第5章 Swing容器的使用

Swing采用自顶向下的方式构建GUI,即先创建容器,再向容器中添加组件。通常父容器一旦创建并且添加了组件,以后不能随意改变。组件在创建时添加到父容器中,销毁时从父容器中删除。容器也是进行界面设计和布局的重要工具。在设计复杂界面时,可以切分成几个板块,不同板块使用不同类型的容器组织组件,从而简化设计。

窗体JFrame是最顶层的容器,运行时构成程序的窗口。关于窗体的设计2.2节较为详细地进行了介绍。NetBeans IDE的"组件"面板(Palette)中列出了八种常见的容器(见图5.1)。这些都是中间容器,它们置于顶层容器内帮助管理组件和进行布局,且不能在顶层容器之外独立存在。本章介绍其中主要容器组件的使用方法、属性设置及应用。

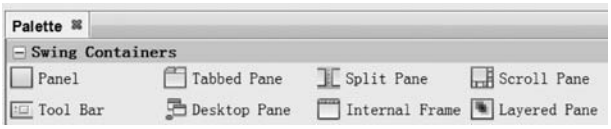


图5.1 "组件"面板中的Swing容器

5.1 面板容器

面板(JPanel)是一个轻量容器组件,也是最常用的中间容器,用于容纳界面元素。为面板设置适当的布局管理器,可以对组件进行不同的布局组合,可以通过容器的嵌套构建复杂的界面。

5.1.1 使用方法

创建一个面板的方法与创建其他组件相同。首先需要一个顶级容器,然后在该顶级容器中创建面板。

例如,先创建一个Java应用程序项目chap5,在该项目中创建包book.container.demo,再创建一个窗体(JFrame),类名为JPanelDemo。设置该窗体的布局为Box布局,Axis属性为Y Axis。

在Palette的Swing Containers组中单击Panel组件图标,然后将鼠标移到窗体中单击,即创建了一个面板组件jPanel1。

另一种创建面板的方法是,在IDE的主菜单中选择File→New File菜单项,在New File对话框中选择Swing GUI Forms类别,在文件类型列表中选择JPanel Form,单击Next按钮,输入类名,单击Finish按钮即可创建一个独立的面板。当设计完成该面板后进

行编译，然后从 Projects 窗口中将该面板文件拖放到其他容器，即可使它作为接收容器的一个面板组件使用。

与窗体一样，在面板中可以创建 Swing 控件和下级容器等组件，因此对面板要设置布局管理器，以便使面板对其中的组件进行合理布局。方法是，在 Navigator 窗口或 Design 视图中右击面板，在快捷菜单中选择 Set layout 菜单项，然后在级联菜单中选择一种合适的布局。

如果要向窗体中添加另一个面板，则在 Navigator 窗口的 JFrame 节点上右击，在出现的快捷菜单上选择 Add From Palette→Swing Containers→Panel 菜单项（见图 5.2）。否则很容易将第二个面板添加到第一个面板上。组件之间的层次关系从 Navigator 窗口的节点所属关系可以检查。

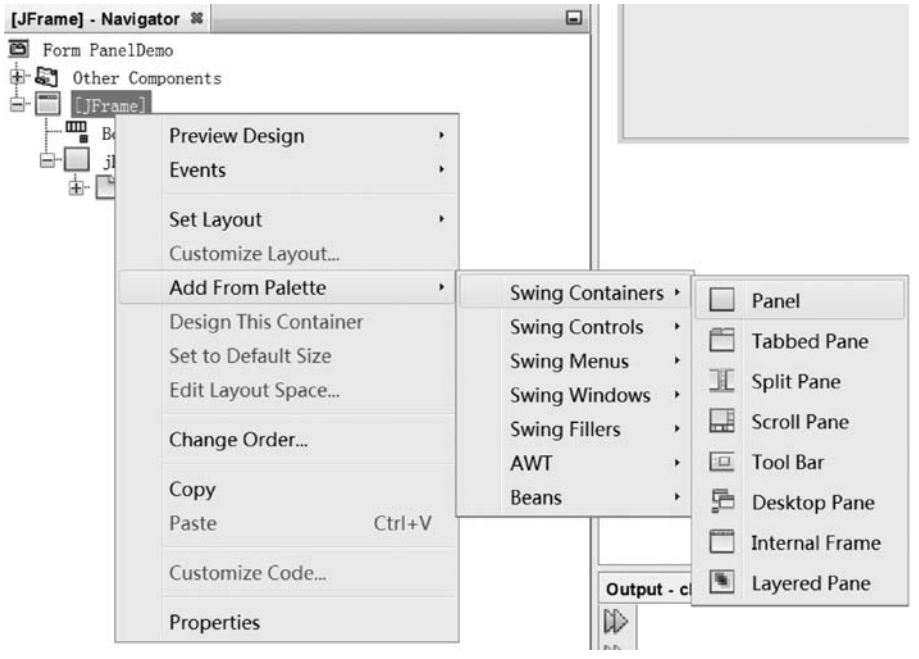


图 5.2　从快捷菜单添加面板组件

要在某一个已添加到窗体上的面板中进行界面设计，可以在 Navigator 窗口中双击这个面板节点（如 jPanel1），或者右击该面板，然后在快捷菜单中选择 Design This Container 命令，则 Design 视图中只显示该面板，之后可以在该面板上添加、移动、设置属性和删除组件。再次双击该面板，或右击该面板，然后在快捷菜单中选择 Design Parent 菜单的下级菜单列出的某个容器，则 Design 视图显示所选上层容器及其中的组件等。如图 5.3 中 jPanel1 是被选面板 JPanelDemo1 的直接父容器，[Top Parent]则是指顶层窗体[JFrame]（即 JPanelDemo）。

5.1.2　属性

面板首先是所在父容器中的一个组件，所以可以与其他 Swing 控件一样设置属性。主要属性包括背景色、前景色、边框、工具提示（toolTipText）等，其含义与用法与前面所述相

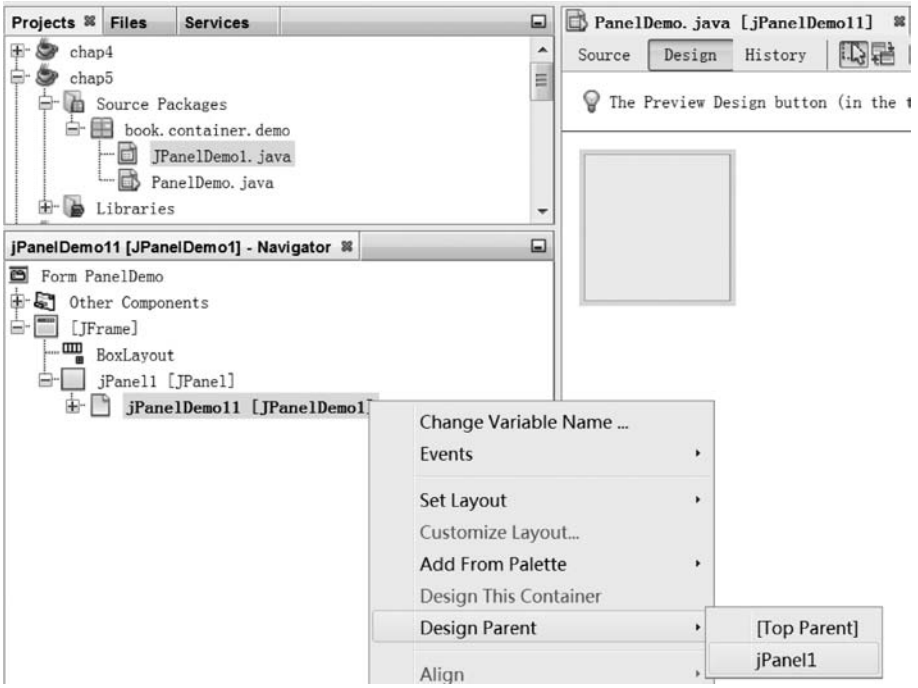


图 5.3　容器的层级关系及 Design Parent 菜单

同。面板的双缓冲 doubleBuffered 默认是打开的，该特性能够改进频繁变化的组件的显示效果。此外，如果面板所在容器采用 BoxLayout 布局和叠加布局，则 alignmentX 和 alignmentY 属性的效果与第 4 章所述一样。

如果面板所在容器采用网格包布局和 GroupLayout 布局（自由设计模式），该面板与其他组件的布局关系也会受到布局参数的制约。

面板中组件的布局决定于该面板所采用的布局管理器。一般地，如果面板中包含的组件较多，布局比较复杂，则往往需要拆分为多个面板，以便简化面板内的布局设计。

5.1.3　应用举例

例 5.1　为学生成绩管理系统设计学生注册界面，界面原型如图 5.4 所示。

分析：从图 5.4 可知，该界面用单一的容器和布局实现较为复杂。但可以把窗体分为上、中、下三部分，各用一个面板分别布局。上部面板显示标题信息，由于信息在面板中间显示，且只有一个标签组件，用边框式布局最为简单；中部面板设计登录信息输入界面，界面可以看作是 5 行 4 列的网格，可以用网格包布局，或者直接用自由设计模式；下部面板显示操作按钮，组件排列在一行，且不允许换行，按钮之间及按钮与左右边框之间有固定间距，可以用 BoxLayout 布局。

图 5.4　学生注册界面原型

解：按以下步骤操作。

（1）在 Projects 窗口中右击 StdScoreMana0.2 项目，在快捷菜单中单击 Copy 菜单项，在对话框中输入新项目名称“StdScoreMana0.3”，单击 Copy 按钮。

（2）展开 StdScoreMana0.3 项目中的 Source Packages 节点，右击 book.chap3.stdscoreui 包名，在快捷菜单中单击 Refactor→Rename 菜单项，新名称输入“book.stdscoreui”，单击 Refactor 按钮。

