighlighter ()
```

### Methods

Object	addHighlight (int p0, int p1, Highlighter.HighlightPainter p)
void	changeHighlight (Object tag, int p0, int p1)
void	deinstall (JTextComponent c)
boolean	drawsLayeredHighlights ()
Highlighter.Highlight []	getHighlights ()
void	install (JTextComponent c)

```

void           paint (Graphics g)
void           paintLayeredHighlights (Graphics g, int p0, int p1,
Shape viewBounds, JTextComponent editor, View view)
void           removeAllHighlights ()
void           removeHighlight (Object tag)
void           setDrawsLayeredHighlights (boolean newValue)

```

## DefaultHighlighter.DefaultHighlightPainter 类

```

java.lang.Object
|
+-- javax.swing.text.LayeredHighlighter.LayerPainter
|
+-- javax.swing.text.DefaultHighlighter.DefaultHighlightPainter
public static class DefaultHighlighter.DefaultHighlightPainter
    extends LayeredHighlighter.LayerPainter

```

这个类表示缺省的加亮绘图器，它用纯色填充加亮的区域。

### Constructor

```
DefaultHighlighter.DefaultHighlightPainter (Color c)
```

### Methods

```

Color   getColor ()
void    paint (Graphics g, int offs0, int offs1, Shape bounds,
              JTextComponent c)
Shape   paintLayer (Graphics g, int offs0, int offs1, Shape bounds,
                    JTextComponent c, View view)

```

## DefaultStyledDocument 类

```

java.lang.Object
|
+-- javax.swing.text.AbstractDocument
|
+-- javax.swing.text.DefaultStyledDocument
public class DefaultStyledDocument
    extends AbstractDocument
    implements StyledDocument

```

这个类表示具有一定样式的文档，它能够类似于 Rich Text 格式那样对字符和段落进行标记。

### Inner Classes

```

static class    DefaultStyledDocument.AttributeUndoableEdit
class          DefaultStyledDocument.ElementBuffer
static class   DefaultStyledDocument.ElementSpec
protected class DefaultStyledDocument.SectionElements

```

### Fields

```
protected DefaultStyledDocument.ElementBuffer      buffer
```

static int BUFFER\_SIZE\_DEFAULT

### Constructors

DefaultStyledDocument ()  
DefaultStyledDocument (AbstractDocument.Content c, StyleContext styles)  
DefaultStyledDocument (StyleContext styles)

### Methods

void	addDocumentListener (DocumentListener listener)
Style	addStyle (String nm, Style parent)
protected void	create (DefaultStyledDocument. ElementSpec [] data)
protected AbstractDocument.AbstractElement	createDefaultRoot ()
Color	getBackground (AttributeSet attr)
Element	getCharacterElement (int pos)
Element	getDefaultRootElement ()
Font	getFont (AttributeSet attr)
Color	getForeground (AttributeSet attr)
Style	getLogicalStyle (int p)
Element	getParagraphElement (int pos)
Style	getStyle (String nm)
Enumeration	getStyleNames ()
protected void	insert (int offset, DefaultStyledDocument. ElementSpec [] data)
protected void	insertUpdate (AbstractDocument. DefaultDocumentEvent chng, AttributeSet attr)
void	removeDocumentListener (DocumentListener listener)
void	removeStyle (String nm)
protected void	removeUpdate (AbstractDocument. DefaultDocumentEvent chng)
void	setCharacterAttributes (int offset, int length, AttributeSet s, boolean replace)
void	setLogicalStyle (int pos, Style s)
void	setParagraphAttributes (int offset, int length, AttributeSet s, boolean replace)
protected void	styleChanged (Style style)

## DefaultStyledDocument.AttributeUndoableEdit 类

```
java.lang.Object
|
+-- javax.swing.undo.AbstractUndoableEdit
    |
    +-- javax.swing.text.DefaultStyledDocument.AttributeUndoableEdit
public static class DefaultStyledDocument.AttributeUndoableEdit
    extends AbstractUndoableEdit
```

这个类表示可撤消的编辑操作，它用记忆元素属性集内所做的改变。

### Fields

protected AttributeSet	copy
protected Element	element
protected boolean	isReplacing
protected AttributeSet	newAttributes

### Constructor

```
DefaultStyledDocument.AttributeUndoableEdit (Element element,
AttributeSet newAttributes, boolean isReplacing)
```

### Methods

```
void redo ()
void undo ()
```

## DefaultStyledDocument.ElementSpec 类

```
java.lang.Object
|
+-- javax.swing.text.DefaultStyledDocument.ElementSpec
public static class DefaultStyledDocument.ElementSpec
extends Object
```

这个类表示建立文档元素的规范。

### Fields

static short	ContentType
static short	EndTagType
static short	JoinFractureDirection
static short	JoinNextDirection
static short	JoinPreviousDirection
static short	OriginateDirection
static short	StartTagType

### Constructors

```
DefaultStyledDocument.ElementSpec (AttributeSet a, short type)
DefaultStyledDocument.ElementSpec (AttributeSet a, short type, char [] txt,
int offs, int len)
```

`DefaultStyledDocument.ElementSpec (AttributeSet a, short type, int len)`

### Methods

<code>char []</code>	<code>getArray ()</code>
<code>AttributeSet</code>	<code>getAttributes ()</code>
<code>short</code>	<code>getDirection ()</code>
<code>int</code>	<code>getLength ()</code>
<code>int</code>	<code>getOffset ()</code>
<code>short</code>	<code>getType ()</code>
<code>void</code>	<code>setDirection (short direction)</code>
<code>void</code>	<code>setType (short type)</code>
<code>String</code>	<code>toString ()</code>

## ElementIterator 类

```
java.lang.Object
|
+-- javax.swing.text.ElementIterator
public class ElementIterator
    extends Object
    implements Cloneable
```

这个类表示元素迭代器，它实现元素树的迭代。

### Constructors

```
ElementIterator (Document document)
ElementIterator (Element root)
```

### Methods

<code>Object</code>	<code>clone ()</code>
<code>Element</code>	<code>current ()</code>
<code>int</code>	<code>depth ()</code>
<code>Element</code>	<code>first ()</code>
<code>Element</code>	<code>next ()</code>
<code>Element</code>	<code>previous ()</code>

## FieldView 类

```
java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.PlanView
|
+-- javax.swing.text.FieldView
public class FieldView
    extends PlainView
```

这个类编辑单行纯文本的视图。

**Constructor**

```
FieldView (Element elem)
```

**Methods**

protected Shape	adjustAllocation (Shape a)
protected FontMetrics	getFontMetrics ()
float	getPreferredSpan (int axis)
int	getResizeWeight (int axis)
void	insertUpdate (DocumentEvent changes, Shape a, ViewFactory f)
Shape	modelToView (int pos, Shape a, Position.Bias b)
void	paint (Graphics g, Shape a)
void	removeUpdate (DocumentEvent changes, Shape a, ViewFactory f)
int	viewToModel (float fx, float fy, Shape a, Position.Bias [] bias)

**GapContent 类**

```
java.lang.Object
|
+-- javax.swing.text.GapVector
|
+-- javax.swing.text.GapContent
public class GapContent
    extends javax.swing.text.GapVector
    implements AbstractDocument.Content, Serializable
```

这个类表示 AbstractDocument.Content 接口的实现，它以此表示空白内容，就像在 emacs 编辑器中所见的情况一样。

**Constructors**

```
GapContent ()
GapContent (int initialLength)
```

**Methods**

protected Object	allocateArray (int len)
Position	createPosition (int offset)
protected int	getArrayLength ()
void	getChars (int where, int len, Segment chars)
protected Vector	getPositionsInRange (Vector v, int offset, int length)
String	getString (int where, int len)
UndoableEdit	insertString (int where, String str)
int	length ()
UndoableEdit	remove (int where, int nitems)
protected void	resetMarksAtZero ()

```

protected void      shiftEnd ( int newSize )
protected void      shiftGap ( int newGapStart )
protected void      shiftGapEndUp ( int newGapEnd )
protected void      shiftGapStartDown ( int newGapStart )
protected void      updateUndoPositions ( Vector positions, int offset,
                                         int length )

```

## IconView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.IconView
public class IconView
    extends View

```

这个类表示装饰图标的视图，它实现了 View 接口。

### Constructor

```
IconView (Element elem)
```

### Methods

```

float      getAlignment ( int axis )
float      getPreferredSpan ( int axis )
Shape      modelToView ( int pos, Shape a, Position.Bias b )
void       paint ( Graphics g, Shape a )
void       setSize ( float width, float height )
int        viewToModel ( float x, float y, Shape a, Position.Bias [] bias )

```

## JTextComponent 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.text.JTextComponent
public abstract class JTextComponent
    extends JComponent
    implements Scrollable, Accessible

```

这个类表示 Swing 中使用的所有文本组件的父类。

### Inner Classes

```

class      JTextComponent.AccessibleJTextComponent
static class  JTextComponent.KeyBinding

```

## 从 javax.swing.JComponent 继承的内部类

JComponent.AccessibleJComponent

### Fields

static String DEFAULT\_KEYMAP  
static String FOCUS\_ACCELERATOR\_KEY

### Constructor

JTextComponent ()

### Methods

void addCaretListener (CaretListener listener)  
static Keymap addKeymap (String nm, Keymap parent)  
void copy ()  
void cut ()  
protected void fireCaretUpdate (CaretEvent e)  
AccessibleContext getAccessibleContext ()  
Action [] getActions ()  
Caret getCaret ()  
Color getCaretColor ()  
int getCaretPosition ()  
Color getDisabledTextColor ()  
Document getDocument ()  
char getFocusAccelerator ()  
Highlighter getHighlighter ()  
Keymap getKeymap ()  
static Keymap getKeymap (String nm)  
Insets getMargin ()  
Dimension getPreferredScrollableViewportSize ()  
int getScrollableBlockIncrement (Rectangle visibleRect,  
int orientation, int direction)  
boolean getScrollableTracksViewportHeight ()  
boolean getScrollableTracksViewportWidth ()  
int getScrollableUnitIncrement (Rectangle visibleRect,  
int orientation, int direction)  
String getSelectedText ()  
Color getSelectedTextColor ()  
Color getSelectionColor ()  
int getSelectionEnd ()  
int getSelectionStart ()  
String getText ()  
String getText (int offs, int len)  
TextUI getUI ()  
boolean isEditable ()

```

boolean      isFocusTraversable ()
boolean      isOpaque ()
static void   loadKeymap (Keymap map,
                           JTextComponent.KeyBinding [] bindings, Action [] actions)
Rectangle    modelToView (int pos)
void         moveCaretPosition (int pos)
protected String paramString ()
void         paste ()
protected void processComponentKeyEvent (KeyEvent e)
void         read (Reader in, Object desc)
void         removeCaretListener (CaretListener listener)
static Keymap removeKeymap (String rm)
void         removeNotify ()
void         replaceSelection (String content)
void         select (int selectionStart, int selectionEnd)
void         selectAll ()
void         setCaret (Caret c)
void         setCaretColor (Color c)
void         setCaretPosition (int position)
void         setDisabledTextColor (Color c)
void         setDocument (Document doc)
void         setEditable (boolean b)
void         setEnabled (boolean b)
void         setFocusAccelerator (char aKey)
void         setHighlighter (Highlighter h)
void         setKeymap (Keymap map)
void         setMargin (Insets m)
void         setOpaque (boolean o)
void         setSelectedTextColor (Color c)
void         setSelectionColor (Color c)
void         setSelectionEnd (int selectionEnd)
void         setSelectionStart (int selectionStart)
void         setText (String t)
void         setUI (TextUI ui)
void         updateUI ()
int          viewToModel (Point pt)
void         write (Writer out)

```

### JTextComponent.KeyBinding 类

```

java.lang.Object
+
+-- javax.swing.text.JTextComponent.KeyBinding
public static class JTextComponent.KeyBinding
    extends Object

```

这个类表示创建键盘绑定 (key binding) 关系的目录。

**Fields**

String	actionName
KeyStroke	key

**Constructor**

`JTextComponent.KeyBinding (KeyStroke key, String actionName)`

**LabelView 类**

```
java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.LabelView
public class LabelView
    extends View
    implements TabableView
```

这个类表示样式文本的标签视图。

**Constructor**

`LabelView (Element elem)`

**Methods**

View	breakView (int axis, int p0, float pos, float len)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
View	createFragment (int p0, int p1)
float	getAlignment (int axis)
int	getBreakWeight (int axis, float pos, float len)
protected Font	getFont ()
protected FontMetrics	getFontMetrics ()
int	getNextVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
float	getPartialSpan (int p0, int p1)
float	getPreferredSpan (int axis)
float	getTabbedSpan (float x, TabExpander e)
void	insertUpdate (DocumentEvent e, Shape a, ViewFactory t)
Shape	modelToView (int pos, Shape a, Position.Bias b)
void	paint (Graphics g, Shape a)
protected void	setPropertiesFromAttributes ()
protected void	setStrikeThrough (boolean s)
protected void	setSubscript (boolean s)
protected void	setSuperscript (boolean s)
protected void	setUnderline (boolean u)
int	viewToModel (float x, float y, Shape a, Position.Bias [] biasReturn)

## LabelView2D 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.LabelView2D
public class LabelView2D
    extends View

```

这个类表示二维的标签视图。

### Constructor

```
LabelView2D (Element elem)
```

### Methods

View	breakView (int axis, int p0, float pos, float len)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
View	createFragment (int p0, int p1)
float	getAlignment (int axis)
int	getBreakWeight (int axis, float pos, float len)
protected Font	getFont ()
protected FontMetrics	getFontMetrics ()
int	getNextVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
float	getPreferredSpan (int axis)
void	insertUpdate (DocumentEvent e, Shape a, ViewFactory f)
Shape	modelToView (int pos, Shape a, Position.Bias b)
void	paint (Graphics g, Shape a)
void	removeUpdate (DocumentEvent changes, Shape a, ViewFactory f)
protected void	setPropertiesFromAttributes ()
protected void	setStrikeThrough (boolean s)
protected void	setSubscript (boolean s)
protected void	setSuperscript (boolean s)
protected void	setUnderline (boolean u)
String	toString ()
int	viewToModel (float x, float y, Shape a, Position.Bias [] biasReturn)

## LayeredHighlighter 类

```

java.lang.Object
|
+-- javax.swing.text.LayeredHighlighter
public abstract class LayeredHighlighter
    extends Object
    implements Highlighter

```

这个类表示分层次的高亮选择器，它由象标签视图这样的视图对象所使用。

**Inner Class**

```
static class LayeredHighlighter.LayerPainter
```

**Constructor**

```
LayeredHighlighter()
```

**Method**

```
abstract void paintLayeredHighlights (Graphics g, int p0, int p1,
Shape viewBounds, JTextComponent editor, View view)
```

**LayeredHighlighter.LayerPainter 类**

```
java.lang.Object
|
+
+-- javax.swing.text.LayeredHighlighter.LayerPainter
public abstract static class LayeredHighlighter.LayerPainter
    extends Object
    implements Highlighter.HighlightPainter
```

这个类表示分层加亮绘制程序。

**Constructor**

```
LayeredHighlighter.LayerPainter()
```

**Method**

```
abstract Shape paintLayer (Graphics g, int p0, int p1, Shape viewBounds,
JTextComponent editor, View view)
```

**ParagraphView 类**

```
java.lang.Object
|
+
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.ParagraphView
public class ParagraphView
    extends BoxView
    implements TabExpander
```

这个类表示一种段落的视图，这种段落由具有多种字体和颜色的字符行、组件及嵌入其中的图标组合而成。

**Field**

```
protected int firstLineIndent
```

**Constructor**

```
ParagraphView (Element elem)
```

**Methods**

protected void	adjustRow (javax.swing.text.ParagraphView.Row r, int desiredSpan, int x)
View	breakView (int axis, float len, Shape a)
protected SizeRequirements	calculateMinorAxisRequirements (int axis, SizeRequirements r)
void	changedUpdate (DocumentEvent changes, Shape a, ViewFactory f)
protected int	findOffsetToCharactersInString (char [] string, int start)
protected boolean	flipEastAndWestAtEnds (int position, Position.Bias bias)
float	getAlignment (int axis)
int	getBreakWeight (int axis, float len)
protected int	getClosestPositionTo (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet, int rowIndex, int x)
protected View	getLayoutView (int index)
protected int	getLayoutViewCount ()
protected int	getNextNorthSouthVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
protected float	getPartialSize (int startOffset, int endOffset)
protected float	getTabBase ()
protected TabSet	getTabSet ()
protected View	getViewAtPosition (int pos, Rectangle a)
protected int	getViewIndexAtPosition (int pos)
void	insertUpdate (DocumentEvent changes, Shape a, ViewFactory f)
protected void	layout (int width, int height)
protected void	loadChildren (ViewFactory f)
float	nextTabStop (float x, int tabOffset)
void	paint (Graphics g, Shape a)
void	removeUpdate (DocumentEvent changes, Shape a, ViewFactory f)
protected void	setFirstLineIndent (float fi)
protected void	setJustification (int j)
protected void	setLineSpacing (float ls)
protected void	setPropertiesFromAttributes ()

## PasswordField 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.PlainView
|
+-- javax.swing.text.FieldView
|
+-- javax.swing.text.PasswordView
public class PasswordView
    extends FieldView

```

这个类表示口令字段的视图。

### Constructor

```
PasswordField (Element elem)
```

### Methods

protected int	drawEchoCharacter (Graphics g, int x, int y, char c)
protected int	drawSelectedText (Graphics g, int x, int y, int p0, int p1)
protected int	drawUnselectedText (Graphics g, int x, int y, int p0, int p1)
Shape	modelToView (int pos, Shape a, Position.Bias b)
int	viewToModel (float fx, float fy, Shape a, Position.Bias [] bias)

## PlainDocument 类

```

java.lang.Object
|
+-- javax.swing.text.AbstractDocument
|
+-- javax.swing.text.PlainDocument
public class PlainDocument
    extends AbstractDocument

```

这个类表示纯文档，该纯文档的元素结构如文本中的行。

### Fields

static String	lineLimitAttribute
static String	tabSizeAttribute

### Constructors

```

PlainDocument ()
protected PlainDocument (AbstractDocument.Content c)

```

### Methods

protected AbstractDocument.AbstractElement	createDefaultRoot ()
Element	getDefaultRootElement ()

```

Element           getParagraphElement (int pos)
protected void    insertUpdate (AbstractDocument.
                               DefaultDocumentEvent chng,
                               AttributeSet attr)
protected void    removeUpdate (AbstractDocument.
                               DefaultDocumentEvent chng)

```

## PlainView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.PlainView
public class PlainView
    extends View
    implements TabExpander

```

这个类表示简单的多行文本的视图。

### Field

```
protected FontMetrics metrics
```

### Constructor

```
PlainView (Element elem)
```

### Methods

void	changedUpdate (DocumentEvent changes, Shape a, ViewFactory f)
protected void	drawLine (int lineIndex, Graphics g, int x, int y)
protected int	drawSelectedText (Graphics g, int x, int y, int p0, int p1)
protected int	drawUnselectedText (Graphics g, int x, int y, int p0, int p1)
protected Segment	getLineBuffer ()
float	getPreferredSpan (int axis)
protected int	getTabSize ()
void	insertUpdate (DocumentEvent changes, Shape a, ViewFactory f)
Shape	modelToView (int pos, Shape a, Position.Bias b)
float	nextTabStop (float x, int tabOffset)
void	paint (Graphics g, Shape a)
void	preferenceChanged (View child, boolean width, boolean height)
void	removeUpdate (DocumentEvent changes, Shape a, ViewFactory f)
int	viewToModel (float fx, float fy, Shape a, Position.Bias [] bias)