（3）在 book.stdscoreui 包名上右击，在快捷菜单中单击 New→JFrame Form 菜单项，类名输入“StdRegister”，单击 Finish 按钮。

（4）右击窗体 StdRegister，在快捷菜单中单击 Set Layout→Box Layout 菜单项。

（5）在 Navigator 窗口的 BoxLayout 节点上右击，选择 Properties 菜单项，在对话框中 Axis 属性右侧单击下三角按钮，选择 Y Axis，单击 Close 按钮。

（6）在 Navigator 窗口中的 JFrame 节点上右击，在快捷菜单中单击 Add From Palette→Swing Containers→Panel 菜单项。

（7）重复步骤(6)两次。

（8）在 Navigator 窗口中双击 jPanel1 节点，在 Properties 窗口中设置该面板的 maximumSize、minimumSize、preferredSize 属性的高度均为 60px。

（9）在 Design 视图中右击面板 jPanel1，在快捷菜单中单击 Set Layout→Border Layout 菜单项。

（10）单击 Palette 中的 Swing Controls 组中的 Label 组件，将鼠标移到 Design 视图面板 jPanel1 的中间部位单击（确保标签放置在 CENTER 位置）。

（11）在 Navigator 窗口中单击 jPanel1 下的 jLabel1 节点，在 Properties 窗口中设置该标签的 horizontalAlignment 属性为 CENTER、text 属性为“学生注册”、font 属性为宋体、粗体、18pt。

（12）在 Navigator 窗口中双击 jPanel3 节点，在 Properties 窗口中设置该面板的 maximumSize、minimumSize、preferredSize 属性的高度均为 60px。

（13）在 Design 视图中右击面板 jPanel3，在快捷菜单中单击 Set Layout→Box Layout 菜单项。

（14）单击 Palette 中的 Swing Controls 组中的 Button 组件，将鼠标移到 Design 视图中，按住 Shift 键单击 3 次。

（15）单击 Palette 中的 Swing Filler 组中的 Horizontal Strut 组件，输入宽度值 50，将鼠标移到最后一个按钮右侧单击。用同样方法在第一个和第二个、第二个和第三个按钮之间创建宽度为 30px 的 Horizontal Strut 组件。

（16）单击 Palette 中 Swing Filler 组中的 Horizontal Glue 组件，在第一个按钮左侧单击。

（17）依次修改 3 个按钮上的文字为“保存”“清除”和“关闭”。

（18）在 Navigator 窗口中双击 jPanel2 节点，接着右击，在快捷菜单中单击 Set Layout→Grid Bag Layout 菜单项。

（19）在 Design 视图中右击，在快捷菜单中单击 Customize Layout 菜单项。

（20）在 Customize Layout 对话框右上部网格区域的第一个网格单元(0,0)中右击，在快捷菜单中单击 Add Component→Swing Filler→Horizontal Glue 菜单项。其他网格单元

中的组件如图 5.5 所示，创建方法相同。其中，第 4 列(列标题为 3)的第 0、1、2、3、4 标题行网格单元均为宽度 50px 及高度 30px 的 Rigid Area 填充器，设置(0,5)网格单元标签 jLabel4 的 Grid Height 属性值为 3，(3,5)网格单元文本区域组件所在的 jScrollPane1 的网格 Grid Height 属性值为 2。(2,2)网格单元插入的是一个 Combo Box 组件，在此先占位置，对此组件及文本区域组件后面再介绍。第 3 列 0～4 行组件设置 Horizontal Fill 属性。

图 5.5　jPanel2 面板网格单元的组件分布

(21) 关闭 Customize Layout 对话框。在设计视图中按表 5.1 修改组件的变量名和文字。

表 5.1　jPanel2 面板网格单元组件的变量名和文字

	0	1	2	3	4	5	6
0		jLabelID,学号	jTextFieldID		jLabelPic,照片	jLabelImg	
1		jLabelName,姓名	jTextFieldName				
2		jLabelDept,专业	jComboBoxDept				
3		jLabelGrade,年级	jTextFieldGrade		jLabelInterest,兴趣	jScrollPaneInt	
4		jLabelClass,班级	jTextFieldClass				

(22) 修改 4 个文本字段 jTextFieldID、jTextFieldName、jTextFieldGrade、jTextFieldClass 和组合框 jComboBoxDept 的左插入量为 10，jTextFieldClass 和 jComboBoxDept 组件的右插入量为 20，jLabelPic 和 jLabelInterest 的右插入量为 10。

(23) 制作一幅图片(大小为 50×70px)，复制到项目的默认包下，设置 jLabelImg 组件的 icon 属性值为该图片。

至此，学生注册界面设计完成，运行程序发现，该界面表现比使用单一容器和单一布局的界面(如登录界面)更好。

5.2　滚 动 窗 格

某些情况下，界面中的组件需要占用的显示面积超过了容器能够显示的面积，这时 GUI 一般使用滚动条让用户通过移动可视区域在组件上的位置看到以前没有显示出来的部分。Panel 容器并未提供滚动条属性，也可不看到滚动条。事实上，Swing 提供了滚动窗格容器 JScrollPane，通过滚动条移动观察窗而显示超出显示面积的组件部分。

5.2.1 使用方法

滚动窗格也属于中间容器，需要一个窗体或其他顶级容器作为它的父容器。例如，在项目 chap5 的 book.container.demo 包中创建名为 ScrollPaneDemo 的 JFrame，设置为边框式布局。单击 Palette 面板 Swing Containers 组中的 Scroll Pane 组件，然后在窗体的中央部位单击，即创建了一个滚动窗格组件 jScrollPane1。单击 Palette→Swing Controls→Label 组件，再到 Design 视图的滚动窗格组件 jScrollPane1 上单击，创建一个标签 jLabel1。在 Properties 窗口中设置标签 jLabel1 的 icon 属性值为一幅大图片。此时，Design 视图的面板上出现滚动条（见图 5.6）。运行程序，当窗口缩小到一定程度时，出现水平和垂直滚动条，且移动滚动条可以看到图像的不同部分。

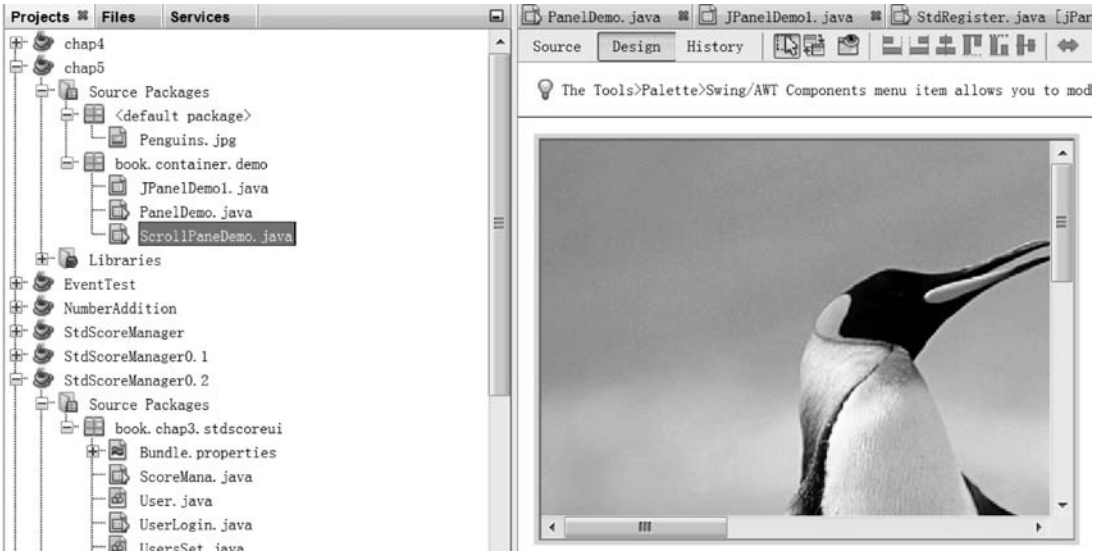


图 5.6 滚动面板设计视图

此外，例 5.1 中第(20)步骤在(3,5)网格单元创建文本区域组件时自动创建了一个滚动窗格 jScrollPane1，且在该滚动窗格中创建了一个文本区域组件 jTextArea1。亦即，在创建某些需要滚动条的组件时会自动创建滚动窗格，并在滚动窗格中创建该组件。这是 NetBeans IDE 的 GUI 构建器高效和智能的特性。

5.2.2 内部组成及属性设置

JScrollPane 提供轻量级组件的 scrollable 视图，由视口 JViewport（为数据源提供的一个窗口）、可选的垂直和水平滚动条 JScrollBars、可选的行标题和列标题（row header 和 column header）视口，以及它们之间的连线组成（见图 5.7）。注意，JScrollPane 不支持重量级组件。

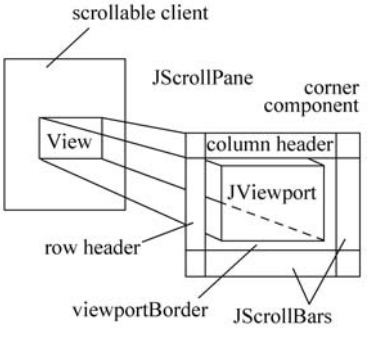


图 5.7 滚动窗格的组成

在视口自动左右滚动时，列标题（column header）跟踪主视口（JViewport）。行标题（row header）的滚动方

式与此类似。在两个滚动条的交汇处、行标题与列标题的交汇处，或者滚动条与其中一个标题的交汇处，留下一个默认情况下为空的矩形空间。四个角都有可能存在这些空间。在图5.7中，右上角存在该空间，由标签 corner component 标识。

滚动窗格的主要属性包括以下几个。

1. verticalScrollBarPolicy 和 horizontalScrollBarPolicy

设置垂直和水平滚动条的显示策略。在 Properties 窗口中该属性右侧下拉列表中提供了三个值：ALWAYS、NERVER 和 AS_NEEDED，分别指定始终显示、从不显示和需要时显示滚动条。

2. background

background 属性设置视口的背景色。此颜色可在主视口小于视口或透明时使用。设置 JViewport 而不是 JScrollPane 颜色的原因是，默认情况下 JViewport 为不透明，当 JScrollPane 绘制其背景时，视口通常将在它上面绘制并用视口背景色完全填充滚动窗格背景。

3. viewportBorder

要围绕主视口添加一个边界，可设置 viewportBorder 属性。单击 Properties 窗口 viewportBorder 属性右侧的“…”按钮，出现 viewportBorder 对话框（见图 5.8），在其中选择一种边框，在中间的 Properties 列表做适当设置，即可设置主视口的边框。

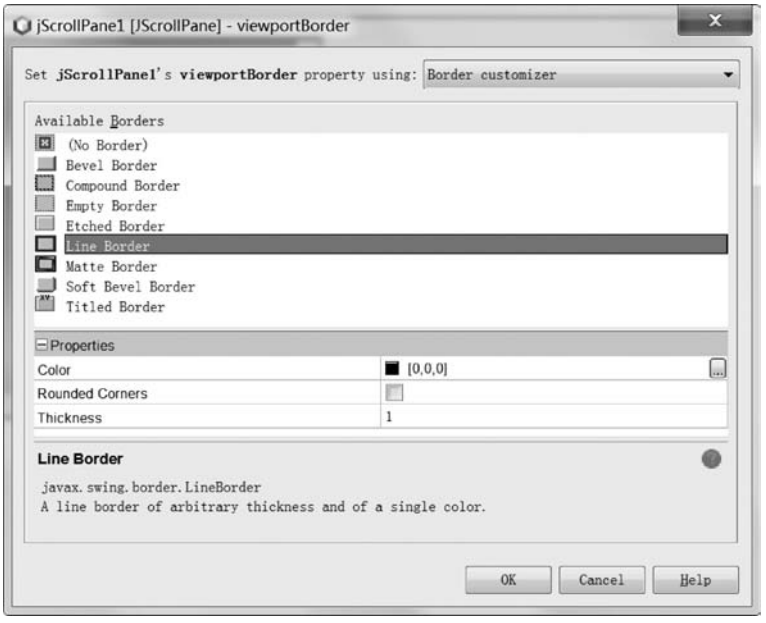


图 5.8　滚动窗格的 viewportBorder 对话框

4. preferredSize、maximumSize 和 minimumSize

设置滚动窗格的首选、最大和最小尺寸，方法与前面章节所述相同。

此外，看到 wheelScrollingEnabled 默认设置为选取状态，说明可以使用鼠标滚轮移动滚动条。

默认情况下，滚动窗格使用 ScrollPaneLayout 处理其子组件的布局。ScrollPaneLayout 使

用以下方法确定视口视图的大小：如果视图实现了 Scrollable，将使用 getPreferredScrollableViewportSize、getScrollableTracksViewportWidth 和 getScrollableTracks ViewportHeight 的组合，否则使用 getPreferredSize。

5.2.3 文本区域

文本区域 JTextArea 是一个显示纯文本的多行区域。它是一个轻量级组件，实现了 javax. swing. Scrollable 接口。通过把文本区域放置在滚动窗格内提供滚动条，从而支持长文档的显示和编辑。

除了与文本字段相同的属性之外，文本区域还有下列重要属性。

1. rows

该属性指定文本区域显示的首选行数。该属性值是一个正整数，在该属性右侧列单击之后直接输入。

2. columns

该属性指定文本区域显示的首选列数。该属性值是一个正整数，在该属性右侧列单击之后直接输入。

3. lineWrap

该属性指定一行字符数超过行的可显示列数时是否自动换行。单击该属性右侧复选框选取该属性，则提供自动换行功能。默认为不自动换行。

4. WrapStyleWord

该属性设置换行方式。如果设置为 true，则当行的长度大于所分配的宽度时，将在单词边界（空白）处换行。如果设置为 false，则将在字符边界处换行。此属性默认为 false。

5. tabSize

该属性设置转换 Tab 键为多少个空格字符。

5.3 拆分窗格

拆分窗格 JSplitPane 是一个中间容器组件，它把父容器的空间分隔成两个部分，并提供一条分隔条。可以通过拖动分隔条调整各部分的大小。拆分窗格可以嵌套在其他拆分窗格中，从而形成复杂的分隔空间。

5.3.1 使用方法

使用拆分窗格首先需要一个父容器。例如，创建一个空白的窗体 SplitPaneDemo，设置为边框式布局。单击 Palette→Swing Containers→Split Pane 组件，然后在窗体的中央部位单击，即创建了一个拆分窗格组件 jSplitPane1。一旦创建，该拆分窗格所在的父容器区域即被分隔条划分为左右两个部分，且每个部分都有一个按钮作为占位符，分别标有“左键”和“右键”字样。向其中一个部分添加某个组件时，该组件即取代占位符按钮。可以向每个分隔部分添加任何组件。特别地，如果添加的是另一个拆分窗格组件，则该部分又被分隔成两个部分，从而形成复杂的布局。设计时在 GUI 构建器中不能通过拖动分隔条调整两部分的分隔比例（或大小），但运行时可以拖动分隔条调整分隔比例。

5.3.2 属性

拆分窗格的属性对它的外观和行为都有明显的影响。在 Navigator 窗口中右击拆分窗格，在快捷菜单中选择 Properties 命令，则打开面板的 Properties 对话框（见图 5.9）。以下介绍主要属性。

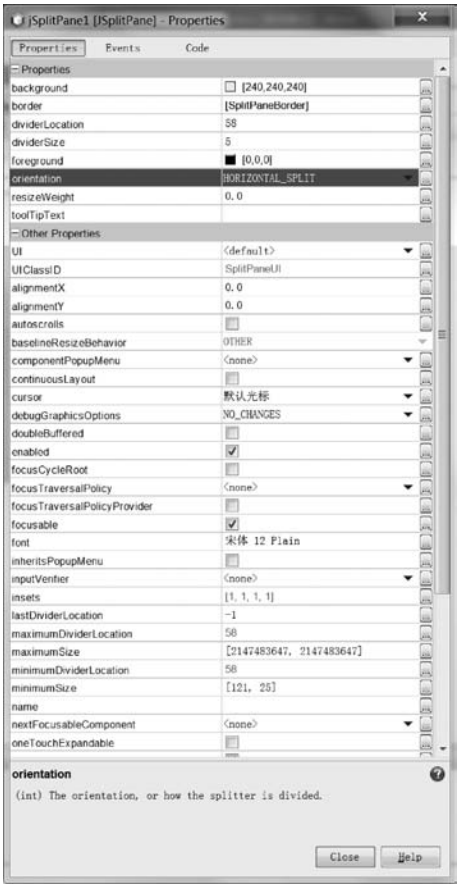


图 5.9　拆分窗格属性对话框

1. orientation

该属性设置分隔方向。取值为 HORIZONTAL_SPLIT，水平方向分隔为左右两个部分；取值为 VERTICAL _SPLIT，垂直方向分隔为上下两个部分。默认为水平分隔。

2. dividerLocation

该属性设置分隔条的位置。单击 dividerSize 属性行的右侧，输入一个整数值。这个值表示分隔条距离左边框（水平分隔）或上边框（垂直分隔）的绝对像素数。

3. dividerSize

该属性设置分隔条的大小（宽度或高度）。单击 dividerLocation 属性行的右侧，输入一个整数值。

4. 组件尺寸与 onTouchExpandable 属性

可以设置拆分窗格本身的首选、最大和最小尺寸。但更应该注意拆分窗格中包含的组

件的尺寸，因为则拆分窗格不允许用户拖动分隔条小于所包含组件的最小尺寸。如果组件的最小尺寸对于拆分窗格来说过大，就需要修改以满足拆分窗格的约束。

将拆分窗格的 onTouchExpandable 属性设置为 true（即处于选取状态），会在分隔条的上部（或左部）添加一个左右（或上下）箭头的“一触即展”图标（见图 5.10）。在单击分隔条上的向左箭头（向上箭头）时，将右部（或下部）的组件扩展为占据拆分窗格的全部尺寸；再次单击分隔条上的向右箭头，又会使分隔条回到其先前的位置。单击右箭头有同样效果。单击分隔条上此图标以外的位置会将分隔条定位到使得折叠的组件位于其最优尺寸处。

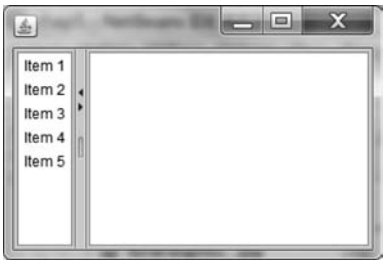


图 5.10　可扩展的分隔符

5. resizeWeight

如果在拆分窗格中存在其所包含的组件的最优尺寸所不需要的额外空间时，这个空间会依据 resizeWeight 属性设置进行分配。这个属性的初始设置为 0.0，意味着右边或是下边的组件会获得额外的空间。将这个设置修改为 1.0 会将所有的额外空间指定给左边或上边的组件；修改为 0.5 时则会在两个组件之间均分额外空间。

5.3.3　列表初步

列表 JList 是显示一组对象并且允许用户选择一个或多个项的组件。列表中一般提供了多个列表项，这些列表项可以是字符串，也可以是其他对象。列表 JList 提供了单独的模型 ListModel 维护其内容，但是本节不涉及这部分内容，而只介绍简单列表。列表的使用和维护比想象的也复杂，因此一般选项在五个以下时使用单选按钮或复选按钮更容易。

列表本身不具备滚动条及列表滚动功能，因此一般都是自动创建在滚动窗格上。在 GUI 构建器的 Palette 中单击 List 组件，然后在容器（如拆分窗格的左边）中单击，可以看到同时创建了滚动窗格及其内部的列表组件。