## Position.Bias 类

```
java.lang.Object
|
+-- javax.swing.text.Position.Bias
public static final class Position.Bias
    extends Object
```

这个类表示枚举对象，它显示该对象在模型中的偏移位置。

### Fields

```
static Position.Bias Backward
static Position.Bias Forward
```

### Method

```
String    toString ()
```

## Segment 类

```
java.lang.Object
|
+-- javax.swing.text.Segment
public class Segment
    extends Object
```

这个类表示一段字符数组。这一段则代表文本的片段。

### Fields

char []	array
int	count
int	offset

### Constructors

```
Segment ()
Segment (char [] array, int offset, int count)
```

### Method

```
String    toString ()
```

## SimpleAttributeSet 类

```
java.lang.Object
|
+-- javax.swing.text.SimpleAttributeSet
public class SimpleAttributeSet
    extends Object
    implements MutableAttributeSet, Serializable, Cloneable
```

这个类用哈希表来表示易变属性集的简单实现。

**Field**

```
static AttributeSet EMPTY
```

**Constructors**

```
SimpleAttributeSet()
SimpleAttributeSet(AttributeSet source)
```

**Methods**

void	addAttribute (Object name, Object value)
void	addAttributes (AttributeSet attributes)
Object	clone ()
boolean	containsAttribute (Object name, Object value)
boolean	containsAttributes (AttributeSet attributes)
AttributeSet	copyAttributes ()
boolean	equals (Object obj)
Object	getAttribute (Object name)
int	getAttributeCount ()
Enumeration	getAttributeNames ()
AttributeSet	getResolveParent ()
int	hashCode ()
boolean	isDefined (Object attrName)
boolean	isEmpty ()
boolean	isEqual (AttributeSet attr)
void	removeAttribute (Object name)
void	removeAttributes (AttributeSet attributes)
void	removeAttributes (Enumeration names)
void	setResolveParent (AttributeSet parent)
String	toString ()

**StringContent 类**

```
java.lang.Object
|
+
+ -- javax.swing.text.StringContent
public final class StringContent
    extends Object
    implements AbstractDocument.Content, Serializable
```

这个类表示字符串内容的实现，这种字符串内容适合小文档。

**Constructors**

```
StringContent()
StringContent (int initialLength)
```

**Methods**

```
Position createPosition (int offset)
```

void	getChars ( int where, int len, Segment chars)
protected Vector	getPositionsInRange (Vector v, int offset, int length)
String	getString ( int where, int len)
UndoableEdit	insertString ( int where, String str)
int	length ()
UndoableEdit	remove ( int where, int nitems)
protected void	updateUndoPositions (Vector positions)

## StyleConstants 类

```
java.lang.Object
+
+
+ -- javax.swing.text.StyleConstants
public class StyleConstants
    extends Object
```

这个类表示通常用于样式常量的集合对象。

### Inner Classes

static class	StyleConstants.CharacterConstants
static class	StyleConstants.ColorConstants
static class	StyleConstants.FontConstants
static class	StyleConstants.ParagraphConstants

### Fields

static int	ALIGN_CENTER
static int	ALIGN_JUSTIFIED
static int	ALIGN_LEFT
static int	ALIGN_RIGHT
static Object	Alignment
static Object	Background
static Object	BidiLevel
static Object	Bold
static Object	ComponentAttribute
static String	ComponentElementName
static Object	ComposedTextAttribute
static Object	FirstLineIndent
static Object	FontFamily
static Object	FontSize
static Object	Foreground
static Object	IconAttribute
static String	IconElementName
static Object	Italic
static Object	LeftIndent
static Object	LineSpacing
static Object	ModelAttribute
static Object	NameAttribute
static Object	Orientation

static Object	ResolveAttribute
static Object	RightIndent
static Object	SpaceAbove
static Object	SpaceBelow
static Object	StrikeThrough
static Object	Subscript
static Object	Superscript
static Object	TabSet
static Object	Underline

**Methods**

static int	getAlignment (AttributeSet a)
static Color	getBackground (AttributeSet a)
static int	getBidiLevel (AttributeSet a)
static Component	getComponent (AttributeSet a)
static float	getFirstLineIndent (AttributeSet a)
static String	getFontFamily (AttributeSet a)
static int	getFontSize (AttributeSet a)
static Color	getForeground (AttributeSet a)
static Icon	getIcon (AttributeSet a)
static float	getLeftIndent (AttributeSet a)
static float	getLineSpacing (AttributeSet a)
static float	getRightIndent (AttributeSet a)
static float	getSpaceAbove (AttributeSet a)
static float	getSpaceBelow (AttributeSet a)
static TabSet	getTabSet (AttributeSet a)
static boolean	isBold (AttributeSet a)
static boolean	isItalic (AttributeSet a)
static boolean	isStrikeThrough (AttributeSet a)
static boolean	isSubscript (AttributeSet a)
static boolean	isSuperscript (AttributeSet a)
static boolean	isUnderline (AttributeSet a)
static void	setAlignment (MutableAttributeSet a, int align)
static void	setBackground (MutableAttributeSet a, Color fg)
static void	setBidiLevel (MutableAttributeSet a, int o)
static void	setBold (MutableAttributeSet a, boolean b)
static void	setComponent (MutableAttributeSet a, Component c)
static void	setFirstLineIndent (MutableAttributeSet a, float i)
static void	setFontFamily (MutableAttributeSet a, String fam)
static void	setFontSize (MutableAttributeSet a, int s)
static void	setForeground (MutableAttributeSet a, Color fg)
static void	setIcon (MutableAttributeSet a, Icon c)
static void	setItalic (MutableAttributeSet a, boolean b)
static void	setLeftIndent (MutableAttributeSet a, float i)
static void	setLineSpacing (MutableAttributeSet a, float i)
static void	setRightIndent (MutableAttributeSet a, float i)
static void	setSpaceAbove (MutableAttributeSet a, float i)

```

static void      setSpaceBelow (MutableAttributeSet a, float i)
static void      setStrikeThrough (MutableAttributeSet a, boolean b)
static void      setSubscript (MutableAttributeSet a, boolean b)
static void      setSuperscript (MutableAttributeSet a, boolean b)
static void      setTabSet (MutableAttributeSet a, TabSet tabs)
static void      setUnderline (MutableAttributeSet a, boolean b)
String          toString ()

```

## StyleConstants.CharacterConstants 类

```

java.lang.Object
|
+-- javax.swing.text.StyleConstants
|
+-- javax.swing.text.StyleConstants.CharacterConstants
public static class StyleConstants.CharacterConstants
    extends StyleConstants
    implements AttributeSet.CharacterAttribute

```

这个类表示字符常量的集合对象，如大小、字体类型等等。

### Fields

static Object	Background
static Object	BidiLevel
static Object	Bold
static Object	ComponentAttribute
static Object	Family
static Object	Foreground
static Object	IconAttribute
static Object	Italic
static Object	Size
static Object	StrikeThrough
static Object	Subscript
static Object	Superscript
static Object	Underline

## StyleConstants.ColorConstants 类

```

java.lang.Object
|
+-- javax.swing.text.StyleConstants
|
+-- javax.swing.text.StyleConstants.ColorConstants
public static class StyleConstants.ColorConstants
    extends StyleConstants
    implements AttributeSet.ColorAttribute,
               AttributeSet.CharacterAttribute

```

这个类表示颜色常量的集合对象。

**Fields**

static Object	Background
static Object	Foreground

**StyleConstants.FontConstants 类**

```
java.lang.Object
|
+-- javax.swing.text.StyleConstants
|
+-- javax.swing.text.StyleConstants.FontConstants
public static class StyleConstants.FontConstants
    extends StyleConstants
    implements AttributeSet.FontAttribute,
               AttributeSet.CharacterAttribute
```

这个类表示字体常量的集合对象。

**Fields**

static Object	Bold
static Object	Family
static Object	Italic
static Object	Size

**StyleConstants.ParagraphConstants 类**

```
java.lang.Object
|
+-- javax.swing.text.StyleConstants
|
+-- javax.swing.text.StyleConstants.ParagraphConstants
public static class StyleConstants.ParagraphConstants
    extends StyleConstants implements AttributeSet.ParagraphAttribute
```

这个类表示段落常量的集合对象。

**Fields**

static Object	Alignment
static Object	FirstLineIndent
static Object	LeftIndent
static Object	LineSpacing
static Object	Orientation
static Object	RightIndent
static Object	SpaceAbove
static Object	SpaceBelow
static Object	TabSet

## StyleContext 类

```

java.lang.Object
|
+
+-- javax.swing.text.StyleContext
public class StyleContext
    extends Object
    implements Serializable, AbstractDocument.AttributeContext
这个类表示样式内容。

```

### Inner Classes

```

class StyleContext.NamedStyle
class DWStyleContext.SmallAttributeSet

```

### Field

```
static String DEFAULT_STYLE
```

### Constructor

```
StyleContext()
```

### Methods

AttributeSet	addAttribute (AttributeSet old, Object name, Object value)
AttributeSet	addAttributes (AttributeSet old, AttributeSet attr)
void	addChangeListener (ChangeListener l)
Style	addStyle (String nm, Style parent)
protected MutableAttributeSet	createLargeAttributeSet (AttributeSet a)
protected StyleContext.SmallAttributeSet	createSmallAttributeSet (AttributeSet a)
Color	getBackground (AttributeSet attr)
protected int	getCompressionThreshold ()
static StyleContext	getDefaultStyleContext ()
AttributeSet	getEmptySet ()
Font	getFont (AttributeSet attr)
Font	getFont (String family, int style, int size)
FontMetrics	getFontMetrics (Font f)
Color	getForeground (AttributeSet attr)
static Object	getStaticAttribute (Object key)
static Object	getStaticAttributeKey (Object key)
Style	getStyle (String nm)
Enumeration	getStyleNames ()
void	readAttributes

```

        (ObjectInputStream in,
        MutableAttributeSet a)
readAttributeSet
        (ObjectInputStream in,
        MutableAttributeSet a)
reclaim (AttributeSet a)
registerStaticAttributeKey
        (Object key)
removeAttribute (AttributeSet old,
Object name)
removeAttributes (AttributeSet old,
AttributeSet attrs)
removeAttributes (AttributeSet old,
Enumeration names)
removeChangeListener
        (ChangeListener l)
removeStyle (String nm)
toString ()
writeAttributes
        (ObjectOutputStream out,
AttributeSet a)
writeAttributeSet
        (ObjectOutputStream out,
AttributeSet a)

```

### StyledEditorKit 类

```

java.lang.Object
|
+-- javax.swing.text.EditorKit
|
+-- javax.swing.text.DefaultEditorKit
|
+-- javax.swing.text.StyledEditorKit
public class StyledEditorKit
    extends DefaultEditorKit

```

这个类表示一个风格编辑器工具箱，它提供了风格文本的基本编辑特征。

#### Inner Classes

static class	StyledEditorKit.AlignmentAction
static class	StyledEditorKit.BoldAction
static class	StyledEditorKit.FontFamilyAction
static class	StyledEditorKit.FontSizeAction
static class	StyledEditorKit.ForegroundAction
static class	StyledEditorKit.ItalicAction
static class	StyledEditorKit.StyledTextAction
static class	StyledEditorKit.UnderlineAction

**Constructor**

```
StyledEditorKit ()
```

**Methods**

Object	clone ()
Document	createDefaultDocument ()
protected void	createInputAttributes (Element element, MutableAttributeSet set)
void	deinstall (JEditorPane c)
Action []	getActions ()
Element	getCharacterAttributeRun ()
MutableAttributeSet	getInputAttributes ()
ViewFactory	getViewFactory ()
void	install (JEditorPane c)

**StyledEditorKit.AlignmentAction 类**

```
java.lang.Object
|
+
+ -- javax.swing.AbstractAction
|
|
+ -- javax.swing.text.TextAction
|
|
+ -- javax.swing.text.StyledEditorKit.StyledTextAction
|
|
+ -- javax.swing.text.StyledEditorKit.AlignmentAction
public static class StyledEditorKit.AlignmentAction
    extends StyledEditorKit.StyledTextAction
```

这个类表示设置段落对齐方式的动作。

**Constructor**

```
StyledEditorKit.AlignmentAction (String nm, int a)
```

**Method**

```
void actionPerformed (ActionEvent e)
```

**StyledEditorKit.BoldAction 类**

```
java.lang.Object
|
|
+ -- javax.swing.AbstractAction
|
|
+ -- javax.swing.text.TextAction
|
|
+ -- javax.swing.text.StyledEditorKit.StyledTextAction
|
|
+ -- javax.swing.text.StyledEditorKit.BoldAction
public static class StyledEditorKit.BoldAction
    extends StyledEditorKit.StyledTextAction
```

这个类表示一个切换粗体属性的动作对象。

**Constructor**

```
StyledEditorKit.BoldAction ()
```

**Method**

```
void actionPerformed (ActionEvent e)
```

**StyledEditorKit.FontFamilyAction 类**

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.FontFamilyAction
public static class StyledEditorKit.FontFamilyAction
    extends StyledEditorKit.StyledTextAction
```

这个类表示设置字体族的动作。

**Constructor**

```
StyledEditorKit.FontFamilyAction (String nm, String family)
```

**Method**

```
void actionPerformed (ActionEvent e)
```

**StyledEditorKit.FontSizeAction 类**

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.FontSizeAction
public static class StyledEditorKit.FontSizeAction
    extends StyledEditorKit.StyledTextAction
```

这个类表示给字体大小赋值的动作对象。

**Constructor**

```
StyledEditorKit.FontSizeAction (String nm, int size)
```

**Method**

```
void actionPerformed (ActionEvent e)
```

## StyledEditorKit.ForegroundAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.ForegroundAction
public static class StyledEditorKit.ForegroundAction
    extends StyledEditorKit.StyledTextAction

```

这个类表示给前景颜色赋值的动作。

### Constructor

```
StyledEditorKit.ForegroundAction (String nm, Color fg)
```

### Method

```
void actionPerformed (ActionEvent e)
```

## StyledEditorKit.ItalicAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.ItalicAction
public static class StyledEditorKit.ItalicAction
    extends StyledEditorKit.StyledTextAction

```

这个类表示一个切换斜体属性的动作。

### Constructor

```
StyledEditorKit.ItalicAction ()
```

### Method

```
void actionPerformed (ActionEvent e)
```

## StyledEditorKit.StyledTextAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
public abstract static class StyledEditorKit.StyledTextAction
    extends TextAction

```

这个类表示一个动作对象，它包含一些用来改变字符或段落级别属性的很方便的方法。

### Constructor

```
StyledEditorKit.StyledTextAction (String nm)
```

### Methods

protected JEditorPane	getEditor (ActionEvent e)
protected StyledDocument	getStyledDocument (JEditorPane e)
protected StyledEditorKit	getStyledEditorKit (JEditorPane e)
protected void	setCharacterAttributes (JEditorPane editor, AttributeSet attr, boolean replace)
protected void	setParagraphAttributes (JEditorPane editor, AttributeSet attr, boolean replace)

## StyledEditorKit.UnderlineAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.StyledEditorKit.UnderlineAction
public static class StyledEditorKit.UnderlineAction
    extends StyledEditorKit.StyledTextAction
```

这个类表示一个切换下划线属性的动作对象

### Constructor

```
StyledEditorKit.UnderlineAction ()
```

### Method

```
void actionPerformed (ActionEvent e)
```

## TableView 类

```
java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.TableView
public abstract class TableView
    extends BoxView
```

这个类是 View 接口的实现形式，它代表一个表格。

**Inner Classes**

```
class TableView.TableCell
class TableView.TableRow
```

**Constructor**

```
TableView (Element elem)
```

**Methods**

protected SizeRequirements	calculateMinorAxisRequirements (int axis, SizeRequirements r)
protected TableView.TableCell	createTableCell (Element elem)
protected TableView.TableRow	createTableRow (Element elem)
protected View	getViewAtPosition (int pos, Rectangle a)
protected void	layoutColumns (int targetSpan, int [] offsets, int [] spans, SizeRequirements [] reqs)
protected void	layoutMinorAxis (int targetSpan, int axis, int [] offsets, int [] spans)
protected void	loadChildren (ViewFactory f)

**TabSet 类**

```
java.lang.Object
|
+-- javax.swing.text.TabSet
public class TabSet
    extends Object
    implements Serializable
```

这个类表示一个制表符的集合，它由若干个制表符停止位组合而成。

**Constructor**

```
TabSet (TabStop [] tabs)
```

**Methods**

TabStop	getTab (int index)
TabStop	getTabAfter (float location)
int	getTabCount ()
int	getTabIndex (TabStop tab)
int	getTabIndexAfter (float location)
String	toString ()

**TabStop 类**

```
java.lang.Object
|
+-- javax.swing.text.TabStop
public class TabStop
    extends Object
    implements Serializable
```

这个类表示一个制表停止符。

### Fields

```
static int ALIGN_BAR
static int ALIGN_CENTER
static int ALIGN_DECIMAL
static int ALIGN_LEFT
static int ALIGN_RIGHT
static int LEAD_DOTS
static int LEAD_EQUALS
static int LEAD_HYPHENS
static int LEAD_NONE
static int LEAD_THICKLINE
static int LEAD_UNDERLINE
```

### Constructors

```
TabStop (float pos)
TabStop (float pos, int align, int leader)
```

### Methods

```
boolean equals (Object other)
int getAlignment ()
int getLeader ()
float getPosition ()
int hashCode ()
String toString ()
```

## TextAction 类

```
java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
public abstract class TextAction
    extends AbstractAction
```

这个类表示一个动作，它用于对由几个文本组件所共享的按键进行绑定 (binding)。

### Constructor

```
TextAction (String name)
```

### Methods

static Action []	augmentList (Action [] list1, Action [] list2)
protected JTextComponent	getFocusedComponent ()
protected JTextComponent	getTextComponent (ActionEvent e)

## Utilities 类

```
java.lang.Object
|
+-- javax.swing.text.Utilities
public class Utilities
    extends Object
```

这个类表示一个实用工具方法 (utility method) 的集合，这些方法用来处理各种与文本相关功能。

### Constructor

```
Utilities ()
```

### Methods

static int	drawTabbedText (Segment s, int x, int y, Graphics g, TabExpander e, int startOffset)
static int	getBreakLocation (Segment s, FontMetrics metrics, int x0, int x, TabExpander e, int startOffset)
static int	getNextWord (JTextComponent c, int off)
static Element	getParagraphElement (JTextComponent c, int off)
static int	getPositionAbove (JTextComponent c, int off, int x)
static int	getPositionBelow (JTextComponent c, int off, int x)
static int	getPreviousWord (JTextComponent c, int off)
static int	getRowEnd (JTextComponent c, int off)
static int	getRowStart (JTextComponent c, int off)
static int	getTabbedTextOffset (Segment s, FontMetrics metrics, int x0, int x, TabExpander e, int startOffset)
static int	getTabbedTextOffset (Segment s, FontMetrics metrics, int x0, int x, TabExpander e, int startOffset, boolean round)
static int	getTabbedTextWidth (Segment s, FontMetrics metrics, int x, TabExpander e, int startOffset)
static int	getWordEnd (JTextComponent c, int off)
static int	getWordStart (JTextComponent c, int off)

## View 类

```
java.lang.Object
|
+-- javax.swing.text.View
public abstract class View
    extends Object
    implements SwingConstants
```

这个类表示文档的一个特定部分的视图。

### Fields

```
static int BadBreakWeight
```

```

static int ExcellentBreakWeight
static int ForcedBreakWeight
static int GoodBreakWeight
static int X_AXIS
static int Y_AXIS

```

**Constructor**

```
View (Element elem)
```

**Methods**

View	breakview (int axis, int offset, float pos, float len)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
View	createFragment (int p0, int p1)
float	getAlignment (int axis)
AttributeSet	getAttributes ()
int	getBreakWeight (int axis, float pos, float len)
Shape	getChildAllocation (int index, Shape a)
Container	getContainer ()
Document	getDocument ()
Element	getElement ()
int	getEndOffset ()
float	getMaximumSpan (int axis)
float	getMinimumSpan (int axis)
int	getNextVisualPositionFrom (int pos, Position.Bias b, Shape a, int direction, Position.Bias [] biasRet)
View	getParent ()
abstract float	getPreferredSpan (int axis)
int	getResizeWeight (int axis)
int	getStartOffset ()
View	getView (int n)
int	getViewCount ()
ViewFactory	getViewFactory ()
void	insertUpdate (DocumentEvent e, Shape a, ViewFactory f)
boolean	isVisible ()
Shape	modelToView (int p0, Position.Bias b0, int p1, Position.Bias b1, Shape a)
Shape	modelToView (int pos, Shape a)
abstract Shape	modelToView (int pos, Shape a, Position.Bias b)
abstract void	paint (Graphics g, Shape allocation)
void	preferenceChanged (View child, boolean width, boolean height)
void	removeUpdate (DocumentEvent e, Shape a, ViewFactory f)
void	setParent (View parent)
void	setSize (float width, float height)
int	viewToModel (float x, float y, Shape a)
abstract int	viewToModel (float x, float y, Shape a, Position.Bias [] biasReturn)

## WrappedPlainView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.WrappedPlainView
public class WrappedPlainView
    extends BoxView
    implements TabExpander

```

这个类表示能够折行的纯文本视图。该纯文本只能使用一种字体和颜色。

### Constructors

```

WrappedPlainView (Element elem)
WrappedPlainView (Element elem, boolean wordWrap)

```

### Methods

protected int	calculateBreakPosition (int p0, int p1)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
protected void	drawLine (int p0, int p1, Graphics g, int x, int y)
protected int	drawSelectedText (Graphics g, int x, int y, int p0, int p1)
protected int	drawUnselectedText (Graphics g, int x, int y, int p0, int p1)
protected Segment	getLineBuffer ()
float	getMaximumSpan (int axis)
float	getMinimumSpan (int axis)
float	getPreferredSize (int axis)
protected int	getTabSize ()
void	insertUpdate (DocumentEvent e, Shape a, ViewFactory f)
protected void	loadChildren (ViewFactory f)
float	nextTabStop (float x, int tabOffset)
void	paint (Graphics g, Shape a)
void	removeUpdate (DocumentEvent e, Shape a, ViewFactory f)
void	setSize (float width, float height)

## BadLocationException 类

```

java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- javax.swing.text.BadLocationException
public class BadLocationException
    extends Exception

```

这个类表示一个异常，如果试图引用一个不存在的单元，就会抛出该异常。

### Constructor

```
BadLocationException (String s, int offs)
```

### Method

```
int offsetRequested ()
```

## ChangedCharSetException 类

```
java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- java.io.IOException
|
+-- javax.swing.text.ChangedCharSetException
public class ChangedCharSetException
    extends IOException
```

这个类表示一个异常。当字符集改变时，就会抛出该异常。

### Constructor

```
ChangedCharSetException (String charSetSpec, boolean charSetKey)
```

### Methods

```
String getCharSetSpec ()
boolean keyEqualsCharSet ()
```

## javax.swing.text.html

### BlockView 类

```
java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.html.BlockView
public class BlockView
    extends BoxView
```

这个类表示一个视图，它表示了一个带有 CSS 规范的文本块。

### Constructor

```
BlockView (Element elem, int axis)
```

**Methods**

float	getAlignment ( int axis)
AttributeSet	getAttributes ()
int	getResizeWeight ( int axis)
protected StyleSheet	getStyleSheet ()
void	paint ( Graphics g, Shape allocation)
protected void	setPropertiesFromAttributes ()

**CSS 类**

```
java.lang.Object
|
+-- javax.swing.text.html.CSS
public class CSS
    extends Object
```

这个类表示 CSS 属性。HTML 文档的这个视图实现需要使用 CSS 属性来决定怎样绘制信息内容。

**Inner Class**

static class CSS.Attribute

**Constructor**

CSS ()

**Methods**

static CSS.Attribute []	getAllAttributeKeys ()
static CSS.Attribute	getAttribute (String name)

**CSS.Attribute 类**

```
java.lang.Object
|
+-- javax.swing.text.html.CSS.Attribute
public static final class CSS.Attribute
    extends Object
```

这个内部的（也是最终的）类代表在属性集上用作按键的定义，该属性集包含了 CSS 属性。

**fields**

static CSS.Attribute	BACKGROUND
static CSS.Attribute	BACKGROUND_ATTACHMENT
static CSS.Attribute	BACKGROUND_COLOR
static CSS.Attribute	BACKGROUND_IMAGE
static CSS.Attribute	BACKGROUND_POSITION
static CSS.Attribute	BACKGROUND_REPEAT
static CSS.Attribute	BORDER

---

static CSS.Attribute	BORDER_BOTTOM
static CSS.Attribute	BORDER_BOTTOM_WIDTH
static CSS.Attribute	BORDER_COLOR
static CSS.Attribute	BORDER_LEFT
static CSS.Attribute	BORDER_LEFT_WIDTH
static CSS.Attribute	BORDER_RIGHT
static CSS.Attribute	BORDER_RIGHT_WIDTH
static CSS.Attribute	BORDER_STYLE
static CSS.Attribute	BORDER_TOP
static CSS.Attribute	BORDER_TOP_WIDTH
static CSS.Attribute	BORDER_WIDTH
static CSS.Attribute	CLEAR
static CSS.Attribute	COLOR
static CSS.Attribute	DISPLAY
static CSS.Attribute	FLOAT
static CSS.Attribute	FONT
static CSS.Attribute	FONT_FAMILY
static CSS.Attribute	FONT_SIZE
static CSS.Attribute	FONT_STYLE
static CSS.Attribute	FONT_VARIANT
static CSS.Attribute	FONT_WEIGHT
static CSS.Attribute	HEIGHT
static CSS.Attribute	LETTER_SPACING
static CSS.Attribute	LINE_HEIGHT
static CSS.Attribute	LIST_STYLE
static CSS.Attribute	LIST_STYLE_IMAGE
static CSS.Attribute	LIST_STYLE_POSITION
static CSS.Attribute	LIST_STYLE_TYPE
static CSS.Attribute	MARGIN
static CSS.Attribute	MARGIN_BOTTOM
static CSS.Attribute	MARGIN_LEFT
static CSS.Attribute	MARGIN_RIGHT
static CSS.Attribute	MARGIN_TOP
static CSS.Attribute	PADDING
static CSS.Attribute	PADDING_BOTTOM
static CSS.Attribute	PADDING_LEFT
static CSS.Attribute	PADDING_RIGHT
static CSS.Attribute	PADDING_TOP
static CSS.Attribute	TEXT_ALIGN
static CSS.Attribute	TEXT_DECORATION
static CSS.Attribute	TEXT_INDENT
static CSS.Attribute	TEXT_TRANSFORM
static CSS.Attribute	VERTICAL_ALIGN

```

static CSS.Attribute    WHITE_SPACE
static CSS.Attribute    WIDTH
static CSS.Attribute    WORD_SPACING

```

### Methods

```

String      getDefaultValue ()
boolean     isInherited ()
String      toString ()

```

## FormView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.ComponentView
|
+-- javax.swing.text.html.FormView
public class FormView
    extends ComponentView
    implements ActionListener

```

这个类表示一个针对窗体元素、输入、文本区域以及选择窗体视图。

### Inner Class

```
protected class FormView.MouseEventListener
```

### Fields

```

static String    RESET
static String    SUBMIT

```

### Constructor

```
FormView (Element elem)
```

### Methods

```

void          actionPerformed (ActionEvent evt)
protected Component  createComponent ()
protected void    imageSubmit (String imageData)
protected void    submitData (String data)

```

## HTML 类

```

java.lang.Object
|
+-- javax.swing.text.html.HTML
public class HTML
    extends Object

```

这个类表示一个对象，它包含了用于 HTML 文档的常量。

**Inner Classes**

static class	HTML.Attribute
static class	HTML.Tag
static class	HTML.UnknownTag

**Field**

static String	NULL_ATTRIBUTE_VALUE
---------------	----------------------

**Constructor**

HTML ()
---------

**Methods**

static HTML.Attribute []	getAllAttributeKeys ()
static HTML.Tag []	getAllTags ()
static HTML.Attribute	getAttributeKey (String attName)
static int	getIntegerAttributeValue (AttributeSet attr,
	HTML.Attribute key, int def)
static HTML.Tag	getTag (String tagName)

**HTML.Attribute 类**

```

java.lang.Object
|
+-- javax.swing.text.html.HTML.Attribute
public static final class HTML.Attribute
    extends Object

```

这个内部类代表一个 HTML 属性。

**Fields**

static HTML.Attribute	ACTION
static HTML.Attribute	ALIGN
static HTML.Attribute	ALINK
static HTML.Attribute	ALT
static HTML.Attribute	ARCHIVE
static HTML.Attribute	BACKGROUND
static HTML.Attribute	BGCOLOR
static HTML.Attribute	BORDER
static HTML.Attribute	CELLPADDING
static HTML.Attribute	CELLSPACING
static HTML.Attribute	CHECKED
static HTML.Attribute	CLASS
static HTML.Attribute	CLASSID
static HTML.Attribute	CLEAR
static HTML.Attribute	CODE
static HTML.Attribute	CODERASE

static HTML.Attribute	CODETYPE
static HTML.Attribute	COLOR
static HTML.Attribute	COLS
static HTML.Attribute	COLSPAN
static HTML.Attribute	COMMENT
static HTML.Attribute	COMPACT
static HTML.Attribute	CONTENT
static HTML.Attribute	COORDS
static HTML.Attribute	DATA
static HTML.Attribute	DECLARE
static HTML.Attribute	DIR
static HTML.Attribute	DUMMY
static HTML.Attribute	ENCTYPE
static HTML.Attribute	ENDTAG
static HTML.Attribute	FACE
static HTML.Attribute	FRAMEBORDER
static HTML.Attribute	HALIGN
static HTML.Attribute	HEIGHT
static HTML.Attribute	HREF
static HTML.Attribute	HSPACE
static HTML.Attribute	HTTPEQUIV
static HTML.Attribute	ID
static HTML.Attribute	ISMAP
static HTML.Attribute	LANG
static HTML.Attribute	LANGUAGE
static HTML.Attribute	LINK
static HTML.Attribute	LOWSRC
static HTML.Attribute	MARGINHEIGHT
static HTML.Attribute	MARGINWIDTH
static HTML.Attribute	maxlength
static HTML.Attribute	METHOD
static HTML.Attribute	MULTIPLE
static HTML.Attribute	N
static HTML.Attribute	NAME
static HTML.Attribute	NOHREF
static HTML.Attribute	NORESIZE
static HTML.Attribute	NOSHADE
static HTML.Attribute	NOWRAP
static HTML.Attribute	PROMPT
static HTML.Attribute	REL
static HTML.Attribute	REV
static HTML.Attribute	ROWS
static HTML.Attribute	ROWSPAN

---

static HTML.Attribute	SCROLLING
static HTML.Attribute	SELECTED
static HTML.Attribute	SHAPE
static HTML.Attribute	SHAPES
static HTML.Attribute	SIZE
static HTML.Attribute	SRC
static HTML.Attribute	STANDBY
static HTML.Attribute	START
static HTML.Attribute	STYLE
static HTML.Attribute	TARGET
static HTML.Attribute	TEXT
static HTML.Attribute	TITLE
static HTML.Attribute	TYPE
static HTML.Attribute	USEMAP
static HTML.Attribute	VALIGN
static HTML.Attribute	VALUE
static HTML.Attribute	VALUETYPE
static HTML.Attribute	VERSION
static HTML.Attribute	VLINK
static HTML.Attribute	VSPACE
static HTML.Attribute	WIDTH

**Method**

```
String    toString ()
```

**HTML.Tag 类**

```
java.lang.Object
|
+-- javax.swing.text.html.HTML.Tag
public static class HTML.Tag
    extends Object
```

这个类表示 HTML 标志。

**Fields**

static HTML.Tag	A
static HTML.Tag	ADDRESS
static HTML.Tag	APPLET
static HTML.Tag	AREA
static HTML.Tag	B
static HTML.Tag	BASE
static HTML.Tag	BASEFONT
static HTML.Tag	BIG
static HTML.Tag	BLOCKQUOTE
static HTML.Tag	BODY

static HTML.Tag	BR
static HTML.Tag	CAPTION
static HTML.Tag	CENTER
static HTML.Tag	CITE
static HTML.Tag	CODE
static HTML.Tag	COMMENT
static HTML.Tag	CONTENT
static HTML.Tag	DD
static HTML.Tag	DEN
static HTML.Tag	DIR
static HTML.Tag	DIV
static HTML.Tag	DL
static HTML.Tag	DT
static HTML.Tag	EM
static HTML.Tag	FONT
static HTML.Tag	FORM
static HTML.Tag	FRAME
static HTML.Tag	FRAMESET
static HTML.Tag	H1
static HTML.Tag	H2
static HTML.Tag	H3
static HTML.Tag	H4
static HTML.Tag	H5
static HTML.Tag	H6
static HTML.Tag	HEAD
static HTML.Tag	HR
static HTML.Tag	HTML
static HTML.Tag	I
static HTML.Tag	IMG
static HTML.Tag	IMPLIED
static HTML.Tag	INPUT
static HTML.Tag	ISINDEX
static HTML.Tag	KBD
static HTML.Tag	LI
static HTML.Tag	LINK
static HTML.Tag	MAP
static HTML.Tag	MENU
static HTML.Tag	META
static HTML.Tag	NOFRAMES
static HTML.Tag	OBJECT
static HTML.Tag	OL
static HTML.Tag	OPTION
static HTML.Tag	P

```

static HTML.Tag    PARAM
static HTML.Tag    PRE
static HTML.Tag    S
static HTML.Tag    SAMP
static HTML.Tag    SCRIPT
static HTML.Tag    SELECT
static HTML.Tag    SMALL
static HTML.Tag    STRIKE
static HTML.Tag    STRONG
static HTML.Tag    STYLE
static HTML.Tag    SUB
static HTML.Tag    SUP
static HTML.Tag    TABLE
static HTML.Tag    TD
static HTML.Tag    TEXTAREA
static HTML.Tag    TH
static HTML.Tag    TITLE
static HTML.Tag    TR
static HTML.Tag    TT
static HTML.Tag    U
static HTML.Tag    UL
static HTML.Tag    VAR

```

**Constructors**

```

protected   HTML.Tag (String id)
protected   HTML.Tag (String id, boolean causesBreak, boolean isBlock)

```

**Methods**

```

boolean      breaksFlow ()
true.boolean isBlock ()
boolean      isPreformatted ()
String       toString ()

```

**HTML.UnknownTag 类**

```

java.lang.Object
|
+-- javax.swing.text.html.HTML.Tag
|
+-- javax.swing.text.html.HTML.UnknownTag
public static class HTML.UnknownTag
    extends HTML.Tag
    implements Serializable

```

这个内部类代表一个未知的标志。

**Constructor**

```
HTML.UnknownTag (String id)
```

## Methods

```
boolean equals (Object obj)
int hashCode ()
```

## HTMLDocument 类

```
java.lang.Object
|
+-- javax.swing.text.AbstractDocument
|
+-- javax.swing.text.DefaultStyledDocument
|
+-- javax.swing.text.html.HTMLDocument
public class HTMLDocument
    extends DefaultStyledDocument
```

这个类表示 HTML 的内容模型，它支持浏览和编辑。

## Inner Classes

```
class HTMLDocument.BlockElement
class HTMLDocument.HTMLReader
static class HTMLDocument.Iterator
class HTMLDocument.RunElement
```

## Field

```
static String AdditionalComments
```

## Constructors

```
HTMLDocument ()
HTMLDocument (AbstractDocument.Content c, StyleSheet styles)
HTMLDocument (StyleSheet styles)
```

## Methods

protected void	create (DefaultStyledDocument,
	ElementSpec [] data)
protected Element	createBranchElement
	(Element parent, AttributeSet a)
protected AbstractDocument.AbstractElement	createDefaultRoot ()
protected Element	createLeafElement (Element parent,
	AttributeSet a, int p0, int p1)
URL	getBase ()
HTMLDocument.Iterator	getIterator (HTML.Tag t)
boolean	getPreservesUnknownTags ()
HMLEditorKit.ParserCallback	getReader (int pos)
HMLEditorKit.ParserCallback	getReader (int pos, int popDepth,
	int pushDepth,
	HTML.Tag insertTag)
StyleSheet	getStyleSheet ()

```

int                               getTokenThreshold ()
protected void                  insert (int offset,
                                         DefaultStyledDocument,
                                         ElementSpec [] data)
protected void                  insertUpdate (AbstractDocument,
                                         DefaultDocumentEvent chng,
                                         AttributeSet attr)
void                            processHTMLFrameHyperlinkEvent
                                (HTMLFrameHyperlinkEvent e)
void                            setBase (URL u)
void                            setPreservesUnknownTags (boolean
                                         preservesTags)
void                            setTokenThreshold (int n)

```

## **HTMLDocument.Iterator 类**

```

java.lang.Object
|
+-- javax.swing.text.html.HTMLDocument.Iterator
public abstract static class HTMLDocument.Iterator
    extends Object

```

这个类表示一个迭代器（它不是安全线程），它只对某些特定类型的标志进行迭代。

### **Constructor**

```
HTMLDocument.Iterator ()
```

### **Methods**

abstract AttributeSet	getAttributes ()
abstract int	getEndOffset ()
abstract int	getStartOffset ()
abstract HTML.Tag	getTag ()
abstract boolean	isValid ()
abstract void	next ()

## **HTMLEditorKit 类**

```

java.lang.Object
|
+-- javax.swing.text.EditorKit
|
+-- javax.swing.text.DefaultEditorKit
|
+-- javax.swing.text.StyledEditorKit
|
+-- javax.swing.text.html.HTMLEditorKit
public class HTMLEditorKit
    extends StyledEditorKit

```

这个类表示一个 HTML 3.2 版的编辑工具箱，注意它不支持 applet 标志。

**Inner Classes**

static class	HMLEditorKit.HTMLFactory
static class	HMLEditorKit.HTMLTextAction
static class	HMLEditorKit.InsertHTMLTextAction
static class	HMLEditorKit.LinkController
static class	HMLEditorKit.Parser
static class	HMLEditorKit.ParserCallback

**Fields**

static String	BOLD_ACTION
static String	COLOR_ACTION
static String	DEFAULT_CSS
static String	FONT_CHANGE_BIGGER
static String	FONT_CHANGE_SMALLER
static String	IMG_ALIGN_BOTTOM
static String	IMG_ALIGN_MIDDLE
static String	IMG_ALIGN_TOP
static String	IMG_BORDER
static String	ITALIC_ACTION
static String	LOGICAL_STYLE_ACTION
static String	PARA_INDENT_LEFT
static String	PARA_INDENT_RIGHT

**Constructor**

HMLEditorKit ()