列表的主要属性如下。

1. model

该属性设置列表项。单击该属性右侧的“…”按钮，出现 model 对话框（见图 5.11）。此处的列表项都是以字符串形式使用，可以在其中添加、修改和删除列表项。

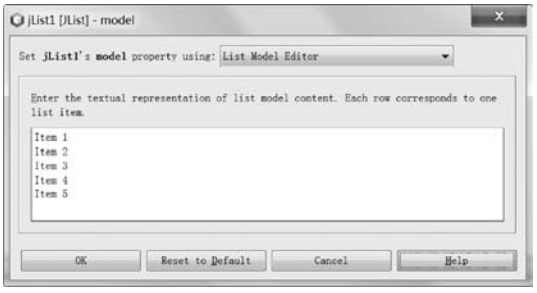


图 5.11　列表组件的 model 对话框

2. selectionMode

该属性设置列表项的选择模式。单击该属性行右侧的下三角按钮，出现以下三个选项。

SINGLE：一次只能选择一个列表项。

SINGLE_INTERVAL：一次只能选择一个连续区间的列表选项。运行时，用户选择列表项时按住 Shift 键选择一个连续区间，或按住 Ctrl 键连续单击相邻的几个选项。

MULTIPLE_INTERVAL：在此模式中，不存在对选择的限制，可以选择一个或多个连续或不连续的选项。此模式是默认设置。

3. fixedCellWidth

该属性定义单个列表项的显示宽度。默认－1 表示自动计算，输入一个大于 0 的像素值，则设置指定宽度。但不应小于最小宽度。

4. fixedCellHeight

该属性定义单个列表项的显示高度。默认－1 表示自动计算，输入一个大于 0 的像素值，则设置指定高度。

5. layoutOrientation

该属性定义布局列表项的方式。单击该属性行右侧的下三角按钮，有以下三种选项。

VERTICAL：在单个列中垂直布置列表项。这也是默认值。

HORIZONTAL_WRAP：水平布置列表项，根据需要可以换行显示列表项。如果 visibleRowCount 属性小于或等于 0，则包装由该列表的宽度确定；否则，以确保列表中 visibleRowCount 行的方式进行包装。

VERTICAL_WRAP：垂直布置列表项，根据需要将列表项包装到新列中。如果 visibleRowCount 属性小于或等于 0，则包装由该列表的宽度确定；否则，在 visibleRowCount 行进行包装。

6. visibleRowCount

该属性指定不需要滚动列表时，首选显示的列表项数。

7. selectedIndex 和 selectedIndices

该属性存储用户所选列表项的索引。如果列表框允许多选，则 selectedIndices 返回一个索引数组记录用户所选各列表项的索引。

8. selectedValue 和 selectedValues

该属性存储用户所选列表项的值。如果列表框允许多选，则 selectedValues 返回一个对象数组记录用户所选各列表项的值。

5.3.4 应用举例

例 5.2 设计一个类似资源管理器界面的文件阅读器程序，把窗体的整个客户区划分为左右两部分。左边列出文件目录，右边显示所选文本文件内容。

分析：窗体客户区的两部分划分可以使用拆分窗格实现。文件列表可以使用列表组件显示，右边窗格使用多行文本框显示文本文件内容。程序初始在文件列表中显示盘符（如 C:、D:等），如果用户单击选择的列表项是一个文本文件（扩展名为.txt），则读出文件内容并在右边的多行文本框中显示。如果是一个子目录，则将列表框中的内容更替为该目录下的目录列表。拆分窗格提供一触即展图标，以便在需要时最大化文件列表或文件内容显示区域。

解：按照以下步骤操作。

（1）新建 Java 应用程序项目，取名为 TextFileReader0.1。

（2）在该项目中新建 JFrame 窗体，类名为 MyFileReader，包名为 book.filereader。

（3）设置该窗体为边框式布局。

（4）单击 Palette→Swing Containers→Split Pane 组件，然后在窗体的中央部位单击，即创建了一个拆分窗格组件，并重命名为 jSplitPanefr。

（5）单击 Palette→Swing Controls→List 组件，然后将鼠标移到分隔窗格的“左键”部分单击，即创建了一个列表组件，并重命名为 jListFile。

（6）单击 Palette→Swing Controls→Text Area 组件，然后将鼠标移到分隔窗格的“右键”部分单击，并重命名为 jTextAreaText。

（7）在 Navigator 窗口中单击 jSplitPanefr 组件，在 Properties 窗口中单击 preferredSize 属性行右侧的“…”按钮，在对话框中输入宽度 800，高度 600。

（8）在 Properties 窗口中单击 minimumSize 属性行右侧的“…”按钮，在对话框中输入宽度 800，高度 600。

（9）在 Properties 窗口中单击 dividerLocation 属性行右侧，输入 280。

（10）在 Properties 窗口中单击 resizeWeight 属性行右侧，输入 0.3。

（11）在 Properties 窗口中单击 onTouchExpandable 属性右侧复选框，使其成为选取状态。

（12）在 Properties 窗口中单击列表组件 jListFile 的 model 属性右侧“…”按钮，接着单击 model 对话框中顶行的下拉箭头，选择 Custom Code 列表项，在代码输入区输入下列代码。

```
new javax.swing.AbstractListModel() {
    File[] files = File.listRoots();
    public int getSize() { return files.length; }
    public Object getElementAt(int i) { return files[i]; }
}
```

然后单击 OK 按钮。此段代码使列表框 jListFile 初始运行时显示系统中的盘符。

（13）在 jListFile 的 Properties 窗口中单击 selectionMode 属性右侧的下三角按钮，选择 SINGLE 列表项。

（14）在 Design 视图中右击 jListFile 列表组件，在快捷菜单中单击 Events→ListSelection→valueChanged 菜单项。在 Source 视图中设计 jListFileValueChanged 事件处理方法。

首先，在 MyFileReader 类中添加字段变量“String old="";”。

接着编写一个内部类，实现 AbstractListModel 抽象类的有关方法，代码如下。

```
private static class AbstractListModelImpl extends AbstractListModel {
    private final File[] files;
    public AbstractListModelImpl(File[] files) {
        this.files = files;
    }
    public int getSize() {
        return files.length;
    }
    public Object getElementAt(int i) {
```

```
            return files[i];
        }
    }
```

第三步，编写一个辅助方法，用于处理目录列表中的返回父目录相关问题，代码如下。

```
File[] sFile(File[] files) {
    File[] sfiles = new File[files.length + 1];
    sfiles[0] = new File("[返回]");
    for (int i = 0; i < files.length; i++) {
        sfiles[i + 1] = files[i];
    }
    return sfiles;
}
```