**Methods**

Object	clone ()
Document	createDefaultDocument ()
protected void	createInputAttributes (Element element, MutableAttributeSet set)
void	deinstall (JEditorPane c)
Action []	getActions ()
String	getContentType ()
protected HMLEditorKit.Parser	getParser ()
StyleSheet	getStyleSheet ()
ViewFactory	getViewFactory ()
void	insertHTML (HTMLDocument doc, int offset, String html, int popDepth, int pushDepth, HTML.Tag insertTag)
void	install (JEditorPane c)
void	read (Reader in, Document doc, int pos)

```

void           setStyleSheet (StyleSheet s)
void           write (Writer out, Document doc,
                     int pos, int len)

```

## HTMLEditorKit.HTMLFactory 类

```

java.lang.Object
|
+-- javax.swing.text.html.HTMLEditorKit.HTMLFactory
public static class HTMLEditorKit.HTMLFactory
    extends Object
    implements ViewFactory

```

这个内部类代表一个建立 HTML 内容的视图的场所。

### Constructor

```
HTMLEditorKit.HTMLFactory ()
```

### Method

```
View create (Element elem)
```

## HTMLEditorKit.HTMLTextAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
    |
    +-- javax.swing.text.TextAction
        |
        +-- javax.swing.text.StyledEditorKit.StyledTextAction
            |
            +-- javax.swing.text.html.HTMLEditorKit.HTMLTextAction
public abstract static class HTMLEditorKit.HTMLTextAction
    extends StyledEditorKit.StyledTextAction

```

这个内部类代表一个抽象动作，它支持特定的方法，将 HTML 包括在已存在的文档中。

### Constructor

```
HTMLEditorKit.HTMLTextAction (String name)
```

### Methods

protected int	elementCountToTag (HTMLDocument doc, int offset, HTML.Tag tag)
protected Element	findElementMatchingTag (HTMLDocument doc, int offset, HTML.Tag tag)
protected Element []	getElementsAt (HTMLDocument doc, int offset)
protected HTMLDocument	getHTMLDocument (JEditorPane e)
protected HTMLEditorKit	getHTMLEditorKit (JEditorPane e)

## HTMLEditorKit.InsertHTMLTextAction 类

```

java.lang.Object
|
+-- javax.swing.AbstractAction
|
+-- javax.swing.text.TextAction
|
+-- javax.swing.text.StyledEditorKit.StyledTextAction
|
+-- javax.swing.text.html.HTMLEditorKit.HTMLTextAction
|
+-- javax.swing.text.html.HTMLEditorKit.InsertHTMLTextAction
public static class HTMLEditorKit.InsertHTMLTextAction
    extends HTMLEditorKit.HTMLTextAction

```

这个类表示一个动作对象，它被用来将 HTML 插入到已存在的 HTML 文档中。

### Fields

protected HTML.Tag	addTag
protected HTML.Tag	alternateAddTag
protected HTML.Tag	alternateParentTag
protected String	html
protected HTML.Tag	parentTag

### Constructors

```

HTMLEditorKit.InsertHTMLTextAction (String name, String html, HTML.Tag parentTag, HTML.Tag add-
tag)
HTMLEditorKit.InsertHTMLTextAction (String name, String html, HTML.Tag parentTag, HTML.Tag add-
tag, HTML.Tag alternateParentTag, HTML.Tag alternateAddTag)

```

### Methods

void	actionPerformed (ActionEvent ae)
protected void	insertAtBoundary (JEditorPane editor, HTMLDocument doc, int offset, Element insertElement, String html, HTML.Tag parentTag, HTML.Tag addTag)
protected void	insertHTML (JEditorPane editor, HTMLDocument doc, int offset, String html, int popDepth, int pushDepth, HTML.Tag addTag)

## HTMLEditorKit.LinkController 类

```

java.lang.Object
|
+-- java.awt.event.MouseAdapter
|
+-- javax.swing.text.html.HTMLEditorKit.LinkController
public static class HTMLEditorKit.LinkController
    extends MouseAdapter
    implements Serializable

```

这个类表示一个对象，来观察相关的组件并在必要时发出超链接事件。

**Constructor**

```
HTMLEditorKit.LinkController ()
```

**Methods**

```
protected void activateLink (int pos, JEditorPane html)  
void mouseClicked (MouseEvent e)
```

**HTMLEditorKit.Parser 类**

```
java.lang.Object  
|  
+-- javax.swing.text.html.HTMLEditorKit.Parser  
public abstract static class HTMLEditorKit.Parser  
    extends Object
```

这个内部类代表一个解析器 (parser)。

**Constructor**

```
HTMLEditorKit.Parser ()
```

**Method**

```
abstract void parse (Reader r, HTMLEditorKit.ParserCallback cb,  
                     boolean ignoreCharSet)
```

**HTMLEditorKit.ParserCallback 类**

```
java.lang.Object  
|  
+-- javax.swing.text.html.HTMLEditorKit.ParserCallback  
public static class HTMLEditorKit.ParserCallback  
    extends Object
```

这个内部类代表解析器调用返回 (parser callbacks) 动作。

**Constructor**

```
HTMLEditorKit.ParserCallback ()
```

**Methods**

```
void flush ()  
void handleComment (char [] data, int pos)  
void handleEndTag (HTML.Tag t, int pos)  
void handleError (String errorMsg, int pos)  
void handleSimpleTag (HTML.Tag t, MutableAttributeSet a, int pos)  
void handleStartTag (HTML.Tag t, MutableAttributeSet a, int pos)  
void handleText (char [] data, int pos)
```

## HTMLFrameHyperlinkEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- javax.swing.event.HyperlinkEvent
|
+-- javax.swing.text.html.HTMLFrameHyperlinkEvent
public class HTMLFrameHyperlinkEvent
    extends HyperlinkEvent

```

这个类表示一个事件，它通知已注册的监听类，已在框架中激活了链接。

### Constructors

```

HTMLFrameHyperlinkEvent (Object source, HyperlinkEvent.EventType type,
URL targetURL, Element sourceElement, String targetFrame)
HTMLFrameHyperlinkEvent (Object source, HyperlinkEvent.EventType type,
URL targetURL, String targetFrame)
HTMLFrameHyperlinkEvent (Object source, HyperlinkEvent.EventType type,
URL targetURL, String desc, Element sourceElement, String targetFrame)
HTMLFrameHyperlinkEvent (Object source, HyperlinkEvent.EventType type,
URL targetURL, String desc, String targetFrame)

```

### Methods

```

Element getSourceElement ()
String getTarget ()

```

## HTMLWriter 类

```

java.lang.Object
|
+-- javax.swing.text.AbstractWriter
|
+-- javax.swing.text.html.HTMLWriter
public class HTMLWriter
    extends AbstractWriter

```

这个类表示 HTML 文档的一个书写器。

### Constructors

```

HTMLWriter (Writer w, HTMLDocument doc)
HTMLWriter (Writer w, HTMLDocument doc, int pos, int len)

```

### Methods

protected void	closeOutUnwantedEmbeddedTags (AttributeSet attr)
protected void	comment (Element elem)
protected void	emptyTag (Element elem)
protected void	endTag (Element elem)

```

protected boolean      isBlockTag (AttributeSet attr)
protected boolean      matchNameAttribute (AttributeSet attr, HTML.Tag tag)
protected void          selectContent (AttributeSet attr)
protected void          startTag (Element elem)
protected boolean      synthesizedElement (Element elem)
protected void          text (Element elem)
protected void          textAreaContent (AttributeSet attr)
void                  write ()
protected void          write (String content)
protected void          writeAttributes (AttributeSet attr)
protected void          writeEmbeddedTags (AttributeSet attr)
protected void          writeOption (Option option)

```

## InlineView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.LabelView
|
+-- javax.swing.text.html.InlineView
public class InlineView
    extends LabelView

```

这个类表示一个对象，它显示基于 CSS 属性的内联元素风格。

### Constructor

```
InlineView (Element elem)
```

### Methods

AttributeSet	getAttributes ()
protected StyleSheet	getStyleSheet ()
boolean	isVisible ()
protected void	setPropertiesFromAttributes ()

## ListView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.html.BlockView
|
+-- javax.swing.text.html.ListView
public class ListView
    extends BlockView

```

这个类表示一个显示 HTML 列表视图对象。

### Constructor

```
ListView (Element elem)
```

### Methods

float	getAlignment (int axis)
void	paint (Graphics g, Shape allocation)
protected void	paintChild (Graphics g, Rectangle alloc, int index)

## MinimalHTMLWriter 类

```
java.lang.Object
|
+-- javax.swing.text.AbstractWriter
    |
    +-- javax.swing.text.html.MinimalHTMLWriter
public class MinimalHTMLWriter
    extends AbstractWriter
```

这个类表示一个最小的 HTML 书写器。

### Constructors

```
MinimalHTMLWriter (Writer w, StyledDocument doc)
MinimalHTMLWriter (Writer w, StyledDocument doc, int pos, int len)
```

## ObjectView 类

```
java.lang.Object
|
+-- javax.swing.text.View
    |
    +-- javax.swing.text.ComponentView
        |
        +-- javax.swing.text.html.ObjectView
public class ObjectView
    extends ComponentView
```

这个类表示一个组件视图装饰器。

### Constructor

```
ObjectView (Element elem)
```

### Method

```
protected Component createComponent ()
```

## Option 类

```
java.lang.Object
|
+-- javax.swing.text.html.Option
public class Option
    extends Object
```

这个类表示选项单元。

### Constructor

Option (AttributeSet attr)

### Methods

AttributeSet	getAttributes ()
String	getLabel ()
String	getValue ()
boolean	isSelected ()
void	setLabel (String label)
protected void	setSelection (boolean state)
String	toString ()

## ParagraphView 类

```

java.lang.Object
|
+-- javax.swing.text.View
|
+-- javax.swing.text.CompositeView
|
+-- javax.swing.text.BoxView
|
+-- javax.swing.text.ParagraphView
|
+-- javax.swing.text.html.ParagraphView

public class ParagraphView
    extends ParagraphView

```

这个类通过使用 CSS 属性代表 Paragraph 的一个视图。

### Constructor

ParagraphView (Element elem)

### Methods

protected SizeRequirements	calculateMinorAxisRequirements (int axis, SizeRequirements r)
void	changedUpdate (DocumentEvent e, Shape a, ViewFactory f)
AttributeSet	getAttributes ()
float	getMaximumSpan (int axis)
float	getMinimumSpan (int axis)
float	getPreferredSpan (int axis)
protected StyleSheet	getStyleSheet ()
boolean	isVisible ()
void	setParent (View parent)
protected void	setPropertiesFromAttributes ()

## StyleSheet 类

```

java.lang.Object
|
+-- javax.swing.text.StyleContext
    |
    +-- javax.swing.text.html.StyleSheet
public class StyleSheet
    extends StyleContext

```

这个类表示一个样式单，它将 HTML 文档变成一种可视形式。

### Inner Classes

```

static class StyleSheet.BoxPainter
static class StyleSheet.ListPainter

```

### Constructor

```
StyleSheet ()
```

### Methods

void	addRule (String rule)
Color	getBackground (AttributeSet a)
StyleSheet.BoxPainter	getBoxPainter (AttributeSet a)
AttributeSet	getDeclaration (String decl)
Font	getFont (AttributeSet a)
Color	getForeground (AttributeSet a)
static int	getIndexOfSize (float pt)
StyleSheet.ListPainter	getListPainter (AttributeSet a)
float	getPointSize (int index)
float	getPointSize (String size)
Style	getRule (HTML.Tag t, Element e)
Style	getRule (String selector)
AttributeSet	getViewAttributes (View v)
void	loadRules (Reader in, URL ref)
void	setBaseFontSize (int sz)
void	setBaseFontSize (String size)
Color	stringToColor (String str)
AttributeSet	translateHTMLToCSS (AttributeSet htmlAttrSet)

### Class StyleSheet.BoxPainter

```

java.lang.Object
|
+-- javax.swing.text.html.StyleSheet.BoxPainter
public static class StyleSheet.BoxPainter
    extends Object
    implements Serializable

```

这个内部类代表一个对象，它处理对 HTML 文档的 CSS 格式化。

#### Methods

```
float    getInset (int side, View v)
void    paint (Graphics g, float x, float y, float w, float h, View v)
```

### Class StyleSheet.ListPainter

```
java.lang.Object
|
+-- javax.swing.text.html.StyleSheet.ListPainter
public static class StyleSheet.ListPainter
    extends Object
    implements Serializable
```

这个类也代表一个对象，该对象负责对 HTML 文档进行 CSS 格式化处理。

#### Methods

```
void    paint (Graphics g, float x, float y, float w, float h, View v, int item)
```

## javax.swing.tree

### Interface MutableTreeNode

```
public abstract interface MutableTreeNode
    extends TreeNode
```

这个接口为易变的或可变的树节点定义了设计级抽象。一个易变的树节点易于节点的增加和删除操作。

#### Methods

```
void    insert (MutableTreeNode child, int index)
void    remove (int index)
void    remove (MutableTreeNode node)
void    removeFromParent ()
void    setParent (MutableTreeNode newParent)
void    setUserObject (Object object)
```

### Interface RowMapper

```
public abstract interface RowMapper
```

这个接口描述了一个映射，它将树的路径数据翻译成树的行。

#### Method

```
int []    getRowsForPaths (TreePath [] path)
```

### Interface TreeCellEditor

```
public abstract interface TreeCellEditor
    extends CellEditor
```

这个接口描述了树的单元编辑器。你必须用返回该组件所支持的方法来定义该编辑器组

件的属性。

### Method

```
Component getTreeCellEditorComponent (JTree tree, Object value,
                                     boolean isSelected, boolean expanded, boolean leaf, int row)
```

### Interface TreeCellRenderer

```
public abstract interface TreeCellRenderer
```

这个接口描述了一个树节点的单元绘制器。必须用返回该组件所支持的方法来定义该绘制器组件的属性。

### Method

```
Component getTreeCellRendererComponent (JTree tree, Object value,
                                         boolean selected, boolean expanded, boolean leaf, int row,
                                         boolean hasFocus)
```

### Interface TreeModel

```
public abstract interface TreeModel
```

这个接口描述了一个树对象的数据模型。该接口在创建定制的树模式时是很有用的。

### Methods

void	addTreeModelListener (TreeModelListener l)
Object	getChild (Object parent, int index)
int	getChildCount (Object parent)
int	getIndexofChild (Object parent, Object child)
Object	getRoot ()
boolean	isLeaf (Object node)
void	removeTreeModelListener (TreeModelListener l)
void	valueForPathChanged (TreePath path, Object newValue)

### Interface TreeNode

```
public abstract interface TreeNode
```

这个接口描述了树节点。

### Methods

Enumeration	children ()
boolean	getAllowsChildren ()
TreeNode	getChildAt (int childIndex)
int	getChildCount ()
int	getIndex (TreeNode node)
TreeNode	getParent ()
boolean	isLeaf ()

### Interface TreeSelectionModel

```
public abstract interface TreeSelectionModel
```

这个接口描述了存放树中选择的数据模型。

### Fields

```
static int    CONTIGUOUS_TREE_SELECTION
static int    DISCONTIGUOUS_TREE_SELECTION
static int    SINGLE_TREE_SELECTION
```

### Methods

```
void        addPropertyChangeListener (PropertyChangeListener listener)
void        addSelectionPath (TreePath path)
void        addSelectionPaths (TreePath [] paths)
void        addTreeSelectionListener (TreeSelectionListener x)
void        clearSelection ()
TreePath    getLeadSelectionPath ()
int         getLeadSelectionRow ()
int         getMaxSelectionRow ()
int         getMinSelectionRow ()
RowMapper   getRowMapper ()
int         getSelectionCount ()
int         getSelectionMode ()
TreePath    getSelectionPath ()
TreePath [] getSelectionPaths ()
int []      getSelectionRows ()
boolean     isPathSelected (TreePath path)
boolean     isRowSelected (int row)
boolean     isSelectionEmpty ()
void        removePropertyChangeListener (PropertyChangeListener listener)
void        removeSelectionPath (TreePath path)
void        removeSelectionPaths (TreePath [] paths)
void        removeTreeSelectionListener (TreeSelectionListener x)
void        resetRowSelection ()
void        setRowMapper (RowMapper newMapper)
void        setSelectionMode (int mode)
void        setSelectionPath (TreePath path)
void        setSelectionPaths (TreePath [] paths)
```

## AbstractLayoutCache 类

```
java.lang.Object
|
+-- javax.swing.tree.AbstractLayoutCache
public abstract class AbstractLayoutCache
    extends Object
    implements RowMapper
```

这个类表示树的布局的一个抽象高速缓存，树的布局包括节点个数、层数、树模型以及

和树选择模型。

### Inner Class

```
static class AbstractLayoutCache.NodeDimensions
```

### Fields

protected AbstractLayoutCache.NodeDimensions	nodeDimensions
protected boolean	rootVisible
protected int	rowHeight
protected TreeModel	treeModel
protected TreeSelectionModel	treeSelectionModel

### Constructor

```
AbstractLayoutCache ()
```

### Methods

abstract Rectangle	getBounds (TreePath path, Rectangle placeIn)
abstract boolean	getExpandedState (TreePath path)
TreeModel	getModel ()
AbstractLayoutCache.NodeDimensions	getNodeDimensions ()
protected	Rectangle getNodeDimensions (Object value, int row, int depth, boolean expanded, Rectangle placeIn)
abstract TreePath	getPathClosestTo (int x, int y)
abstract TreePath	getPathForRow (int row)
int	getPreferredHeight ()
int	getPreferredWidth (Rectangle bounds)
abstract int	getRowCount ()
abstract int	getRowForPath (TreePath path)
int	getRowHeight ()
int []	getRowsForPaths (TreePath [] paths)
TreeSelectionModel	getSelectionModel ()
abstract int	getVisibleChildCount (TreePath path)
abstract Enumeration	getVisiblePathsFrom (TreePath path)
abstract void	invalidatePathBounds (TreePath path)
abstract void	invalidateSizes ()
abstract boolean	isExpanded (TreePath path)
protected boolean	isFixedRowHeight ()
boolean	isRootVisible ()
abstract void	setExpandedState (TreePath path, boolean isExpanded)
void	setModel (TreeModel newModel)
void	setNodeDimensions (AbstractLayoutCache.NodeDimensions nd)

---

void	setRootVisible (boolean rootVisible)
void	setRowHeight (int rowHeight)
void	setSelectionModel (TreeSelectionModel newSM)
abstract void	treeNodesChanged (TreeModelEvent e)
abstract void	treeNodesInserted (TreeModelEvent e)
abstract void	treeNodesRemoved (TreeModelEvent e)
abstract void	treeStructureChanged (TreeModelEvent e)

## AbstractLayoutCache.NodeDimensions 类

```
java.lang.Object
|
+-- javax.swing.tree.AbstractLayoutCache.NodeDimensions
public abstract static class AbstractLayoutCache.NodeDimensions
    extends Object
```

这是一个内部类，它由树的布局的抽象高速缓存使用。

### Constructor

```
AbstractLayoutCache.NodeDimensions ()
```

### Method

```
abstract Rectangle getNodeDimensions (Object value, int row, int depth,
                                    boolean expanded, Rectangle bounds)
```

## DefaultMutableTreeNode 类

```
java.lang.Object
|
+-- javax.swing.tree.DefaultMutableTreeNode
public class DefaultMutableTreeNode
    extends Object
    implements Cloneable, MutableTreeNode, Serializable
```

这个类表示易变的树节点的一个缺省实现。

### Fields

protected boolean	allowsChildren
protected Vector	children
static Enumeration	EMPTY_ENUMERATION
protected MutableTreeNode	parent
protected Object	userObject

### Constructors

```
DefaultMutableTreeNode ()
DefaultMutableTreeNode (Object userObject)
DefaultMutableTreeNode (Object userObject, boolean allowsChildren)
```

**Methods**

void	add (MutableTreeNode newChild)
Enumeration	breadthFirstEnumeration ()
Enumeration	children ()
Object	clone ()
Enumeration	depthFirstEnumeration ()
boolean	getAllowsChildren ()
TreeNode	getChildAfter (TreeNode aChild)
TreeNode	getChildAt (int index)
TreeNode	getChildBefore (TreeNode aChild)
int	getChildCount ()
int	getDepth ()
TreeNode	getFirstChild ()
DefaultMutableTreeNode	getFirstLeaf ()
int	getIndex (TreeNode aChild)
TreeNode	getLastChild ()
DefaultMutableTreeNode	getLastLeaf ()
int	getLeafCount ()
int	getLevel ()
DefaultMutableTreeNode	getNextLeaf ()
DefaultMutableTreeNode	getNextNode ()
DefaultMutableTreeNode	getNextSibling ()
TreeNode	getParent ()
TreeNode []	getPath ()
protected TreeNode []	getPathToRoot (TreeNode aNode, int depth)
DefaultMutableTreeNode	getPreviousLeaf ()
DefaultMutableTreeNode	getPreviousNode ()
DefaultMutableTreeNode	getPreviousSibling ()
TreeNode	getRoot ()
TreeNode	getSharedAncestor (DefaultMutableTreeNode aNode)
int	getSiblingCount ()
Object	getUserObject ()
Object []	getUserObjectPath ()
void	insert (MutableTreeNode newChild, int childIndex)
boolean	isLeaf ()
boolean	isNodeAncestor (TreeNode anotherNode)
boolean	isNodeChild (TreeNode aNode)
boolean	isNodeDescendant (DefaultMutableTreeNode anotherNode)
boolean	isNodeRelated (DefaultMutableTreeNode aNode)
boolean	isNodeSibling (TreeNode anotherNode)
boolean	isRoot ()
Enumeration	pathFromAncestorEnumeration (TreeNode ancestor)

Enumeration	postorderEnumeration ()
Enumeration	preorderEnumeration ()
void	remove (int childIndex)
void	remove (MutableTreeNode aChild)
void	removeAllChildren ()
void	removeFromParent ()
void	setAllowsChildren (boolean allows)
void	setParent (MutableTreeNode newParent)
void	setUserObject (Object userObject)
String	toString ()

## DefaultTreeCellEditor 类

```
java.lang.Object
|
+-- javax.swing.tree.DefaultTreeCellEditor
public class DefaultTreeCellEditor
    extends Object
    implements ActionListener, TreeCellEditor, TreeSelectionListener
```

这个类表示用于树单元的缺省编辑器组件。

### Inner Classes

```
class DefaultTreeCellEditor.DefaultTextField
class DefaultTreeCellEditor.EditorContainer
```

### Fields

protected Color	borderSelectionColor
protected boolean	canEdit
protected Component	editingComponent
protected Container	editingContainer
protected Icon	editingIcon
protected Font	font
protected TreePath	lastPath
protected int	lastRow
protected int	offset
protected TreeCellEditor	realEditor
protected DefaultTreeCellRenderer	renderer
protected Timer	timer
protected JTree	tree

### Constructors

```
DefaultTreeCellEditor (JTree tree, DefaultTreeCellRenderer renderer)
DefaultTreeCellEditor (JTree tree, DefaultTreeCellRenderer renderer,
TreeCellEditor editor)
```

### Methods

void	actionPerformed (ActionEvent e)
------	---------------------------------

```

void addCellEditorListener (CellEditorListener l)
void cancelCellEditing ()
protected boolean carEditImmediately (EventObject event)
protected Container createContainer ()
protected TreeCellEditor createTreeCellEditor ()
protected void determineOffset (JTree tree, Object value,
boolean isSelected, boolean expanded, boolean leaf,
int row)
Color getBorderSelectionColor ()
Object getCellEditorValue ()
Font getFont ()
Component getTreeCellEditorComponent (JTree tree,
Object value, boolean isSelected, boolean expanded,
boolean leaf, int row)
protected boolean inHitRegion (int x, int y)
boolean isCellEditable (EventObject event)
protected void prepareForEditing ()
void removeCellEditorListener (CellEditorListener l)
void setBorderSelectionColor (Color newColor)
void setFont (Font font)
protected void setTree (JTree newTree)
boolean shouldSelectCell (EventObject event)
protected boolean shouldStartEditingTimer (EventObject event)
protected void startEditingTimer ()
boolean stopCellEditing ()

```

## Class DefaultTreeCellRenderer

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container
|
+-- javax.swing.JComponent
|
+-- javax.swing.JLabel
|
+-- javax.swing.tree.DefaultTreeCellRenderer
public class DefaultTreeCellRenderer
    extends JLabel
    implements TreeCellRenderer

```

这个类表示 Swing 树的缺省单元着色器。

### Fields

protected Color	backgroundNonSelectionColor
protected Color	backgroundSelectionColor
protected Color	borderSelectionColor

protected Icon	closedIcon
protected Icon	leafIcon
protected Icon	openIcon
protected boolean	selected
protected Color	textNonSelectionColor
protected Color	textSelectionColor

**Constructor**

```
DefaultTreeCellRenderer ()
```

**Methods**

Color	getBackgroundNonSelectionColor ()
Color	getBackgroundSelectionColor ()
Color	getBorderSelectionColor ()
Icon	getClosedIcon ()
Icon	getDefaultClosedIcon ()
Icon	getDefaultLeafIcon ()
Icon	getDefaultOpenIcon ()
Icon	getLeafIcon ()
Icon	getOpenIcon ()
Dimension	getPreferredSize ()
Color	getTextNonSelectionColor ()
Color	getTextSelectionColor ()
Component	getTreeCellRendererComponent (JTree tree, Object value, boolean sel, boolean expanded, boolean leaf, int row, boolean hasFocus)
void	paint (Graphics g)
void	setBackground (Color color)
void	setBackgroundNonSelectionColor (Color newColor)
void	setBackgroundSelectionColor (Color newColor)
void	setBorderSelectionColor (Color newColor)
void	setClosedIcon (Icon newIcon)
void	setFont (Font font)
void	setLeafIcon (Icon newIcon)
void	setOpenIcon (Icon newIcon)
void	setTextNonSelectionColor (Color newColor)
void	setTextSelectionColor (Color newColor)

**DefaultTreeModel 类**

```
java.lang.Object
|
+-- javax.swing.tree.DefaultTreeModel
public class DefaultTreeModel
    extends Object
    implements Serializable, TreeModel
```

这个类表示树的缺省数据模型。

**Fields**

protected boolean	asksAllowsChildren
protected EventListenerList	listenerList
protected TreeNode	root

**Constructors**

```
DefaultTreeModel (TreeNode root)
DefaultTreeModel (TreeNode root, boolean asksAllowsChildren)
```

**Methods**

void	addTreeModelListener (TreeModelListener l)
boolean	asksAllowsChildren ()
protected void	fireTreeNodesChanged (Object source, Object [] path, int [] childIndices, Object [] children)
protected void	fireTreeNodesInserted (Object source, Object [] path, int [] childIndices, Object [] children)
protected void	fireTreeNodesRemoved (Object source, Object [] path, int [] childIndices, Object [] children)
protected void	fireTreeStructureChanged (Object source, Object [] path, int [] childIndices, Object [] children)
Object	getChild (Object parent, int index)
int	getChildCount (Object parent)
int	getIndexOfChild (Object parent, Object child)
TreeNode []	getPathToRoot (TreeNode aNode)
protected TreeNode []	getPathToRoot (TreeNode aNode, int depth)
Object	getRoot ()
void	insertNodeInto (MutableTreeNode newChild, MutableTreeNode parent, int index)
boolean	isLeaf (Object node)
void	nodeChanged (TreeNode node)
void	nodesChanged (TreeNode node, int [] childIndices)
void	nodeStructureChanged (TreeNode node)
void	nodesWereInserted (TreeNode node, int [] childIndices)
void	nodesWereRemoved (TreeNode node, int [] childIndices, Object [] removedChildren)
void	reload ()
void	reload (TreeNode node)
void	removeNodeFromParent (MutableTreeNode node)
void	removeTreeModelListener (TreeModelListener l)
void	setAsksAllowsChildren (boolean newValue)
void	setRoot (TreeNode root)
void	valueForPathChanged (TreePath path, Object newValue)

## DefaultTreeSelectionModel 类

```

java.lang.Object
+
+-- javax.swing.tree.DefaultTreeSelectionModel
public class DefaultTreeSelectionModel
    extends Object
    implements Cloneable, Serializable, TreeSelectionModel

```

这个类表示缺省树选择模型。

### Fields

protected SwingPropertyChangeSupport	changeSupport
protected int	leadIndex
protected TreePath	leadPath
protected int	leadRow
protected EventListenerList	listenerList
protected DefaultListSelectionModel	listSelectionModel
protected RowMapper	rowMapper
protected TreePath []	selection
static String	SELECTION_MODE_PROPERTY
protected int	selectionMode

### Constructor

```
DefaultTreeSelectionModel ()
```

### Methods

void	addPropertyChangeListener (PropertyChangeListener listener)
void	addSelectionPath (TreePath path)
void	addSelectionPaths (TreePath [] paths)
void	addTreeSelectionListener (TreeSelectionListener x)
protected boolean	arePathsContiguous (TreePath [] paths)
protected boolean	canPathsBeAdded (TreePath [] paths)
protected boolean	canPathsBeRemoved (TreePath [] paths)
void	clearSelection ()
Object	clone ()
protected void	fireValueChanged (TreeSelectionEvent e)
TreePath	getLeadSelectionPath ()
int	getLeadSelectionRow ()
int	getMaxSelectionRow ()
int	getMinSelectionRow ()
RowMapper	getRowMapper ()
int	getSelectionCount ()
int	getSelectionMode ()
TreePath	getSelectionPath ()
TreePath []	getSelectionPaths ()

```

int []           getSelectionRows ()
protected void   insureRowContinuity ()
protected void   insureUniqueness ()
boolean         isPathSelected (TreePath path)
boolean         isRowSelected (int row)
boolean         isSelectionEmpty ()
protected void   notifyPathChange (Vector changedPaths,
                                  TreePath oldLeadSelection)
void            removePropertyChangeListener (PropertyChangeListener
                                              listener)
void            removeSelectionPath (TreePath path)
void            removeSelectionPaths (TreePath [] paths)
void            removeTreeSelectionListener (TreeSelectionListener x)
void            resetRowSelection ()
void            setRowMapper (RowMapper newMapper)
void            setSelectionMode (int mode)
void            setSelectionPath (TreePath path)
void            setSelectionPaths (TreePath [] pPaths)
String          toString ()
protected void   updateLeadIndex ()

```

## FixedHeightLayoutCache 类

```

java.lang.Object
|
+
+-- javax.swing.tree.AbstractLayoutCache
|
+-- javax.swing.tree.FixedHeightLayoutCache
public class FixedHeightLayoutCache
    extends AbstractLayoutCache

```

这个类表示一个具有固定大小的布局高速缓存。

### Constructor

```
FixedHeightLayoutCache ()
```

### Methods

Rectangle	getBounds (TreePath path, Rectangle placeIn)
boolean	getExpandedState (TreePath path)
TreePath	getPathClosestTo (int x, int y)
TreePath	getPathForRow (int row)
int	getRowCount ()
int	getRowForPath (TreePath path)
int	getVisibleChildCount (TreePath path)
Enumeration	getVisiblePathsFrom (TreePath path)
void	invalidatePathBounds (TreePath path)
void	invalidateSizes ()
boolean	isExpanded (TreePath path)

```

void           setExpandedState (TreePath path, boolean isExpanded)
void           setModel (TreeModel newModel)
void           setRootVisible (boolean rootVisible)
void           setRowHeight (int rowHeight)
void           treeNodesChanged (TreeModelEvent e)
void           treeNodesInserted (TreeModelEvent e)
void           treeNodesRemoved (TreeModelEvent e)
void           treeStructureChanged (TreeModelEvent e)

```

## TreePath 类

```

java.lang.Object
|
+-- javax.swing.tree.TreePath
public class TreePath
    extends Object
    implements Serializable

```

这个类表示指向一个树节点的路径。

### Constructors

protected	TreePath ()
	TreePath (Object singlePath)
	TreePath (Object [] path)
protected	TreePath (Object [] path, int length)
protected	TreePath (TreePath parent, Object lastElement)

### Methods

boolean	equals (Object o)
Object	getLastPathComponent ()
TreePath	getParentPath ()
Object []	getPath ()
Object	getPathComponent (int element)
int	getPathCount ()
int	hashCode ()
boolean	isDescendant (TreePath aTreePath)
TreePath	pathByAddingChild (Object child)
String	toString ()

## VariableHeightLayoutCache 类

```

java.lang.Object
|
+-- javax.swing.tree.AbstractLayoutCache
|
+-- javax.swing.tree.VariableHeightLayoutCache
public class VariableHeightLayoutCache
    extends AbstractLayoutCache

```

这个类表示一个具有可变大小的布局高速缓存。

### Constructor

```
VariableLayoutCache ()
```

### Methods

Rectangle	getBounds (TreePath path, Rectangle placeIn)
boolean	getExpandedState (TreePath path)
TreePath	getPathClosestTo (int x, int y)
TreePath	getPathForRow (int row)
int	getPreferredWidth (Rectangle bounds)
int	getRowCount ()
int	getRowForPath (TreePath path)
int	getVisibleChildCount (TreePath path)
Enumeration	getVisiblePathsFrom (TreePath path)
void	invalidatePathBounds (TreePath path)
void	invalidateSizes ()
boolean	isExpanded (TreePath path)
void	setExpandedState (TreePath path, boolean isExpanded)
void	setModel (TreeModel newModel)
void	setNodeDimensions (AbstractLayoutCache.NodeDimensions nd)
void	setRootVisible (boolean rootVisible)
void	setRowHeight (int rowHeight)
void	treeNodesChanged (TreeModelEvent e)
void	treeNodesInserted (TreeModelEvent e)
void	treeNodesRemoved (TreeModelEvent e)
void	treeStructureChanged (TreeModelEvent e)

### ExpandVetoException 类

```
java.lang.Object
  |
  +-- java.lang.Throwable
      |
      +-- java.lang.Exception
          |
          +-- javax.swing.tree.ExpandVetoException
```

```
public class ExpandVetoException
    extends Exception
```

这个类表示一个例外，当扩充或压缩树路径时出现了反常的条件，就会抛出它。

### Field

```
protected TreeExpansionEvent event
```

### Constructors

```
ExpandVetoException (TreeExpansionEvent event)
```

---

```
ExpandVetoException (TreeExpansionEvent event, String message)
```

## **javax.swing.undo**

### **Interface StateEditable**

```
public abstract interface StateEditable
```

这个接口描述了具有可撤消或可重复编辑状态的对象

#### **Field**

```
static String RCSID
```

#### **Methods**

```
void restoreState (Hashtable state)
void storeState (Hashtable state)
```

### **Interface UndoableEdit**

```
public abstract interface UndoableEdit
```

这个接口描述了一个可撤消的编辑操作，它由一个可撤消的编辑对象实现。

#### **Methods**

```
boolean addEdit (UndoableEdit anEdit)
boolean canRedo ()
boolean canUndo ()
void die ()
String getPresentationName ()
String getRedoPresentationName ()
String getUndoPresentationName ()
boolean isSignificant ()
void redo ()
boolean replaceEdit (UndoableEdit anEdit)
void undo ()
```

### **AbstractUndoableEdit 类**

```
java.lang.Object
|
+-- javax.swing.undo.AbstractUndoableEdit
public class AbstractUndoableEdit
    extends Object
    implements UndoableEdit, Serializable
```

这个抽象类代表一个可撤消的编辑操作，它所实现的方法用对应的接口返回逻辑值。

#### **Fields**

```
protected static String RedoName
```

```
protected static String UndoName
```

### Constructor

```
AbstractUndoableEdit ()
```

### Methods

boolean	addEdit (UndoableEdit anEdit)
boolean	canRedo ()
boolean	canUndo ()
void	die ()
String	getPresentationName ()
String	getRedoPresentationName ()
String	getUndoPresentationName ()
boolean	isSignificant ()
void	redo ()
boolean	replaceEdit (UndoableEdit anEdit)
String	toString ()
void	undo ()

## CompoundEdit 类

```
java.lang.Object
|
+-- javax.swing.undo.AbstractUndoableEdit
    |
    +-- javax.swing.undo.CompoundEdit
public class CompoundEdit
    extends AbstractUndoableEdit
```

这个类表示一个可撤销的混合编辑操作，它事实上就是多个可撤销编辑操作的组合。

### Field

```
protected Vector edits
```

### Constructor

```
CompoundEdit ()
```

### Methods

boolean	addEdit (UndoableEdit anEdit)
boolean	canRedo ()
boolean	canUndo ()
void	die ()
void	end ()
String	getPresentationName ()
String	getRedoPresentationName ()
String	getUndoPresentationName ()
boolean	isInProgress ()
boolean	isSignificant ()
protected UndoableEdit	lastEdit ()

```

void           redo ()
String         toString ()
void          undo ()

```

## StateEdit 类

```

java.lang.Object
|
+-- javax.swing.undo.AbstractUndoableEdit
    |
    +-- javax.swing.undo.StateEdit
public class StateEdit
    extends AbstractUndoableEdit

```

这个类表示对象的一个普通编辑，它经历一个状态变化。

### Fields

protected StateEditable	object
protected Hashtable	postState
protected Hashtable	preState
protected static String	RCSID
protected String	undoRedoName

### Constructors

```

StateEdit (StateEditable anObject)
StateEdit (StateEditable anObject, String name)

```

### Methods

void	end ()
String	getPresentationName ()
protected void	init (StateEditable anObject, String name)
void	redo ()
protected void	removeRedundantState ()
void	undo ()

## UndoableEditSupport 类

```

java.lang.Object
|
+-- javax.swing.undo.UndoableEditSupport
public class UndoableEditSupport
    extends Object

```

这个类表示一个可撤消编辑支持，它用于管理 UndoableEdit 监听类。

### Fields

protected CompoundEdit	compoundEdit
protected Vector	listeners
protected Object	realSource

```
protected int updateLevel
```

### Constructors

```
UndoableEditSupport()
UndoableEditSupport (Object r)
```

### Methods

protected void	postEdit (UndoableEdit e)
void	addUndoableEditListener (UndoableEditListener l)
void	beginUpdate ()
protected CompoundEdit	createCompoundEdit ()
void	endUpdate ()
int	getUpdateLevel ()
void	postEdit (UndoableEdit e)
void	removeUndoableEditListener (UndoableEditListener l)
String	toString ()

## UndoManager 类

```
java.lang.Object
|
+-- javax.swing.undo.AbstractUndoableEdit
    |
    +-- javax.swing.undo.CompoundEdit
        |
        +-- javax.swing.undo.UndoManager
```

```
public class UndoManager
    extends CompoundEdit
    implements UndoableEditListener
```

这个类表示一个撤消管理器，它是 CompoundEdit 的子类。

### Constructor

```
UndoManager ()
```

### Methods

boolean	addEdit (UndoableEdit anEdit)
boolean	canRedo ()
boolean	canUndo ()
boolean	canUndoOrRedo ()
void	discardAllEdits ()
protected UndoableEdit	editToBeRedone ()
protected UndoableEdit	editToBeUndone ()
void	end ()
int	getLimit ()
String	getRedoPresentationName ()
String	getUndoOrRedoPresentationName ()