第四步，编写列表选择事件处理方法，代码如下。

```
private void jListFileValueChanged(javax.swing.event.ListSelectionEvent evt) {
    final File[] files;
    String str = "";
    if (!evt.getValueIsAdjusting()) {
        Object obj = jListFile.getSelectedValue();
        File file = (File)obj;
        if (file!= null && file.getName().equals("[返回]")) {
            file = new File(old).getParentFile();
            if(file == null)
                jListFile.setModel(new AbstractListModelImpl(File.listRoots()));
        }
        if (file != null && file.isDirectory()) {
            old = file.getAbsolutePath();
            files = sFile(file.listFiles());
            jListFile.setModel(new AbstractListModelImpl(files));
        } else if (file != null && (file.getAbsolutePath().endsWith(".txt") ||
                                    file.getAbsolutePath().endsWith(".TXT"))) {
            jTextAreaText.setText("");
            try {
                FileReader fr = new FileReader(file);
                BufferedReader br = new BufferedReader(fr);
                str = br.readLine();
                while (str != null) {
                    jTextAreaText.append(str + "\n");
                    str = br.readLine();
                }
            } catch (FileNotFoundException ex) {
                Logger.getLogger(MyFileReader.class.getName()).
                                                log(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(MyFileReader.class.getName()).
                                                log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

（15）在 Design 视图中单击拆分窗格右部的文本区域 jTextAreaText 组件，在 Properties 窗口中单击 editable 属性右侧的复选框而使之取消选择；单击 lineWrap 属性右侧的复选框而使之处于选择状态。

程序运行基本符合题目要求（见图 5.12）。这些程序在读取大容量磁盘目录及大尺寸文本文件时反应迟钝，需要专门设计工作线程以改善性能。具体改写请读者参考 3.5 节自行完成。

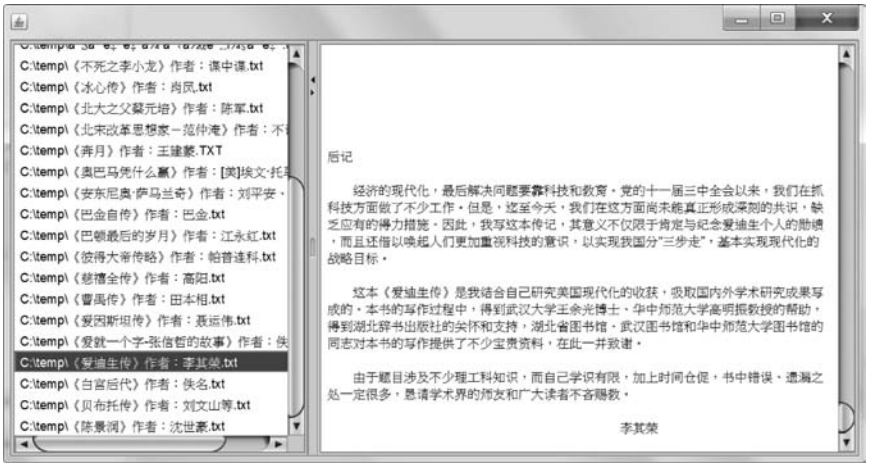


图 5.12　例 5.2 程序运行界面

5.4　标签化窗格

标签化窗格 JTabbedPane 允许用户通过单击具有给定标题和（或）图标的选项卡，在一组组件之间进行切换。很多应用程序模块由于信息量较大，需要使用能够显示多页信息的选项卡界面。例如，NetBeans IDE 中大量使用标签化窗格（见图 5.13），Windows 画图程序的功能区也是一种标签化窗格（见图 5.14）。可以说，标签化窗格是当前 GUI 不可缺少的组成元素。

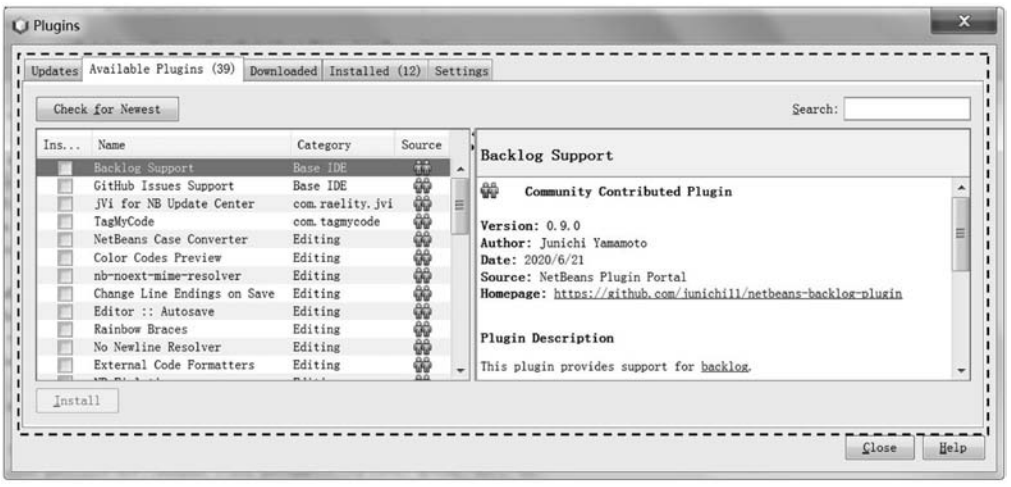


图 5.13　NetBeans 的 Plugins 标签化窗格

图 5.14　Windows 画图程序的功能区

5.4.1　标签化窗格的组成及使用

一个标签化窗格是一个容器，其中包含多个选项卡。选项卡上还有一个显示标识文字的标签(tab)。如图 5.14 所示的画图程序整个功能区是一个标签化窗格，显示的是其中一个选项卡，该选项卡的标签是“主页”。又如，图 5.13 中虚线所框的区域是一个标签化窗格，当前显示的是列出可用插件的一个选项卡，其中 Available Plugins(39)是它的标签。

使用标签化窗格时首先也需要一个顶层容器。在 chap5 项目的 book.container.demo 包中创建一个 JFrame，类名为 TabbedPaneDemo，设置窗体为边框式布局。单击 Palette→Swing Containers→Tabbed Pane 组件，然后在窗体的中央部位单击，即创建了一个标签化窗格组件 jTabbedPane1。

一个选项卡上只能容纳一个组件，而实际应用中的一个选项卡上却有多个有机联系的组件。因此，每个选项卡一般都放置一个中间型容器，在该容器中创建具体的功能组件。向标签化窗格中创建选项卡的一般操作方法是：首先单击选择标签化窗格，然后单击 Palette 上的合适组件，如 Swing Containers 中的 Pane，最后在该标签化窗格上单击，即创建一个选项卡。之后创建第二个选项卡时一定要注意鼠标所指的目标容器应该是标签化窗格，而不是上一步创建的面板。防止出错的方法是：在 Navigator 窗口的标签化窗格组件节点上右击，在快捷菜单中选择 Add From Palette 菜单下的合适二级和三级菜单项，如选择 Swing Containers→Pane 命令。如果将一个选项卡误添加到了另一个选项卡的容器中，可以在 Navigator 窗口的组件树中拖动需要调整的节点到标签化窗格节点上。例如，jPanel3 误放置到第二个选项卡的面板中(见图 5.15(a))，可以用鼠标左键按下拖动 jPanel3 到 jTabbedPane1 节点上(见图 5.15(b))，这样就将 jPanel3 调整为第三个选项卡(见图 5.15(c))。

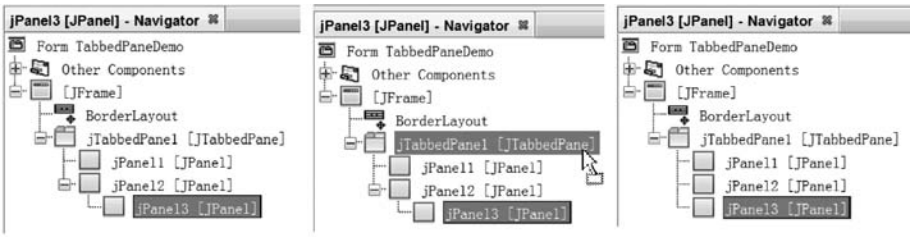


图 5.15　调整组件到选项卡中

5.4.2　属性

设计标签化窗格界面时，需要设置两类组件的属性，即标签化窗格组件的属性和其中每个选项卡组件的属性。

1. 标签化窗格的主要属性

1）tabPlacement

该属性设置各个选项卡标签的位置。默认情况下，标签位于容器的顶部，并且标签数量超过容器宽度时会自动换行形成多行。在 Properties 窗口中单击该属性行右侧的下三角按钮，列表中有 BOTTOM、RIGHT、TOP 和 LEFT 四项选择，分别用于指定标签位于标签化窗格的底部、右边、顶部和左边。

2）tabLayoutPolice

该属性设置标签化窗格布局选项卡标签的策略。属性值也是提供下拉列表选择，有两个选择项，其中，WRAP_TAB_LAYOUT 指定当选项卡较多，标签在一行排列不完时自动换行排列到下一行（见图 5.16(a)）；选择 SCROLL_TAB_LAYOUT 则指定，标签在一行排列不完时不换行，而是出现一组滚动箭头，将选项卡标签以滚动模式显示出来（见图 5.16(b)）。注意，SCROLL_TAB_LAYOUT 在 JDK 1.4 之前是不支持的。

(a) WRAP_TAB_LAYOUT布局　　(b) SCROLL_TAB_LAYOUT布局

图 5.16　tabLayoutPolice 属性值的效果

3）selectedIndex

该属性指定初始界面中所选择的选项卡索引。该属性值是一个整数。第一个选项卡索引为 0。

4）selectedComponent

该属性指定的组件所在的选项卡为初始界面中选取的选项卡。设置方法是，在 Properties 窗口中单击该属性行右侧的下三角按钮，在组件列表中选择一个组件。如果所选组件不是直接放置在标签化窗格而是它下面容器组件中的一个组件，则修改该属性的操作会失败，并有提示对话框出现。如果选项卡 tab5 中只有一个组件 jLabel2，且是标签化窗格组件的直接子节点，那么可以直接选择组件 jLabel2 为该属性值，此时 tab5 选项卡即是运行时初始界面所选取的选项卡。当指定了该属性值时，会自动更新 selectedIndex 的值。反之，当以后又修改了 selectedIndex 属性值时，selectedComponent 属性值为“默认”。