```

String           getUndoPresentationName ()
void            redo ()
protected void  redoTo (UndoableEdit edit)
void            setLimit (int l)
String          toString ()
protected void  trimEdits (int from, int to)
protected void  trimForLimit ()
void            undo ()
void            undoableEditHappened (UndoableEditEvent e)
void            undoOrRedo ()
protected void  undoTo (UndoableEdit edit)

```

## CannotRedoException 类

```

java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- java.lang.RuntimeException
|
+-- javax.swing.undo.CannotRedoException
public class CannotRedoException
    extends RuntimeException

```

这个类表示一个异常，当一个重复操作不能在一个可撤消编辑中实现时，就会抛出它。

### Constructor

```
CannotRedoException ()
```

## CannotUndoException 类

```

java.lang.Object
|
+-- java.lang.Throwable
|
+-- java.lang.Exception
|
+-- java.lang.RuntimeException
|
+-- javax.swing.undo.CannotUndoException
public class CannotUndoException
    extends RuntimeException

```

这个类表示一个例外，当一个撤消操作不能在一个可撤消编辑中实现时，就会抛出它。

### Constructor

```
CannotUndoException ()
```

## 附录 B AWT 事件类快速参考

此附录包含有关 AWT 事件类和事件监听类的快速参考资料。这些接口和类由某些 Swing 组件使用。例如，Swing 按钮也能触发 ActionEvent 类型的事件。ActionEvent 类型的相应监听类被注册为该事件的源程序。

### 程序包综述

程序包 `java.awt.event` 包含的接口和类处理由 AWT 和 Swing 组件触发的不同类型的事件。

#### Interfaces

`ActionListener`  
`AdjustmentListener`  
`AWTEventListener`  
`ComponentListener`  
`ContainerListener`  
`FocusListener`  
`InputMethodListener`  
`ItemListener`  
`KeyListener`  
`MouseListener`  
`MouseMotionListener`  
`TextListener`  
`WindowListener`

#### Classes

<code>ActionEvent</code>	<code>ItemEvent</code>
<code>AdjustmentEvent</code>	<code>KeyAdapter</code>
<code>ComponentAdapter</code>	<code>KeyEvent</code>
<code>ComponentEvent</code>	<code>MouseAdapter</code>
<code>ContainerAdapter</code>	<code>MouseEvent</code>
<code>ContainerEvent</code>	<code>MouseMotionAdapter</code>
<code>FocusAdapter</code>	<code>PaintEvent</code>
<code>FocusEvent</code>	<code>TextEvent</code>
<code>InputEvent</code>	<code>WindowAdapter</code>
<code>InputMethodEvent</code>	<code>WindowEvent</code>
<code>InvocationEvent</code>	

## 接口与类的细节

### ActionListener 接口

```
public abstract interface ActionListener  
    extends EventListener
```

这个接口描述了一个动作监听类。此类具有接收 ActionEvent 类型事件的能力。

#### Method

```
void      actionPerformed (ActionEvent e)
```

### AdjustmentListener 接口

```
public abstract interface AdjustmentListener  
    extends EventListener
```

这个接口描述了一个调整监听类。此类具有接收 AdjustmentEvent 类型事件的能力。

#### Method

```
void      adjustmentValueChanged (AdjustmentEvent e)
```

### AWTEventListener 接口

```
public abstract interface AWTEventListener  
    extends EventListener
```

这个接口描述了一个具有监视 AWT 事件功能的对象。此实现类的对象具有接收 AWTEvent 类型事件的能力，而且使用方法 addAWTEventListener () 注册为 Toolkit。一般情况下不一定要使用这个接口。

#### Method

```
void      eventDispatched (AWTEvent event)
```

### ComponentListener 接口

```
public abstract interface ComponentListener  
    extends EventListener
```

这个接口描述了一个监听类，它具有接收组件事件的能力。这些事件通知监听类有关布局中组件的移动、调整大小、显示或隐藏。

#### Methods

```
void      componentHidden (ComponentEvent e)  
void      componentMoved (ComponentEvent e)  
void      componentResized (ComponentEvent e)  
void      componentShown (ComponentEvent e)
```

### ContainerListener 接口

```
public abstract interface ContainerListener
```

```
extends EventListener
```

这个接口描述了一个监听类对象，它具有接收容器事件的能力。

### Methods

```
void componentAdded (ContainerEvent e)  
void componentRemoved (ContainerEvent e)
```

## FocusListener 接口

```
public abstract interface FocusListener  
extends EventListener
```

这个接口描述了一个监听类对象，它具有接收焦点事件的能力。

### Methods

```
void focusGained (FocusEvent e)  
void focusLost (FocusEvent e)
```

## InputMethodListener 接口

```
public abstract interface InputMethodListener  
extends EventListener
```

这个接口描述了一个监听类对象，它具有接收输入方法事件的能力。当光标位置改变或输入文本改变时，文本组件就会触发此输入方法事件。

### Methods

```
void caretPositionChanged (InputMethodEvent event)  
void inputMethodTextChanged (InputMethodEvent event)
```

## ItemListener 接口

```
public abstract interface ItemListener  
extends EventListener
```

这个接口描述了一个监听类对象，它具有接收项目事件的能力。

### Method

```
void itemStateChanged (ItemEvent e)
```

## KeyListener 接口

```
public abstract interface KeyListener  
extends EventListener
```

这个接口描述了一个按键监听类，当按键被按下、释放或键入时它具有接收事件的能力。

### Methods

```
void keyPressed (KeyEvent e)  
void keyReleased (KeyEvent e)  
void keyTyped (KeyEvent e)
```

## MouseListener 接口

```
public abstract interface MouseListener  
    extends EventListener
```

这个接口描述了一个监听类，当鼠标键被按下、释放或单击时，以及当鼠标进入或离开组件区域时，它具有接收事件的能力。

### Methods

```
void      mouseClicked (MouseEvent e)  
void      mouseEntered (MouseEvent e)  
void      mouseExited (MouseEvent e)  
void      mousePressed (MouseEvent e)  
void      mouseReleased (MouseEvent e)
```

## MouseMotionListener 接口

```
public abstract interface MouseMotionListener  
    extends EventListener
```

这个接口描述了一个监听类，它具有接收鼠标动作事件的能力。

### Methods

```
void      mouseDragged (MouseEvent e)  
void      mouseMoved (MouseEvent e)
```

## TextListener 接口

```
public abstract interface TextListener  
    extends EventListener
```

这个接口描述的监听类，在一个组件中的文本值发生改变时，它具有接收事件的能力。

### Method

```
void      textValueChanged (TextEvent e)
```

## WindowListener 接口

```
public abstract interface WindowListener  
    extends EventListener
```

这个接口描述了一个监听类对象，当一个窗口被打开、关闭、最小化或恢复窗口时，它能够接收该窗口对象所发出的事件。

### Methods

```
void      windowActivated (WindowEvent e)  
void      windowClosed (WindowEvent e)  
void      windowClosing (WindowEvent e)  
void      windowDeactivated (WindowEvent e)  
void      windowDeiconified (WindowEvent e)
```

```
void windowIconified (WindowEvent e)
void windowOpened (WindowEvent e)
```

## ActionEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ActionEvent
public class ActionEvent
    extends AWTEvent
```

这个类表示一个动作事件，当在某一组件中执行一个动作时，就会运行它。

### Fields

static int	ACTION_FIRST
static int	ACTION_LAST
static int	ACTION_PERFORMED
static int	ALT_MASK
static int	CTRL_MASK
static int	META_MASK
static int	SHIFT_MASK

### Constructors

```
ActionEvent (Object source, int id, String command)
ActionEvent (Object source, int id, String command, int modifiers)
```

### Methods

String	getActionCommand ()
int	getModifiers ()
String	paramString ()

## AdjustmentEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.AdjustmentEvent
public class AdjustmentEvent
    extends AWTEvent
```

这个类表示一个事件，当在某一组件的可调整值发生改变时，就会触发它。

### Fields

static int	ADJUSTMENT_FIRST
------------	------------------

```

static int      ADJUSTMENT_LAST
static int      ADJUSTMENT_VALUE_CHANGED
static int      BLOCK_DECREMENT
static int      BLOCK_INCREMENT
static int      TRACK
static int      UNIT_DECREMENT
static int      UNIT_INCREMENT

```

**Constructor**

AdjustmentEvent (Adjustable source, int id, int type, int value)

**Methods**

Adjustable	getAdjustable ()
int	getAdjustmentType ()
int	getValue ()
String	paramString ()

**ComponentAdapter 类**

```

java.lang.Object
|
+
+ - java.awt.event.ComponentAdapter
直接的已知子类：
BasicSliderUI.ComponentHandler, BasicTreeUI.ComponentHandler,
JViewport.ViewListener
public abstract class ComponentAdapter
    extends Object
    implements ComponentListener

```

这是一个适配器类，它可以被子类化用来接收组件事件。

**Constructor**

ComponentAdapter ()

**Methods**

void	componentHidden (ComponentEvent e)
void	componentMoved (ComponentEvent e)
void	componentResized (ComponentEvent e)
void	componentShown (ComponentEvent e)

**ComponentEvent 类**

```

java.lang.Object
|
+
+ - java.util.EventObject
|
+
+ - java.awt.AWTEvent
|
+
+ - java.awt.event.ComponentEvent
public class ComponentEvent
    extends AWTEvent

```

这个类表示一个组件事件，当在某一组件执行移动、调整大小等操作时，就会触发它。

### Fields

static int	COMPONENT_FIRST
static int	COMPONENT_HIDDEN
static int	COMPONENT_LAST
static int	COMPONENT_MOVED
static int	COMPONENT_RESIZED
static int	COMPONENT_SHOWN

### Constructor

```
ComponentEvent (Component source, int id)
```

### Methods

Component	getComponent ()
String	paramString ()

## ContainerAdapter 类

```
java.lang.Object
|
+
+ - java.awt.event.ContainerAdapter
public abstract class ContainerAdapter
    extends Object
    implements ContainerListener
```

这个类表示一个适配器类，它可以被子类化用来接收容器事件。

### Constructor

```
ContainerAdapter ()
```

### Methods

void	componentAdded (ContainerEvent e)
void	componentRemoved (ContainerEvent e)

## ContainerEvent 类

```
java.lang.Object
|
+
+ - java.util.EventObject
|
+
+ - java.awt.AWTEvent
|
+
+ - java.awt.event.ComponentEvent
|
+
+ - java.awt.event.ContainerEvent
public class ContainerEvent
    extends ComponentEvent
```

这个类表示一个容器事件，当一组件被加入或从容器中删除时，就会触发它。

### Fields

static int	COMPONENT_ADDED
------------	-----------------

```

static int      COMPONENT_REMOVED
static int      CONTAINER_FIRST
static int      CONTAINER_LAST

```

**Constructor**

ContainerEvent (Component source, int id, Component child)

**Methods**

```

Component      getChild ()
Container      getContainer ()
String         paramString ()

```

**FocusAdapter 类**

```

java.lang.Object
|
+
+ - java.awt.event.FocusAdapter
public abstract class FocusAdapter
    extends Object
    implements FocusListener

```

这是一个适配器类，它可以用其子类化用来接收焦点事件。

**Constructor**

FocusAdapter ()

**Methods**

```

void          focusGained (FocusEvent e)
void          focusLost (FocusEvent e)

```

**FocusEvent 类**

```

java.lang.Object
|
+
+ - java.util.EventObject
|
+
+ - java.awt.AWTEvent
|
+
+ - java.awt.event.ComponentEvent
|
+
+ - java.awt.event.FocusEvent
public class FocusEvent
    extends ComponentEvent

```

这个类表示一个焦点事件，当一组件获得或失去键盘事件焦点时，就会发出它。

**Fields**

```

static int      FOCUS_FIRST
static int      FOCUS_GAINED
static int      FOCUS_LAST
static int      $FOCUS_LOST

```

## Constructors

```
FocusEvent (Component source, int id)
FocusEvent (Component source, int id, boolean temporary)
```

## Methods

boolean	isTemporary ()
String	paramString ()

## InputEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.InputEvent
public abstract class InputEvent
    extends ComponentEvent
```

这个类表示组件输入事件的基础类。

## Fields

static int	ALT_GRAPH_MASK
static int	ALT_MASK
static int	BUTTON1_MASK
static int	BUTTON2_MASK
static int	BUTTON3_MASK
static int	CTRL_MASK
static int	META_MASK
static int	SHIFT_MASK

## Methods

void	consume ()
int	getModifiers ()
long	getWhen ()
boolean	isAltDown ()
boolean	isAltGraphDown ()
boolean	isConsumed ()
boolean	isControlDown ()
boolean	isMetaDown ()
boolean	isShiftDown ()

## InputMethodEvent 类

```
java.lang.Object
|
+ - java.util.EventObject
|
+ - java.awt.AWEEvent
|
+ - java.awt.event.InputMethodEvent
```

```
public class InputMethodEvent
extends AWEEvent
```

这个类表示一个输入方法事件，当文本开始改变时，就会触发它。

### Fields

static int	CARET_POSITION_CHANGED
static int	INPUT_METHOD_FIRST
static int	INPUT_METHOD_LAST
static int	INPUT_METHOD_TEXT_CHANGED

### Constructors

```
InputMethodEvent (Component source, int id, AttributedCharacterIterator text,
int committedCharacterCount, TextHitInfo caret,
TextHitInfo visiblePosition)
InputMethodEvent (Component source, int id, TextHitInfo caret,
TextHitInfo visiblePosition)
```

### Methods

void	consume ()
TextHitInfo	getCaret ()
Int	getCommittedCharacterCount ()
AttributedCharacterIterator	getText ()
TextHitInfo	getVisiblePosition ()
boolean	isConsumed ()
String	paramString ()

## InvocationEvent 类

```
java.lang.Object
|
+ - java.util.EventObject
|
+ - java.awt.AWEEvent
|
+ - java.awt.event.InvocationEvent
```

```
public class InvocationEvent
extends AWEEvent
implements ActiveEvent
```

这个类表示一个调用事件，它具有在线程中调用 run() 方法的能力。

### Fields

protected boolean	catchExceptions
static int	INVOCATION_DEFAULT
static int	INVOCATION_FIRST
static int	INVOCATION_LAST
protected Object	notifier
protected	
Runnable	
runnable	

### Constructors

```
protected InvocationEvent (Object source, int id, Runnable runnable,
    Object notifier, boolean catchExceptions)
InvocationEvent (Object source, Runnable runnable)
InvocationEvent (Object source, Runnable runnable, Object notifier,
    boolean catchExceptions)
```

### Methods

void	dispatch ()
Exception	getException ()
String	paramString ()

## ItemEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ItemEvent

public class ItemEvent
    extends AWTEvent
```

这个类表示一个项目事件，当一个项目在项目可选组件中被选定或取消选定时，就会由一个组件发出它。

### Fields

static int	DESELECTED
static int	ITEM_FIRST
static int	ITEM_LAST
static int	ITEM_STATE_CHANGED
static int	SELECTED

### Constructor

```
ItemEvent (ItemSelectable source, int id, Object item, int stateChange)
```

**Methods**

Object	getItem ()
ItemSelectable	getItemSelectable ()
Int	getStateChange ()
String	paramString ()

**KeyAdapter 类**

```
java.lang.Object
|
+-- java.awt.event.KeyAdapter
public abstract class KeyAdapter
    extends Object
    implements KeyListener
```

这个类表示一个适配器类，当键盘被操作时，它可以用其子类来接收事件。

**Constructor**

```
KeyAdapter ()
```

**Methods**

void	keyPressed (KeyEvent e)
void	keyReleased (KeyEvent e)
void	keyTyped (KeyEvent e)

**KeyEvent 类**

```
java.lang.Object
|
+-- java.util.EventObject
    |
    +-- java.awt.AWTEvent
        |
        +-- java.awt.event.ComponentEvent
            |
            +-- java.awt.event.InputEvent
                |
                +-- java.awt.event.KeyEvent
public class KeyEvent
    extends InputEvent
```

这个类表示一个事件，当键盘上发生一个击键时，就会发出它。

**Fields**

static char	CHAR_UNDEFINED
static int	KEY_FIRST
static int	KEY_LAST
static int	KEY_PRESSED
static int	KEY_RELEASED

```
static int      KEY_TYPED
static int      VK_0
static int      VK_1
static int      VK_2
static int      VK_3
static int      VK_4
static int      VK_5
static int      VK_6
static int      VK_7
static int      VK_8
static int      VK_9
static int      VK_A
static int      VK_ACCEPT
static int      VK_ADD
static int      VK AGAIN
static int      VK_ALL_CANDIDATES
static int      VK_ALPHANUMERIC
static int      VK_ALT
static int      VK_ALT_GRAPH
static int      VK_AMPERAND
static int      VK_ASTERISK
static int      VK_AT
static int      VK_B
static int      VK_BACK_QUOTE
static int      VK_BACK_SLASH
static int      VK_BACK_SPACE
static int      VK_BRACELEFT
static int      VK_BRACERIGHT
static int      VK_C
static int      VK_CANCEL
static int      VK_CAPS_LOCK
static int      VK_CIRCUMFLEX
static int      VK_CLEAR
static int      VK_CLOSE_BRACKET
static int      VK_CODE_INPUT
static int      VK_COLON
static int      VK_COMMA
static int      VK_COMPOSE
static int      VK_CONTROL
static int      VK_CONVERT
static int      VK_COPY
static int      VK_CUT
static int      VK_D
```

---

static int	VK_DEAD_ABOVEDOT
static int	VK_DEAD_ABOVERING
static int	VK_DEAD_ACUTE
static int	VK_DEAD_BREVE
static int	VK_DEAD_CARON
static int	VK_DEAD_CEDILLA
static int	VK_DEAD_CIRCUMFLEX
static int	VK_DEAD_DIAERESIS
static int	VK_DEAD_DOUBLEACUTE
static int	VK_DEAD_GRAVE
static int	VK_DEAD_IOTA
static int	VK_DEAD_MACRON
static int	VK_DEAD_OGONEK
static int	VK_DEAD_SEMIVOICED_SOUND
static int	VK_DEAD_TILDE
static int	VK_DEAD_VOICED_SOUND
static int	VK_DECIMAL
static int	VK_DELETE
static int	VK_DIVIDE
static int	VK_DOLLAR
static int	VK_DOWN
static int	VK_E
static int	VK_END
static int	VK_ENTER
static int	VK_EQUALS
static int	VK_ESCAPE
static int	VK_EURO_SIGN
static int	VK_EXCLAMATION_MARK
static int	VK_F
static int	VK_F1
static int	VK_F10
static int	VK_F11
static int	VK_F12
static int	VK_F13
static int	VK_F14
static int	VK_F15
static int	VK_F16
static int	VK_F17
static int	VK_F18
static int	VK_F19
static int	VK_F2
static int	VK_F20
static int	VK_F21