5）tabCount 和 tabRunCount

tabCount 是个只读属性，表示总选项卡的个数。

tabRunCount 也是只读属性，表示显示所有的标签所必需的行数（对于顶部或底部标签位置）或是列数（对于左边或是右边位置）。

这两个属性值设计时是自动计算的，不能修改，一般以代码方式访问。

2. 选项卡组件的属性

选项卡组件除了该组件本身原有的属性外，还在该组件的布局属性组中提供了三个设置选项卡标签的属性。

1）Tab Title

该属性设置选项卡标签上的文字。

2）Tab Icon

该属性用于指定选项卡标签上显示的图标。与其他组件 Icon 属性的设置方法相同。设置好之后，标签文字和标签图标会同时显示在选项卡标签上。

3）Tab ToolTip

该属性为选项卡标签设置一个提示框及其中的提示文字。此处设置的是提示框中的文字。当程序运行时，鼠标指到该选项卡标签上稍停留时，会出现一个黄色提示框，其中显示该属性设置的文字。

5.4.3 应用举例

例 5.3 为学生成绩管理系统设计选课界面，课程分为三类：公共基础课、专业基础课和专业课。其中，公共基础课门数较少，采用复选按钮提供选择。专业基础课较多，备选课程在一个列表中显示，选择某一门或几门课程后单击“-->”按钮添加到已选列表中，同时从备选列表中删除；同样选择某一门或几门已选课程之后，单击“<--”按钮把它们添加到备选课程列表中，同时从已选课程列表中删除。专业课分为三个方向：Java 方向、.NET 方向和嵌入式方向，每个方向都有几门课程以复选按钮形式提供给用户选择。界面原型如图 5.17 所示。

图 5.17 学生成绩管理系统设计师生选课界面

分析：三类课程公共基础课、专业基础课和专业课用三个选项卡显示，放在一个标签化窗格组件中。专业课的三个方向 Java 方向、.NET 方向和嵌入式方向也用三个选项卡显示，且该标签化窗格采用左侧标签布局。

解：设计步骤如下。

（1）展开 StdScoreMana0.3 项目中的 Source Packages 节点，右击 book.stdscoreui 包名，在快捷菜单中单击 New→JFrame Form 菜单项，类名输入“SelectCourse”，单击“完成”按钮。

（2）在 Navigator 窗口中右击 SelectCourse 窗体，在快捷菜单中单击 Set Layout→Border Layout 菜单项。

（3）单击 Palette→Swing Containers→Tabbed Pane 组件，然后在窗体的中央部位单击，即创建了一个标签化窗格组件，并重命名为 jTabbedPaneMain。

（4）单击 Palette→Swing Containers→Pane，然后在标签化窗格组件上单击，即创建了一个选项卡组件，保持默认名 jPanel1。

（5）在 Navigator 窗口的 jTabbedPaneMain 组件节点上右击，在快捷菜单中选择 Add From Palette→Swing Containers→Pane 菜单项，保持默认名 jPanel2。

（6）重复步骤（5），创建第三个选项卡面板组件，保持默认名 jPanel3。

（7）在 Navigator 窗口中单击选项卡面板 jPanel1，在 Properties 窗口的 Tab Title 属性行右侧文本框中输入“公共基础课”。

（8）使用与步骤（7）同样方法修改选项卡面板 jPanel2 的 Tab Title 属性值为“专业基础课”、选项卡面板 jPanel3 的 Tab Title 属性值为“专业课”。

（9）在 Navigator 窗口中单击选项卡面板 jPanel3，在 Properties 窗口中的 Tab Icon 属性行右侧单击“…”按钮，选择项目内已经准备好的图像文件，关闭对话框。

（10）在 Design 视图中单击选项卡标签“公共基础课”，再到界面中央部位单击选择第一个选项卡面板 jPanel1，在 Palette 中单击 Swing Controls→Check Box 组件，然后在面板 jPanel1 上的适当位置单击，创建一门课程的复选框组件 jCheckBox1。在 Properties 窗口中单击 text 属性右侧文本框，输入课程名称“大学英语”。

（11）重复步骤（10）5 次，创建另外 5 门课程的复选框，text 属性值分别为“哲学”“法律基础”“大学体育”“大学计算机”和“思想道德修养”。

（12）在 Palette 中单击 Swing Controls→Button 组件，然后在面板 jPanel1 上的适当位置单击，创建一个按钮组件，重命名为 jButtonPSelected。在 Properties 窗口中单击 text 属性右侧文本框，输入文字“选修”。

（13）重复步骤（12），按钮组件重命名为 jButtonPUnselected，text 属性值为“退选”。

（14）在 Design 视图中选择第二个选项卡面板 jPanel2，在 Palette 中单击 Swing Controls→Label 组件，然后在面板 jPanel2 上的靠近左上角位置单击。接着在 Properties 窗口中单击 text 属性右侧文本框，输入文字“备选课程”。

（15）在 Palette 中单击 Swing Controls→List 组件，然后在面板 jPanel2 上“备选课程”标签下方距左边框首选位置及距上方组件较小首选位置处单击，重命名为 jListUnselected。在 Properties 窗口中单击 model 属性右侧的“…”按钮，删除原来的列表项，输入以下列表项。

计算机组成原理
数据结构与算法
操作系统
高等数学
线性代数
离散数学
模拟电子技术
数字逻辑与数字系统
数据库系统原理
计算机网络
软件工程

单击 OK 按钮。注意，正式程序中的列表项是从数据库中读取，由代码添加到列表中的。

(16) 重复步骤(14),在与第一个标签“备选课程”基线对齐的右边稍远处创建另一个标签组件,文字修改为“已选课程”。

(17) 使用与步骤(15)相同的方法,在“已选课程”标签下方创建第二个列表组件,重命名为“jListSelected”。在 Properties 窗口中单击 model 属性右侧的“…”按钮,删除所有列表项。

(18) 在自由设计模式下调整两个列表框的宽度为 130px,高度为 170px,中间保留适当空间。

(19) 在 Palette 中单击 Swing Controls→Button 组件,然后在面板 jPanel2 中两个列表组件中间位置单击,创建按钮,重命名为“jButtonPSelect”,修改 text 属性值为“-->”。

(20) 重复步骤(19),创建按钮“jButtonPUnselect”,按钮上文字为“<--”。

(21) 重复步骤(19)两次,在靠近面板下边框稍右位置创建两个按钮,分别命名为“jButtonSave”和“jButtonClose”,按钮上文字分别为“保存”和“关闭”。

(22) 在 Design 视图中选择第三个选项卡面板 jPanel3,在 Palette 中单击 Swing Containers→Tabbed Pane 组件,然后在面板 jPanel3 上距容器左边框首选位置及距容器上边框首选位置单击,重命名为“jTabbedPaneCategory”。接着拖动该标签化窗格组件的右边框,移至距容器右边框首选位置,拖动下边框移至距容器底边框首选位置。

(23) 单击选择 jTabbedPaneCategory 组件,在 Properties 窗口中设置 tabPlacement 属性值为 LEFT。

(24) 在 Navigator 窗口中的 jTabbedPaneCategory 组件节点上右击,在快捷菜单中选择 Add From Palette→Swing Containers→Pane 菜单项,保持默认名 jPanel4。

(25) 重复步骤(24)两次,保持默认名 jPanel5 和保持默认名 jPanel6。

(26) 使用与步骤(7)相同的方法,分别为选项卡面板 jPanel4、jPanel5 和 jPanel6 设置 Tab Title 属性值为“Java 方向”“.NET 方向”和“嵌入式方向”。

(27) 使用与步骤(10)~(13)相同的方法,分别为“Java 方向”“.NET 方向”和“嵌入式方向”三个选项卡创建选课复选框及按钮。设计完成后,这三个选项卡的 Design 视图如图 5.18 所示。

(a) Java方向选项卡　(b) .NET方向选项卡　(c) 嵌入式方向选项卡

图 5.18　专业课 3 个方向选课选项卡

对于组件比较多的复杂界面,应该认真检查 Navigator 窗口中的组件节点树,审查节点组件之间的关系是否正确,命名是否合理,有没有冲突等。本例题完成后的节点树见图 5.19。该模块的功能实现需要用到列表模型等知识,留待后续相应章节介绍。

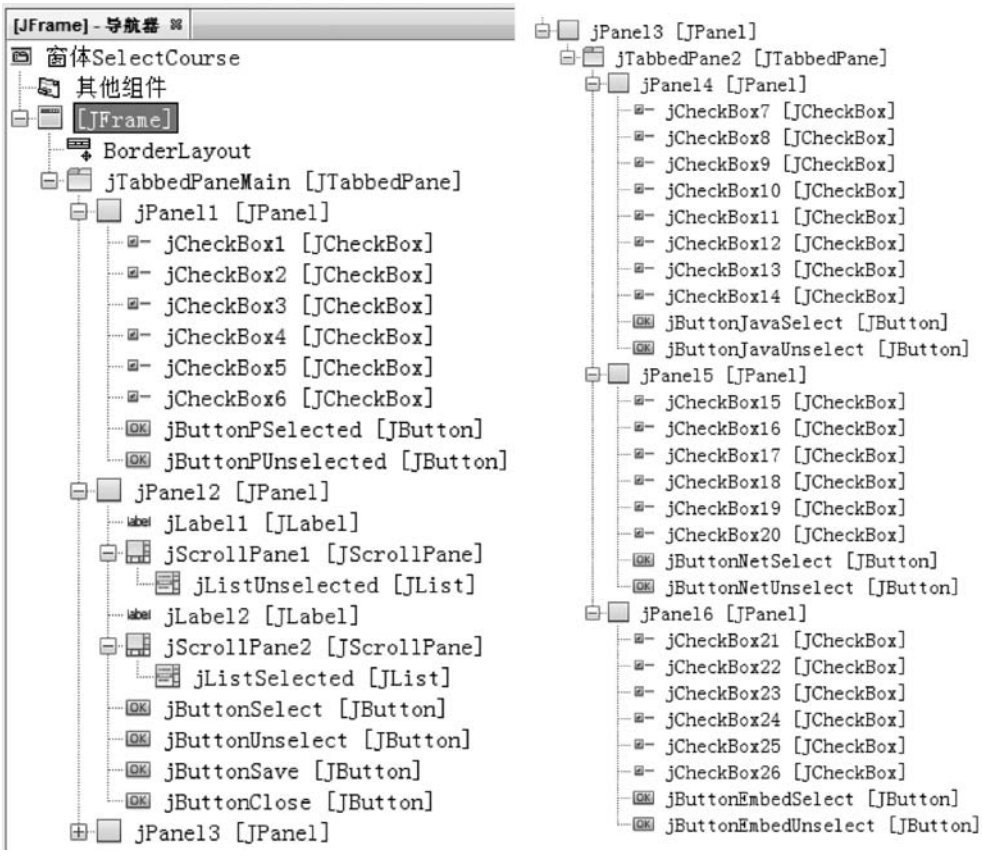


图 5.19　例 5.3 师生选课界面节点树

5.5　Swing 面板层次与分层窗格

正如 2.2 节中的图 2.5 所示，Swing 的高层容器有复杂的层次结构。事实上，Swing 不能将组件直接添加到高层容器中，而只能将这些组件添加到根面板的一部分，然后由根面板来管理这些组件。分层窗格是根面板的主要组成部分，也是根面板的主要组件容器。

5.5.1　Swing 面板层次

Swing 中的四个顶级容器 JFrame、JDialog、JWindow 和 JApplet 以及轻量级非顶级容器 JInternalFrame 都实现了 RootPaneContainer 接口，并且它们都将其操作委托给根面板 JRootPane（见图 5.20）。

在根面板 JRootPane 中只有两个组件：一个分层窗格 JLayeredPane 以及一个玻璃面板 Glass Pane(Component)（见图 5.21）。前面的玻璃面板位于所有窗格之上，可以是任意组件，而且是不可见的，能够截取鼠标移动，保证类似工具提示文本这样的元素显示在其他的 Swing 组件之前。后面是分层窗格 JLayeredPane。分层窗格也由两部分组成：在其上部包含一个可选的 JMenuBar，在其下部的另一层中包含一个内容面板 content Pane。如果已在根面板上设置了 JMenuBar 组件，它将沿窗体的上边缘放置，content Pane 的位置和大小

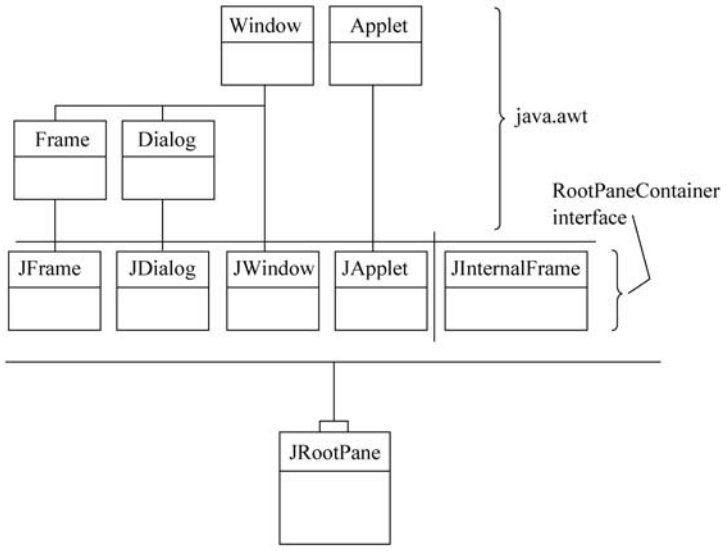


图 5.20　使用根窗格的各个类之间的关系

将进行调整以填充剩余的区域。但如果没有使用 JMenuBar，则 content Pane 就会占据整个版面。通常将组件添加在根面板 JRootPane 中，实际上就是添加在内容面板中。

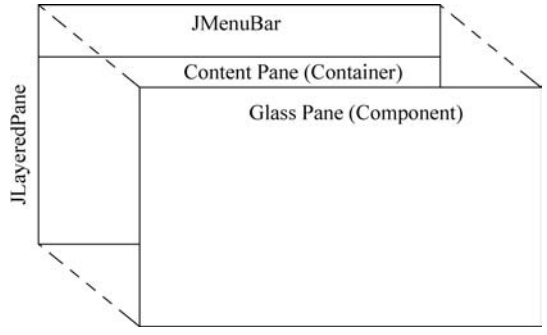


图 5.21　根面板的组成

5.5.2　分层窗格的使用

分层窗格 JLayeredPane 是一种 Swing 容器，提供了管理其内部组件的第三维：深度（也称 Z 顺序或层）。这可以保证在某些情况下，例如创建工具提示文本、弹出菜单与拖曳时，特定的组件可以创建在其他的组件之上。

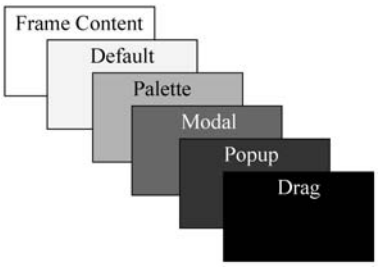


图 5.22　分层窗格预定义层

1. 在层中添加组件

每向分层窗格中添加一个组件时，使用一个整数设置组件的 Z 顺序。层设置越高，则组件绘制离顶层组件就越近。分层窗格预定义了六个层（见图 5.22）及层常量（见表 5.2）。

表 5.2　JLayeredPane 层常量

选　　项	描　　述
FRAME_CONTEND_LAYER	取值－30 000，用于存储菜单栏及内容面板；通常并不为开发者所用
DEFAULT_LAYER	取值 0，用于通常的组件层
PALETTE_LAYER	取值 100，用于存储浮动工具栏以及类似的组件
MODAL_LAYER	取值 200，用于存储显示在默认层，调色板之上以及弹出菜单之下的弹出对话框
POPUP_LAYER	取值 300，用于存储弹出菜单以及工具提示文本
DRAG_LAYER	取值 400，用于存储保持在顶部的拖动对象

尽管可以为层次使用自己的常量，但是使用时要小心，因为系统会在需要时使用预定义的常量。如果设置的常量不正确，组件就不会如希望的那样工作。

2. 使用内容层与位置

分层窗格中的组件同时具有层与位置。当某一层只有一个组件时，其位于位置零。当在相同的层有多个组件时，后添加的组件具有更高的位置数字。位置设置越低，显示距离顶部组件越近。位置－1 自动位于具有最高位置的底层。

3. 属性

分层窗格组件本身没有什么特有的属性。添加到分层窗格内的组件有 Layer 属性设置组件在分层窗格中的 Z 顺序。在 Properties 窗口中单击 Layer 属性右侧的下三角按钮，有 DEFAULT_LAYER、PALETTE_LAYER、MODAL_LAYER、POPUP_LAYER 和 DRAG_LAYER 五个选项，其实这五个选项就是表 5.2 中列出的相应层常量。

5.5.3　应用举例

例 5.4　程序窗口中有 5 个部分重叠的不同颜色的方块，从底层向顶层依次是黄色、洋红色、蓝绿色、红色和绿色方块。窗口中还有一个随鼠标指针而移动的 duke 图标。duke 初始与蓝绿色方块位于同一层，位置与蓝绿色、红色和绿色方块重叠时被遮挡而不能显示或不能完全显示。但用户通过单击窗口下部的单选按钮可改变 duke 的层次，例如，用户单击 green 单选按钮后，它与绿色方块处于同一层而位于其他 4 层的上边，只有移动到与绿色方块重叠时才被遮挡（见图 5.23）。使用 NetBeans IDE 的 GUI 设计器，可视化设计该程序。

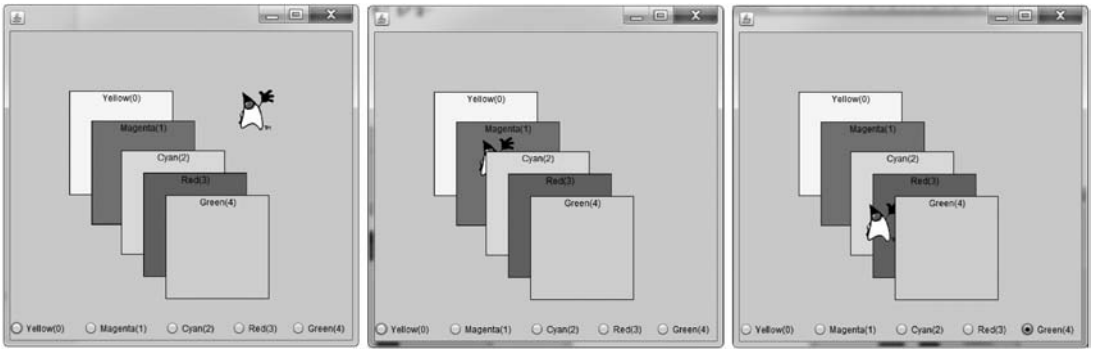


图 5.23　例 5.4 程序的运行界面部分快照

分析：5 个方块及 duke 图标有层次布局，应该放置于分层窗格中，并通过它们的 Layer 属性设置布局层次。每个方块及 duke 图标都可以使用标签组件实现，给标签设置相应背景颜色并设置为不透明。五个单选按钮设置为一个按钮组，并为它们注册和设计 Action 事件监听器来改变 duke 标签的 Layer 属性值。

解：设计步骤如下。

(1) 在 chap5 项目的 book. container. demo 包中新建一个 JFrame Form，类名为 LayeredPaneDemo。设置该窗体为边框式布局。

(2) 单击 Palette→Swing Containers→Layered Pane 组件，然后在窗体的中央部位单击，即创建了一个分层窗格组件，使用默认组件名 jLayeredPane1。

(3) 分层窗格组件 jLayeredPane1 默认使用空值布局，本例设置为绝对布局。

(4) 在 Palette 中单击 Swing Controls→Label 组件，然后在 Design 视图的分层窗格组件 jLayeredPane1 上的适当位置(靠近左上角)单击，创建一个标签组件，并重命名为“jLabelYellow”。

(5) 在 Properties 窗口中单击 background 属性行右侧的“…”按钮，在 background 对话框中选择 Color Chooser → AWT Palette → Yellow 选项，单击 OK 按钮；单击 horizontalAlignment 属性行右侧下三角按钮，选择 CENTER；单击 verticalAlignment 属性行右侧下三角按钮，选择 TOP；在 text 属性行右侧输入“Yellow(0)”；修改首选尺寸、最大尺寸和最小尺寸都为[140, 140]；单击 opaque 属性行右侧复选框(选取)；单击 Layer 属性行右侧下三角按钮，选择 DEFAULT_LAYER。

(6) 重复步骤(4)和(5)，按表 5.3 修改变量名，设置属性值。horizontalAlignment、verticalAlignment、opaque、首选尺寸、最大尺寸和最小尺寸属性都与步骤(5)取相同值。并移动各标签组件位置，使它们按照如图 5.23 所示位置重叠。

表 5.3　例 5.4 中各色块标签属性

默认组件名	重命名组件名	background	text	层	X	Y
jLabel2	jLabelMagenta	洋红色	Magenta(1)	PALETTE_LAYER	110	120
jLabel3	jLabelCyan	青色	Cyan(2)	MODAL _LAYER	150	160
jLabel4	jLabelRed	红色	Red(3)	POPUP _LAYER	180	190
jLabel5	jLabelGreen	绿色	Green(4)	DRAG _LAYER	210	220

(7) 在 Palette 中单击 Swing Controls→Label 组件，然后在 Design 视图的分层窗格组件 jLayeredPane1 上的适当位置(靠近右上角)单击，创建一个标签组件，并重命名为 jLabelDuke。

(8) 准备好 duke 图标的图像文件，复制到本窗体类所在的包中。单击 icon 属性行右侧的“…”按钮，在 icon 对话框中选择图像选择器的项目内图像，指定图标。单击 Layer 属性行右侧下拉箭头，选择 MODAL _LAYER。清除 text 属性右侧的文字。

(9) 在 Palette 中单击 Swing Controls→Button Group 组件，然后在 Design 视图的分层窗格组件 jLayeredPane1 上的左下角适当位置单击，创建一个按钮组组件，使用默认名 buttonGroup1。

(10) 在 Palette 中单击 Swing Controls→Radio Button 组件，然后在 Design 视图的分

层窗格组件 jLayeredPane1 上的下部适当位置单击，创建一个单选按钮组件，重命名为 jRadioButtonYellow。

（11）在 Properties 窗口的 text 属性行右侧输入“Yellow(0);”单击 buttonGroup 属性行右侧下三角按钮，选择 buttonGroup1。

（12）在 Design 视图中右击该单选按钮组件，在快捷菜单中选择 Events→Action→actionPerformed 菜单项，在 Source 视图中该组件的事件处理方法(jRadioButtonYellowActionPerformed)中输入语句：