```
static int      VK_F22
static int      VK_F23
static int      VK_F24
static int      VK_F3
static int      VK_F4
static int      VK_F5
static int      VK_F6
static int      VK_F7
static int      VK_F8
static int      VK_F9
static int      VK_FINAL
static int      VK_FIND
static int      VK_FULLSCREEN_WIDTH
static int      VK_G
static int      VK_GREATER
static int      VK_H
static int      VK_HALF_WIDTH
static int      VK_HELP
static int      VK_HIRAGANA
static int      VK_HOME
static int      VK_I
static int      VK_INSERT
static int      VK_INVERTED_EXCLAMATION_MARK
static int      VK_J
static int      VK_JAPANESE_HIRAGANA
static int      VK_JAPANESE_KATAKANA
static int      VK_JAPANESE_ROMAN
static int      VK_K
static int      VK_KANA
static int      VK_KANJI
static int      VK_KATAKANA
static int      VK_KP_DOWN
static int      VK_KP_LEFT
static int      VK_KP_RIGHT
static int      VK_KP_UP
static int      VK_L
static int      VK_LEFT
static int      VK_LEFT_PARENTHESIS
static int      VK_LESS
static int      VK_M
static int      VK_META
static int      VK_MINUS
static int      VK_MODECHANGE
```

static int	VK_MULTIPLY
static int	VK_N
static int	VK_NONCONVERT
static int	VK_NUM_LOCK
static int	VK_NUMBER_SIGN
static int	VK_NUMPAD0
static int	VK_NUMPAD1
static int	VK_NUMPAD2
static int	VK_NUMPAD3
static int	VK_NUMPAD4
static int	VK_NUMPAD5
static int	VK_NUMPAD6
static int	VK_NUMPAD7
static int	VK_NUMPAD8
static int	VK_NUMPAD9
static int	VK_O
static int	VK_OPEN_BRACKET
static int	VK_P
static int	VK_PAGE_DOWN
static int	VK_PAGE_UP
static int	VK_PASTE
static int	VK_PAUSE
static int	VK_PERIOD
static int	VK_PLUS
static int	VK_PREVIOUS_CANDIDATE
static int	VK_PRINTSCREEN
static int	VK_PROPS
static int	VK_Q
static int	VK_QUOTE
static int	VK_QUOTEDBL
static int	VK_R
static int	VK_RIGHT
static int	VK_RIGHT_PARENTHESIS
static int	VK_ROMAN_CHARACTERS
static int	VK_S
static int	VK_SCROLL_LOCK
static int	VK_SEMICOLON
static int	VK_SEPARATOR
static int	VK_SHIFT
static int	VK_SLASH
static int	VK_SPACE
static int	VK_STOP
static int	VK_SUBTRACT

static int	VK_T
static int	VK_TAB
static int	VK_U
static int	VK_UNDEFINED
static int	VK_UNDERSCORE
static int	VK_UNDO
static int	VK_UP
static int	VK_V
static int	VK_W
static int	VK_X
static int	VK_Y
static int	VK_Z

## Constructors

```
KeyEvent (Component source, int id, long when, int modifiers, int keyCode)
KeyEvent (Component source, int id, long when, int modifiers, int keyCode,
          char keyChar)
```

## Methods

char	getKeyChar ()
Int	getKeyCode ()
static String	getKeyModifiersText (int modifiers)
static String	getKeyText (int keyCode)
Boolean	isActionKey ()
String	paramString ()
void	setKeyChar (char keyChar)
void	setKeyCode (int keyCode)
void	setModifiers (int modifiers)

## MouseAdapter 类

```
java.lang.Object
|
+
+ - java.awt.event.MouseAdapter
public abstract class MouseAdapter
    extends Object
    implements MouseListener
```

这个类表示一个适配器，它可用其子类来接收鼠标事件。

## Constructor

```
MouseAdapter ()
```

## Methods

void	licked (MouseEvent e)
void	ntered (MouseEvent e)

```

void      xited (MouseEvent e)
void      ressed (MouseEvent e)
void      eleased (MouseEvent e)

```

## MouseEvent 类

```

java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.InputEvent
|
+-- java.awt.event.MouseEvent

```

```

public class ent
    extends InputEvent

```

代表一个鼠标事件，当鼠标按钮被操作或鼠标被拖动时，就会触发它。

### Fields

static int	MOUSE_CLICKED
static int	MOUSE_DRAGGED
static int	MOUSE_ENTERED
static int	MOUSE_EXITED
static int	MOUSE_FIRST
static int	MOUSE_LAST
static int	MOUSE_MOVED
static int	MOUSE_PRESSED
static int	MOUSE_RELEASED

### Constructor

```

MouseEvent (Component source, int id, long when, int modifiers, int x,
int y, int clickCount, boolean popupTrigger)

```

### Methods

int	getClickCount ()
Point	getPoint ()
Tnt	getX ()
Int	getY ()
Boolean	isPopupTrigger ()
String	paramString ()
void	translatePoint (int x, int y)

## MouseMotionAdapter 类

```
java.lang.Object
|
+-- java.awt.event.MouseMotionAdapter
public abstract class MouseMotionAdapter
    extends Object
    implements MouseMotionListener
```

这个类表示一个适配器类，它可以被子类化用来接收鼠标事件。

### Constructor

MouseMotionAdapter ()

### Methods

void	mouseDragged (MouseEvent e)
void	mouseMoved (MouseEvent e)

## PaintEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWEEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.PaintEvent
public class PaintEvent
    extends ComponentEvent
```

这个类表示一个画图事件，用来确保 paint () 和 update () 方法在事件对列中串行排列。

### Fields

static int	PAINT
static int	PAINT_FIRST
static int	PAINT_LAST
static int	UPDATE

### Constructor

PaintEvent (Component source, int id, Rectangle updateRect)

### Methods

Rectangle	getUpdateRect ()
String	paramString ()
void	setUpdateRect (Rectangle updateRect)

```

Class TextEvent
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWEEvent
|
+-- java.awt.event.TextEvent

```

```

public class TextEvent
    extends AWEEvent

```

这个类表示一个事件，当文本组件中的文本发生改变时，就会发出它。

### Fields

```

static int      TEXT_FIRST
static int      TEXT_LAST
static int      TEXT_VALUE_CHANGED

```

### Constructor

```
TextEvent (Object source, int id)
```

### Method

```

String          paramString ()
Class WindowAdapter
java.lang.Object
|
+-- java.awt.event.WindowAdapter
public abstract class WindowAdapter
    extends Object
    implements WindowListener

```

这个类表示一个适配器，它可以被子类化用来接收窗口事件。

### Constructor

```
WindowAdapter ()
```

### Methods

```

void           windowActivated (WindowEvent e)
void           windowClosed (WindowEvent e)
void           windowClosing (WindowEvent e)
void           windowDeactivated (WindowEvent e)
void           windowDeiconified (WindowEvent e)
void           windowIconified (WindowEvent e)
void           windowOpened (WindowEvent e)

```

## WindowEvent 类

```
java.lang.Object
|
+-- java.util.EventObject
|
+-- java.awt.AWTEvent
|
+-- java.awt.event.ComponentEvent
|
+-- java.awt.event.WindowEvent
public class WindowEvent
    extends ComponentEvent
```

这个类表示一个窗口事件，当一个窗口被打开、关闭、缩小成图标以及由图标状态恢复为窗口状态时，就会激活这个事件。

### Fields

static int	WINDOW_ACTIVATED
static int	WINDOW_CLOSED
static int	WINDOW_CLOSING
static int	WINDOW_DEACTIVATED
static int	WINDOW_DEICONIFIED
static int	WINDOW_FIRST
static int	WINDOW_ICONIFIED
static int	WINDOW_LAST
static int	WINDOW_OPENED

### Constructor

WindowEvent (Window source, int id)

### Methods

Window	getWindow ()
String	paramString ()

# 附录 C Swing 中重要的 AWT 类

## 程序包综述

这一部分的参考包含了 AWT 程序包 `java.awt` 中重要的接口和类，它们通常用于 `Swing` 程序中。下面是这些类和接口的列表。

### Interfaces

LayoutManager	641
LayoutManager2	642

### Classes

Component	642
Container	647
FlowLayout	649
BorderLayout	650
GridLayout	651
CardLayout	652
GridBagLayout	652
GridBagConstraints	654

## 接口与类的细节

### LayoutManager 接口

```
public abstract interface LayoutManager
```

这个接口代表抽象的布局管理器。其实现类为用于一个容器的特定布局模式定义了方法。

#### Methods:

void	addLayoutComponent (String name, Component comp)
void	layoutContainer (Container parent)
Dimension	minimumLayoutSize (Container parent)
Dimension	preferredLayoutSize (Container parent)
void	removeLayoutComponent (Component comp)

### LayoutManager2 接口

```
public abstract interface DLayoutManager2
```

extends LayoutManager

这是一个由布局类实现的接口，它基于一定的布局约束条件来放置容器。

### Methods

void	addLayoutComponent (Component comp, Object constraints)
float	getLayoutAlignmentX (Container target)
float	getLayoutAlignmentY (Container target)
void	invalidateLayout (Container target)
Dimension	maximumLayoutSize (Container target)

## Component 类

```
java.lang.Object
  |
  +-- java.awt.Component
public abstract class Component
    extends Object
    implements ImageObserver, MenuContainer, Serializable
```

这是所有 AWT 组件的父组件。这个类也在 Swing 组件的基础类的层次结构中。

### Fields

static float	BOTTOM_ALIGNMENT
static float	CENTER_ALIGNMENT
static float	LEFT_ALIGNMENT
static float	RIGHT_ALIGNMENT
static float	TOP_ALIGNMENT

### Constructor

protected Component ()

### Methods

boolean	action (Event evt, Object what)
void	add (PopupMenu popup)
void	addComponentListener (ComponentListener l)
void	addFocusListener (FocusListener l)
void	addInputMethodListener (InputMethodListener l)
void	addKeyListener (KeyListener l)
void	addMouseListener (MouseListener l)
void	addMouseMotionListener (MouseMotionListener l)
void	addNotify ()
void	addPropertyChangeListener ↳ (PropertyChangeListener listener)
void	addPropertyChangeListener (String propertyName, ↳ PropertyChangeListener listener)
Rectangle	bounds ()

```
//Deprecated. As of JDK version 1.1, replaced by
//getBounds () .
int
int
protected AWEvent
boolean
boolean
Image
Image
void
void
protected void
void
void
void
void
protected void
void
protected void
float
float
Color
Rectangle
Rectangle
ColorModel
Component
Component
ComponentOrientation
Cursor
DropTarget
Font
FontMetrics
checkImage (Image image, ImageObserver observer)
checkImage (Image image, int width, int height,
    ➔ImageObserver observer)
coalesceEvents (AWEvent existingEvent, AWEvent
    ➔NewEvent)
contains (int x, int y)
contains (Point p)
createImage (ImageProducer producer)
createImage (int width, int height)
deliverEvent (Event e)
//Deprecated. As of JDK version 1.1, replaced by
//dispatchEvent (AWEvent e) .
disable ()
//Deprecated. As of JDK version 1.1, replaced by
//setEnabled (boolean) .
disableEvents (long eventsToDisable)
dispatchEvent (AWEvent e)
doLayout ()
enable ()
//Deprecated. As of JDK version 1.1, replaced by
//setEnabled (boolean) .
enable (boolean b)
//Deprecated. As of JDK version 1.1, replaced by
//setEnabled (boolean) .
enableEvents (long eventsToEnable)
enableInputMethods (boolean enable)
firePropertyChange (String propertyName, Object
    ➔oldValue, Object newValue)
getAlignmentX ()
getAlignmentY ()
getBackground ()
getBounds ()
getBounds (Rectangle rv)
getColorModel ()
getComponentAt (int x, int y)
getComponentAt (Point p)
getComponentOrientation ()
getCursor ()
getDropTarget ()
getFont ()
getFontMetrics (Font font)
```

```
Color           getForeground ()
Graphics        getGraphics ()
int             getHeight ()
InputContext    getInputContext ()
InputMethodRequests  getInputMethodRequests ()
Locale          getLocale ()
Point           getLocation ()
Point           getLocation (Point rv)
Point           getLocationOnScreen ()
Dimension       getMaximumSize ()
Dimension       getMinimumSize ()
String          getName ()
Container       getParent ()
java.awt.peer.ComponentPeer  getPeer ()
                           //Deprecated. As of JDK version 1.1, programs
                           //should not directly manipulate peers.
                           //Replaced by boolean isDisplayable () .
Dimension       getPreferredSize ()
Dimension       getSize ()
Dimension       getSize (Dimension rv)
Toolkit         getToolkit ()
Object          getTreeLock ()
int             getWidth ()
int             getX ()
int             getY ()
boolean         gotFocus (Event evt, Object what)
                           //Deprecated. As of JDK version 1.1, replaced by
                           //processFocusEvent (FocusEvent) .
boolean         handleEvent (Event evt)
                           //Deprecated. As of JDK version 1.1, replaced by
                           //processEvent (AWTEvent) .
boolean         hasFocus ()
void            hide ()
                           //Deprecated. As of JDK version 1.1, replaced by
                           //setVisible (boolean) .
boolean         imageUpdate (Image img, int flags, int x,
                           int y, int w, int h)
boolean         inside (int x, int y)
                           //Deprecated. As of JDK version 1.1, replaced by
                           //contains (int, int) .
void            invalidate ()
boolean         isDisplayable ()
boolean         isDoubleBuffered ()
```

```
boolean isEnabled ()
boolean isFocusTraversable ()
boolean isLightweight ()
boolean isOpaque ()
boolean isShowing ()
boolean isValid ()
boolean isVisible ()
boolean keyDown (Event evt, int key)
//Deprecated. As of JDK version 1.1, replaced by
//processKeyEvent (KeyEvent) .
boolean keyUp (Event evt, int key)
//Deprecated. As of JDK version 1.1, replaced by
//processKeyEvent (KeyEvent) .
void layout ()
//Deprecated. As of JDK version 1.1, replaced by
//doLayout () .
void list ()
void list (PrintStream out)
void list (PrintStream out, int indent)
void list (PrintWriter out)
void list (PrintWriter out, int indent)
Component locate (int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseEvent (MouseEvent) .
Point location ()
//Deprecated. As of JDK version 1.1, replaced by
//getLocation () .
boolean lostFocus (Event evt, Object what)
//Deprecated. As of JDK version 1.1, replaced by
//processFocusEvent (FocusEvent) .
Dimension minimumSize ()
//Deprecated. As of JDK version 1.1, replaced by
//getMinimumSize () .
boolean mouseDown (Event evt, int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseEvent (MouseEvent) .
boolean mouseDrag (Event evt, int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseMotionEvent (MouseEvent) .
boolean mouseEnter (Event evt, int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseEvent (MouseEvent) .
boolean mouseExit (Event evt, int x, int y)
```

```
//Deprecated. As of JDK version 1.1, replaced by
//processMouseEvent (MouseEvent) .
boolean mouseMove (Event evt, int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseMotionEvent (MouseEvent) .
boolean mouseUp (Event evt, int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//processMouseEvent (MouseEvent) .
void move (int x, int y)
//Deprecated. As of JDK version 1.1, replaced by
//setLocation (int, int) .
void nextFocus ()
//Deprecated. As of JDK version 1.1, replaced by
//transferFocus () .
void paint (Graphics g)
void paintAll (Graphics g)
protected String paramString ()
boolean postEvent (Event e)
//Deprecated. As of JDK version 1.1, replaced by
//dispatchEvent (AWTEvent) .
Dimension preferredSize ()
//Deprecated. As of JDK version 1.1, replaced by
//getPreferredSize () .
boolean prepareImage (Image image, ImageObserver observer)
boolean prepareImage (Image image, int width, int height,
➥ ImageObserver observer)
void print (Graphics g)
void printAll (Graphics g)
protected void processComponentEvent (ComponentEvent e)
protected void processEvent (AWTEvent e)
protected void processFocusEvent (FocusEvent e)
protected void processInputMethodEvent (InputMethodEvent e)
protected void processKeyEvent (KeyEvent e)
protected void processMouseEvent (MouseEvent e)
protected void processMouseMotionEvent (MouseEvent e)
void remove (MenuComponent popup)
void removeComponentListener (ComponentListener l)
void removeFocusListener (FocusListener l)
void removeInputMethodListener
➥ (InputMethodListener l)
void removeKeyListener (KeyListener l)
void removeMouseListener (MouseListener l)
void removeMouseMotionListener
```

```
    ➔ (MouseMotionListener l)
void        removeNotify ()
void        removePropertyChangeListener
    ➔ (PropertyChangeListener listener)
void        removePropertyChangeListener (String
    ➔PropertyName, PropertyChangeListener
    ➔Listener)
void        repaint ()
void        repaint (int x, int y, int width, int height)
void        repaint (long tm)
void        repaint (long tm, int x, int y, int width, int
height)
void        requestFocus ()
void        reshape (int x, int y, int width, int height)
//Deprecated. As of JDK version 1.1, replaced by
//setBounds (int, int, int, int) .
void        resize (Dimension d)
//Deprecated. As of JDK version 1.1, replaced by
//setSize (Dimension d) .
void        resize (int width, int height)
//Deprecated. As of JDK version 1.1, replaced by
//setSize (int, int) .
void        setBackground (Color c)
void        setBounds (int x, int y, int width, int height)
void        setBounds (Rectangle r)
void        setComponentOrientation
    ➔ (ComponentOrientation o)
void        setCursor (Cursor cursor)
void        setDropTarget (DropTarget dt)
void        setEnabled (boolean b)
void        setFont (Font f)
void        setForeground (Color c)
void        setLocale (Locale l)
void        setLocation (int x, int y)
void        setLocation (Point p)
void        setName (String name)
void        setSize (Dimension d)
void        setSize (int width, int height)
void        setVisible (boolean b)
void        show ()
//Deprecated. As of JDK version 1.1, replaced by
//setVisible (boolean) .
void        show (boolean b)
```

```

    //Deprecated. As of JDK version 1.1, replaced by
    //setVisible (boolean) .
Dimension      size ()

    //Deprecated. As of JDK version 1.1, replaced by
    //getSize () .

String         toString ()

void           transferFocus ()

void           update (Graphics g)

void           validate ()

```

## Container 类

```

java.lang.Object
|
+-- java.awt.Component
|
+-- java.awt.Container

public class Container
    extends Component

```

这个类是抽象窗口工具箱 (AWT) 中的一个普通容器。它是 Swing 父类 JComponent 的父类。因为是类 JComponent 扩展了 Container，所以这个 Swing 组件可以象容器一样操作。

### Constructor

Container ()

### Methods

Component	add (Component comp)
Component	add (Component comp, int index)
void	add (Component comp, Object constraints)
void	add (Component comp, Object constraints, int index)
Component	add (String name, Component comp)
void	addContainerListener (ContainerListener l)
protected void	addImpl (Component comp, Object constraints, int index)
void	addNotify ()
int	countComponents ()          //Deprecated. As of JDK version 1.1, replaced by         //getComponentCount () .
void	deliverEvent (Event e)          //Deprecated. As of JDK version 1.1, replaced by         //dispatchEvent (AWTEvent e) .
void	doLayout ()
Component	findComponentAt (int x, int y)
Component	findComponentAt (Point p)
float	getAlignmentX ()
float	getAlignmentY ()

```
Component      getComponent ( int n )
Component      getComponentAt ( int x, int y )
Component      getComponentAt ( Point p )
int            getComponentCount ( )
Component [ ]  getComponents ( )
Insets         getInsets ( )
LayoutManager   getLayout ( )
Dimension     getMaximumSize ( )
Dimension     getMinimumSize ( )
Dimension     getPreferredSize ( )
Insets        insets ( )
                           //Deprecated. As of JDK version 1.1, replaced by
                           //getInsets ( ) .
void           invalidate ( )
boolean        isAncestorOf ( Component c )
void           layout ( )
                           //Deprecated. As of JDK version 1.1, replaced by
                           //doLayout ( ) .
void           list ( PrintStream out, int indent )
void           list ( PrintWriter out, int indent )
Component     locate ( int x, int y )
                           //Deprecated. As of JDK version 1.1, replaced by
                           //getComponentAt ( int, int ) .
Dimension     minimumSize ( )
                           //Deprecated. As of JDK version 1.1, replaced by
                           //getMinimumSize ( ) .
void           paint ( Graphics g )
void           paintComponents ( Graphics g )
protected String paramString ( )
Dimension     preferredSize ( )
                           //Deprecated. As of JDK version 1.1, replaced by
                           //getPreferredSize ( ) .
void           print ( Graphics g )
void           printComponents ( Graphics g )
protected void processContainerEvent ( ContainerEvent e )
protected void processEvent ( AWTEvent e )
void           remove ( Component comp )
void           remove ( int index )
void           removeAll ( )
void           removeContainerListener ( ContainerListener l )
void           removeNotify ( )
void           setFont ( Font f )
void           setLayout ( LayoutManager mgr )
```

```

void           update (Graphics g)
void           validate ()
protected void validateTree ()

```

## FlowLayout 类

```

java.lang.Object
|
+
+-- java.awt.FlowLayout
public class FlowLayout
    extends Object
    implements LayoutManager, Serializable

```

这个类表示的布局管理器叫做流动布局。当这个布局被设计为容器时，该组件被从左加到右、从上加到下，如果没有组件被加到同一行的话。

### Fields

static int	CENTER
static int	LEADING
static int	LEFT
static int	RIGHT
static int	TRAILING

### Constructors

```

FlowLayout ()
FlowLayout (int align)
FlowLayout (int align, int hgap, int vgap)

```

### Methods

void	addLayoutComponent (String name, Component comp)
int	getAlignment ()
int	getHgap ()
int	getVgap ()
void	layoutContainer (Container target)
Dimension	minimumLayoutSize (Container target)
Dimension	preferredLayoutSize (Container target)
void	removeLayoutComponent (Component comp)
void	setAlignment (int align)
void	setHgap (int hgap)
void	setVgap (int vgap)
String	toString ()

## BorderLayout 类

```

java.lang.Object
|
+
+-- java.awt.BorderLayout
public class BorderLayout
    extends Object
    implements LayoutManager2, Serializable

```

这个布局代表的布局管理器叫做边界布局。当这个布局管理器被设计为容器时，就从四个方位北、南、东、西和中心位置加上组件。

### Fields

static String	AFTER_LAST_LINE
static String	AFTER_LINE_ENDS
static String	BEFORE_FIRST_LINE
static String	BEFORE_LINE_BEGINS
static String	CENTER
static String	EAST
static String	NORTH
static String	SOUTH
static String	WEST

### Constructors

```
BorderLayout ()
BorderLayout (int hgap, int vgap)
```

### Methods

void	addLayoutComponent (Component comp, Object constraints)
void	addLayoutComponent (String name, Component comp)
	// Deprecated. As of JDK version 1.1, replaced by
	// addLayoutComponent (Component, Object) .
int	getHgap ()
float	getLayoutAlignmentX (Container parent)
float	getLayoutAlignmentY (Container parent)
int	getVgap ()
void	invalidateLayout (Container target)
void	layoutContainer (Container target)
Dimension	maximumLayoutSize (Container target)
Dimension	minimumLayoutSize (Container target)
Dimension	preferredLayoutSize (Container target)
void	removeLayoutComponent (Component comp)
void	setHgap (int hgap)
void	setVgap (int vgap)
String	toString ()

## GridLayout 类

```
java.lang.Object
|
+-- java.awt.GridLayout
public class GridLayout
    extends Object
    implements LayoutManager, Serializable
```

这个类表示的布局管理器叫做网格布局。当这个类型的布局被设计为容器时，就会加上网格模型。

### Constructors

```
GridLayout ()
GridLayout (int rows, int cols)
GridLayout (int rows, int cols, int hgap, int vgap)
```

### Methods

void	addLayoutComponent (String name, Component comp)
int	getColumns ()
int	getHgap ()
int	getRows ()
int	getVgap ()
void	layoutContainer (Container parent)
Dimension	minimumLayoutSize (Container parent)
Dimension	preferredLayoutSize (Container parent)
void	removeLayoutComponent (Component comp)
void	setColumns (int cols)
void	setHgap (int hgap)
void	setRows (int rows)
void	setVgap (int vgap)
String	toString ()

## CardLayout 类

```
java.lang.Object
|
+-- java.awt.CardLayout
public class CardLayout
    extends Object
    implements LayoutManager2, Serializable
```

这个类表示的布局管理器叫做卡片布局。当这个布局管理器被设计为容器时，组件就会被叠加到互相重叠的不同卡片上。每次只能看到一张卡片。

### Constructors

```
CardLayout ()
CardLayout (int hgap, int vgap)
```

### Methods

void	addLayoutComponent (Component comp, Object constraints)
void	addLayoutComponent (String name, Component comp)
void	first (Container parent)
int	getHgap ()
float	getLayoutAlignmentX (Container parent)
float	getLayoutAlignmentY (Container parent)
int	getVgap ()

void	invalidateLayout (Container target)
void	last (Container parent)
void	layoutContainer (Container parent)
Dimension	maximumLayoutSize (Container target)
Dimension	minimumLayoutSize (Container parent)
void	next (Container parent)
Dimension	preferredLayoutSize (Container parent)
void	previous (Container parent)
void	removeLayoutComponent (Component comp)
void	setHgap (int hgap)
void	setVgap (int vgap)
void	show (Container parent, String name)
String	toString ()

## GridBagLayout 类

```

java.lang.Object
|
+-- java.awt.GridBagLayout
public class GridBagLayout
    extends Object
    implements LayoutManager2, Serializable

```

这个类表示的布局管理器叫做无序网格布局。当这个布局管理器被设计为容器时，就会基于指定的约束条件来加上组件。此约束条件由 `GridBagConstraints` 类型的约束对象代表。这些组件不需要具有同样的大小，它们可以按要求调整、锁定和填补。

### Fields

double []	columnWeights
int []	columnWidths
protected Hashtable	comptable
protected GridBagConstraints	defaultConstraints
protected java.awt.GridBagLayoutInfo	layoutInfo
protected static int	MAXGRIDSIZE
protected static int	MINSIZE
protected static int	PREFERREDSIZE
int []	rowHeights
double []	rowWeights

### Constructor

`GridBagLayout ()`

### Methods

void	addLayoutComponent (Component comp, Object constraints)
void	addLayoutComponent (String name,

```

protected void                                         Component comp)
                                                     AdjustForGravity (GridBagConstraints
                                                     constraints, Rectangle r)
                                                     ArrangeGrid (Container parent)
                                                     getConstraints (Component comp)
                                                     getLayoutAlignmentX (Container parent)
                                                     getLayoutAlignmentY (Container parent)
                                                     getLayoutDimensions ()
                                                     GetLayoutInfo (Container parent, int
                                                     sizeflag)
                                                     getLayoutOrigin ()
                                                     getLayoutWeights ()
                                                     GetMinSize (Container parent,
                                                     java.awt.GridBagLayoutInfo info)
                                                     invalidateLayout (Container target)
                                                     layoutContainer (Container parent)
                                                     location (int x, int y)
                                                     lookupConstraints (Component comp)
                                                     maximumLayoutSize (Container target)
                                                     minimumLayoutSize (Container parent)
                                                     preferredLayoutSize (Container parent)
                                                     removeLayoutComponent (Component comp)
                                                     setConstraints (Component comp,
                                                     GridBagConstraints constraints)
                                                     toString ()

```

## GridBagConstraints 类

```

java.lang.Object
|
+-- java.awt.GridBagConstraints
public class GridBagConstraints
    extends Object
    implements Cloneable, Serializable

```

这个类表示约束条件对象。这个类提供各种约束条件来指定对组件的约束，这些组件是按照无序网格布局来安排的。

### Fields

int	anchor
static int	BOTH
static int	CENTER
static int	EAST
int	fill
int	gridheight

---

int	gridwidth
int	gridx
int	gridy
static int	HORIZONTAL
Insets	insets
int	ipadx
int	ipady
static int	NONE
static int	NORTH
static int	NORTHEAST
static int	NORTHWEST
static int	RELATIVE
static int	REMAINDER
static int	SOUTH
static int	SOUTHEAST
static int	SOUTHWEST
static int	VERTICAL
double	weightx
double	weighty
static int	WEST

### Constructors

GridBagConstraints ()  
GridBagConstraints (int gridx, int gridy, int gridwidth, int gridheight,  
double weightx, double weighty, int anchor, int fill, Insets insets,  
int ipadx, int ipady)

### Method

Object clone ()

## 附录 D 辅助选项快速参考

### 程序包综述

程序包 `javax.accessibility` 支持的接口和类精简了用户接口组件和提供对这些组件的访问的辅助技术系统。

### 接口

`Accessible`  
`AccessibleAction`  
`AccessibleComponent`  
`AccessibleHypertext`  
`AccessibleSelection`  
`AccessibleText`  
`AccessibleValue`

### 类

`AccessibleBundle`  
`AccessibleContext`  
`AccessibleHyperlink`  
`AccessibleResourceBundle`  
`AccessibleRole`  
`AccessibleState`  
`AccessibleStateSet`

### 接口与类的细节

#### 接口辅助选项

```
public abstract interface Accessible
```

这个接口由支持辅助选项的所有组件实现。

#### 方法

```
AccessibleContext getAccessibleContext ()
```

#### AccessibleAction 接口

```
public abstract interface AccessibleAction
```

这个接口描述了一个辅助技术系统的机制，它与实现某个动作的对象相联系。此接口也能决定一个对象的动作。

#### Methods

boolean	doAccessibleAction (int i)
int	getAccessibleActionCount ()
String	getAccessibleActionDescription (int i)

### AccessibleComponent 接口

```
public abstract interface AccessibleComponent
```

这个接口描述了一个辅助技术系统的机制，它决定某个对象的图形代表并给它赋值。在屏幕上绘制的任何组件都应实现这个接口。

#### Methods

void	addFocusListener (FocusListener l)
boolean	contains (Point p)
Accessible	getAccessibleAt (Point p)
Color	getBackground ()
Rectangle	getBounds ()
Cursor	getCursor ()
Font	getFont ()
FontMetrics	getFontMetrics (Font f)
Color	getForeground ()
Point	getLocation ()
Point	getLocationOnScreen ()
Dimension	getSize ()
boolean	isEnabled ()
boolean	isFocusTraversable ()
boolean	isShowing ()
boolean	isVisible ()
void	removeFocusListener (FocusListener l)
void	requestFocus ()
void	setBackground (Color c)
void	setBounds (Rectangle r)
void	setCursor (Cursor cursor)
void	setEnabled (boolean b)
void	setFont (Font f)
void	setForeground (Color c)
void	setLocation (Point p)
void	setSize (Dimension d)
void	setVisible (boolean b)

### AccessibleHypertext 接口

```
public abstract interface AccessibleHypertext
```

---

```
extends AccessibleText
```

这个接口描述了一个辅助技术的机制，它通过其属性和它的位置访问超文本。它也提供处理超链接的含义。

### Methods

AccessibleHyperlink	getLink (int linkIndex)
int	getLinkCount ()
int	getLinkIndex (int charIndex)

## AccessibleSelection 接口

```
public abstract interface AccessibleSelection
```

这个接口描述了一个决定当前选择的辅助技术的机制。

### Methods

void	addAccessibleSelection (int i)
void	clearAccessibleSelection ()
Accessible	getAccessibleSelection (int i)
int	getAccessibleSelectionCount ()
boolean	isAccessibleChildSelected (int i)
void	removeAccessibleSelection (int i)
void	selectAllAccessibleSelection ()

## AccessibleText 接口

```
public abstract interface AccessibleText
```

这个接口描述了附属文本，它由显示文本内容的所有类来实现。

### Fields

static int	CHARACTER
static int	SENTENCE
static int	WORD

### Methods

String	getAfterIndex (int part, int index)
String	getAtIndex (int part, int index)
String	getBeforeIndex (int part, int index)
int	getCaretPosition ()
AttributeSet	getCharacterAttribute (int i)
Rectangle	getCharacterBounds (int i)
int	getCharCount ()
int	getIndexAtPoint (Point p)
String	getSelectedText ()
int	getSelectionEnd ()
int	getSelectionStart ()

## AccessibleValue 接口

```
public abstract interface AccessibleValue
```

这个接口描述了一个辅助技术系统的机制，它决定其数值并给其赋值。

### Methods

Number	getCurrentAccessibleValue ()
Number	getMaximumAccessibleValue ()
Number	getMinimumAccessibleValue ()
boolean	setCurrentAccessibleValue (Number n)

## AccessibleBundle 类

```
java.lang.Object
|
+ - javax.accessibility.AccessibleBundle
public abstract class AccessibleBundle
    extends Object
```

这个类作为 AccessibleState 和 AccessibleRole 的超类，它被用来牢固地维持键入的枚举项。

### Field

```
protected String key
```

### Constructor

```
AccessibleBundle ()
```

### Methods

String	toDisplayString ()
String	toDisplayString (Locale locale)
protected String	toDisplayString (String resourceName, Locale locale)
String	toString ()

## AccessibleContext 类

```
java.lang.Object
|
+ - javax.accessibility.AccessibleContext
public abstract class AccessibleContext
    extends Object
```

这个类表示可访问环境，它是由可访问对象提供的基本信息。

### Fields

static String	ACCESSIBLE_ACTIVE_DESCENDANT_PROPERTY
static String	ACCESSIBLE_CARET_PROPERTY
static String	ACCESSIBLE_CHILD_PROPERTY
static String	ACCESSIBLE_DESCRIPTION_PROPERTY
static String	ACCESSIBLE_NAME_PROPERTY

static String	ACCESSIBLE_SELECTION_PROPERTY
static String	ACCESSIBLE_STATE_PROPERTY
static String	ACCESSIBLE_TEXT_PROPERTY
static String	ACCESSIBLE_VALUE_PROPERTY
static String	ACCESSIBLE_VISIBLE_DATA_PROPERTY
protected String	accessibleDescription
protected String	accessibleName
protected Accessible	accessibleParent

### Constructor

AccessibleContext ()

### Methods

void addPropertyChangeListener (PropertyChangeListener listener)
void firePropertyChange (String propertyName, Object oldValue, Object newValue)
AccessibleAction getAccessibleAction ()
abstract Accessible getAccessibleChild (int i)
abstract int getAccessibleChildrenCount ()
AccessibleComponent getAccessibleComponent ()
String getAccessibleDescription ()
abstract int getAccessibleIndexInParent ()
String getAccessibleName ()
Accessible getAccessibleParent ()
abstract AccessibleRole getAccessibleRole ()
AccessibleSelection getAccessibleSelection ()
abstract AccessibleStateSet getAccessibleStateSet ()
AccessibleText getAccessibleText ()
AccessibleValue getAccessibleValue ()
abstract Locale getLocale ()
void removePropertyChangeListener (PropertyChangeListener listener)
void setAccessibleDescription (String s)
void setAccessibleName (String s)
void setAccessibleParent (Accessible a)

## AccessibleHyperlink 类

```
java.lang.Object
|
+-- javax.accessibility.AccessibleHyperlink
public abstract class AccessibleHyperlink
    extends Object
    implements AccessibleAction
```

这个类表示在超文本文档中可访问的超链接。

### Constructor

AccessibleHyperlink ()

**Methods**

abstract boolean	doAccessibleAction (int i)
abstract Object	getAccessibleActionAnchor (int i)
abstract int	getAccessibleActionCount ()
abstract String	getAccessibleActionDescription (int i)
abstract Object	getAccessibleActionObject (int i)
abstract int	getEndIndex ()
abstract int	getStartIndex ()
abstract boolean	isValid ()

**AccessibleResourceBundle 类**

```

java.lang.Object
|
+-- java.util.ResourceBundle
|
+-- java.util.ListResourceBundle
|
+-- javax.accessibility.AccessibleResourceBundle
public class AccessibleResourceBundle
    extends ListResourceBundle

```

这个类表示保存局部化的字符串集合，这些字符串用于确定可访问性。

**Constructor**

```
AccessibleResourceBundle ()
```

**Method**

```
Object [][] getContents ()
```

**AccessibleRole 类**

```

java.lang.Object
|
+-- javax.accessibility.AccessibleBundle
|
+-- javax.accessibility.AccessibleRole
public class AccessibleRole
    extends AccessibleBundle

```

这个类表示决定组件作用的那些可访问的作用。请参看该类的相应字段下的作用列表。

**Fields**

static AccessibleRole	ALERT
static AccessibleRole	AWT_COMPONENT
static AccessibleRole	CHECK_BOX
static AccessibleRole	COLOR_CHOOSER
static AccessibleRole	COLUMN_HEADER
static AccessibleRole	COMBO_BOX

static AccessibleRole	DESKTOP_ICON
static AccessibleRole	DESKTOP_PANE
static AccessibleRole	DIALOG
static AccessibleRole	DIRECTORY_PANE
static AccessibleRole	FILE_CHOOSER
static AccessibleRole	FILLER
static AccessibleRole	GLASS_PANE
static AccessibleRole	INTERNAL_FRAME
static AccessibleRole	LABEL
static AccessibleRole	LAYERED_PANE
static AccessibleRole	LIST
static AccessibleRole	MENU
static AccessibleRole	MENU_BAR
static AccessibleRole	MENU_ITEM
static AccessibleRole	OPTION_PANE
static AccessibleRole	PAGE_TAB
static AccessibleRole	PAGE_TAB_LIST
static AccessibleRole	PANEL
static AccessibleRole	PASSWORD_TEXT
static AccessibleRole	POPUP_MENU
static AccessibleRole	PROGRESS_BAR
static AccessibleRole	PUSH_BUTTON
static AccessibleRole	RADIO_BUTTON
static AccessibleRole	ROOT_PANE
static AccessibleRole	ROW_HEADER
static AccessibleRole	SCROLL_BAR
static AccessibleRole	SCROLL_PANE
static AccessibleRole	SEPARATOR
static AccessibleRole	SLIDER
static AccessibleRole	SPLIT_PANE
static AccessibleRole	SWING_COMPONENT
static AccessibleRole	TABLE
static AccessibleRole	TEXT
static AccessibleRole	TOGGLE_BUTTON
static AccessibleRole	TOOL_BAR
static AccessibleRole	TOOL_TIP
static AccessibleRole	TREE
static AccessibleRole	UNKNOWN
static AccessibleRole	VIEWPORT
static AccessibleRole	WINDOW

### Constructor

protected      AccessibleRole (String key)

## AccessibleState 类

```
java.lang.Object
|
+-- javax.accessibility.AccessibleBundle
|
+-- javax.accessibility.AccessibleState
public class AccessibleState
    extends AccessibleBundle
```

这个类表示组件的可访问状态。请参看该类的相应字段下的任务列表。

### Fields

static AccessibleState	ACTIVE
static AccessibleState	ARMED
static AccessibleState	BUSY
static AccessibleState	CHECKED
static AccessibleState	COLLAPSED
static AccessibleState	EDITABLE
static AccessibleState	ENABLED
static AccessibleState	EXPANDABLE
static AccessibleState	EXPANDED
static AccessibleState	FOCUSABLE
static AccessibleState	FOCUSED
static AccessibleState	HORIZONTAL
static AccessibleState	ICONIFIED
static AccessibleState	MODAL
static AccessibleState	MULTI_LINE
static AccessibleState	MULTISELECTABLE
static AccessibleState	OPAQUE
static AccessibleState	PRESSED
static AccessibleState	RESIZABLE
static AccessibleState	SELECTABLE
static AccessibleState	SELECTED
static AccessibleState	SHOWING
static AccessibleState	SINGLE_LINE
static AccessibleState	TRANSIENT
static AccessibleState	VERTICAL
static AccessibleState	VISIBLE

### Constructor

```
protected AccessibleState (String key)
```