```
jLayeredPane1.setLayer(jLabelDuke, JLayeredPane.DEFAULT_LAYER);
```

（13）重复步骤(10)～步骤(12)，按照表 5.4 修改组件名、text 属性值及输入事件处理语句。

（14）在 Design 视图中右击窗体 LayeredPaneDemo，在快捷菜单中选择 Events→MouseMotion→mouseMoved 菜单项，在事件处理方法 jLayeredPane1MouseMoved()中添加语句“jLabelDuke.setLocation(evt.getX()－50, evt.getY()－50);”。

表 5.4　例 5.4 中各单选按钮的设置

默认组件名	重命名组件名	text	事件处理方法中输入的语句
jRadioButton2	jRadioButtonMagenta	Magenta(1)	jLayeredPane1.setLayer(jLabelDuke, JLayeredPane.PALETTE _LAYER);
jRadioButton3	jRadioButtonCyan	Cyan(2)	jLayeredPane1.setLayer(jLabelDuke, JLayeredPane. MODAL _LAYER);
jRadioButton4	jRadioButtonRed	Red(3)	jLayeredPane1.setLayer(jLabelDuke, JLayeredPane. POPUP _LAYER);
jRadioButton5	jRadioButtonGreen	Green(4)	jLayeredPane1.setLayer(jLabelDuke, JLayeredPane. DRAG _LAYER);

5.6　桌面窗格与内部框架

有些 GUI 应用程序将信息在多个窗口中显示出来，并且把这些窗口都包含在一个大的窗口之中。当应用程序窗口最小化时，其中包含的所有子窗口都隐藏起来，关闭应用程序窗口，则这些子窗口都被关闭。在 Windows 环境，这种界面称为多文档界面(Multiple Document Interface，MDI)。这种程序的 GUI 就好像一个传统的视窗系统，其中有一个桌面，这个桌面上放置了多个浮动的窗口。Swing 中使用桌面窗格组件创建这种桌面，使用内部框架创建这些浮动窗口。

5.6.1　桌面窗格的使用

桌面窗格 JDesktopPane 是用于创建多文档界面或虚拟桌面的容器。桌面窗格是特殊的分层窗格，管理可能的重叠内部窗体。

桌面窗格是中间容器，使用时需要把它添加到顶级容器或顶级容器所包含的容器中。

例如，在项目 chap5 的 book. container. demo 包中新建一个 JFrame 窗体，命名为 DesktopPaneDemo，并设置为边框式布局。在 Palette 上单击 Swing Containers→Desktop Pane 组件，鼠标移到窗体中央单击，即可创建一个桌面窗格组件 jDesktopPane1。

桌面窗格本身默认是空值布局，灰色([171,171,171])背景和黑色([0,0,0])前景颜色，无边框。有一个 DesktopManager 接口实现类实例的引用，委托该实例管理桌面窗格内部框架。默认设置时，在 Design 视图中桌面窗格是一个灰色的矩形，单击选择后有橙色边框。运行该程序，窗口中的桌面窗格有一个蓝色带图案背景(见图 5.24，但可以通过主题及 L&F 改变)。

图 5.24　设计视图下的桌面窗格(左)与运行时的界面(右)

桌面窗格有以下几个重要属性。

1. dragMode

该属性设置桌面窗格的内部窗口拖曳时的界面更新绘制模式。一般地，在拖曳窗口过程中，桌面管理器会不断要求窗体重新绘制，这样会导致界面更新绘制速度和程序响应速度非常缓慢。为了提高性能，可以设置“边框拖曳”，即当用户拖动窗口时，只有窗口的边框是连续更新的，而窗口的内容只有当拖到最终停止位置时才刷新。但在视频硬件支持拖动操作时，在拖动过程中可以将窗口图像映射到屏幕别的位置，从而界面刷新速度也很快，同时界面观感更好。在桌面窗格的 Properties 窗口中，单击 dragMode 属性行右侧的下三角按钮，选择 OUTLINE_DRAG_MODE 即设置边框拖曳模式，而选择 LIVE_ DRAG_MODE 则设置为采用实况拖曳更新模式。

2. selectedFrame

该属性设置桌面窗格中当前活动的内部窗体。在桌面窗格的 Properties 窗口中，单击 selectedFrame 属性行右侧的下三角按钮设置。

3. allFrames

这是一个只读属性，返回桌面窗格中的所有内部窗体。

5.6.2　内部框架

内部框架 JInternalFrame 是一个轻量级的高层窗口，且有一个根面板，许多方面都很像 JFrame，但它并不是一个顶层窗口。内部框架一般放在桌面窗格中用以构建多文档界面。

要创建一个内部框架，一般先需要创建一个桌面窗格，然后在该桌面窗格中创建内部框架。例如，前面已经创建了桌面窗格 jDesktopPanel，接下来在组件面板的 Swing Containers 组中单击 Internal Frame 组件图标，然后将鼠标光标移到桌面窗格中单击，即创建了一个内部框架 jInternalFrame1，且自动采用边框式布局。初建的内部框架组件很小，它的父容器桌面窗格默认采用空值布局，但 IDE 采用自由设计模式，可以直接拖动边框改变该内部框架的大小。

另一种创建内部框架的方法是，在 IDE 的主菜单中选择 New→New File 菜单项，在 New File 对话框中选择 Swing GUI Forms 类别，在文件类型列表中选 JInternalFrame Form，单击 Next 按钮，输入类名，单击 Finish 按钮即可创建一个独立的内部框架。当设计完成该内部框架后，从 Projects 窗口中将该窗体文件拖放到其他容器，即可使该内部框架作为接收容器的一个内部框架组件使用。也可以在包名节点上右击，选择 New→JInternalFrame Forms，输入类名，单击 OK 按钮即可，这样更为简单。

如果桌面窗格采用自由设计模式，初始创建的内部框架运行时没有标题、最小化、最大化/还原和关闭按钮等修饰部件，不能调整大小，也不能移动。当设置桌面窗格采用空值布局时，如果不设置 JFrame 窗体大小，程序运行时窗口内部高度极小，看不到内部框架，放大窗口后内部窗口却可以自由移动，但不能调整大小。内部框架窗口有控制菜单（见图 5.25）。

图 5.25　内部框架的设计和运行界面

内部框架许多属性可以定制其外观和行为。

1. defaultCloseOperation

该属性设置当用户关闭该内部框架时的行为。取值为 DISPOSE 则销毁该内部窗格，取值为 HIDE 则只是隐藏该内部窗格，取值为 DO_NOTHING 则无响应动作。单击 Properties 窗口中该属性行右侧的下三角按钮，选择其中一种。

2. title

title 属性设置该内部窗格的标题栏文字。单击 Properties 窗口中该属性行右侧的文本框，输入标题文字即可。

3. closable

该属性设置是否在标题栏显示“关闭”按钮。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，则标题栏显示“关闭”按钮，否则标题栏不显示“关闭”按钮。程序运行时，用户单击内部框架窗口“关闭”按钮的效果取决于 defaultCloseOperation 属性的设置。

4. iconifiable

该属性设置是否在标题栏显示“最小化”按钮。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，则标题栏显示“最小化”按钮，否则标题栏不显示“最小化”按钮。程序运行时，用户单击内部框架窗口“最小化”按钮，则该内部窗格缩小为该窗口标题栏，显示在桌面窗格的靠近底边框行，“最小化”按钮变为“还原”按钮；再次单击则还原为最小化前的状态。

5. maximizable

该属性设置是否在标题栏显示“最大化”按钮。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，则标题栏显示“最大化”按钮，否则标题栏不显示“最大化”按钮。程序运行时，用户单击内部框架窗口“最大化”按钮，则该内部窗格扩大占据整个桌面窗格空间，“最大化”按钮变为“还原”按钮；再次单击则还原为最大化前的状态。

6. resizable

该属性设置是否可以通过拖动窗口的调整控柄改变内部框架窗口大小。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，则可以改变该内部框架窗口大小，否则内部框架窗口大小不可通过拖动边框改变。

7. frameIcon

该属性设置在标题栏左端显示的控制菜单图标。单击 Properties 窗口中该属性行右侧的“…”按钮，在对话框的图像选择器中选取一个合适图像文件；如果项目中曾经导入过图标，则可以单击下拉箭头直接在列表中选择。

8. normalBounds

该属性设置该内部框架初始显示的位置和大小。单击 Properties 窗口中该属性行右侧的“…”按钮，在对话框中输入 X、Y、Width 和 Height 值。

9. selected

selected 属性指定该内部框架是否是程序运行时初始界面中选取的内部框架窗口。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，则程序运行时该内部框架窗口处于前台，一般它的标题栏以蓝色显示，否则处于后台且标题栏以灰色显示。

10. visible

visible 属性指定该内部框架在程序运行时初始界面中是否可见。单击 Properties 窗口中该属性行右侧的复选框，如果该复选框被选取，该内部框架在 Design 视图和程序运行窗口中显示出来，否则在桌面窗格中看不见该内部窗格，但在 Navigator 窗口中可以找到该内部窗格的对应节点。还应注意，在设计时修改了该属性，可能在界面上看不到该内部窗格组件，因为此时把它的尺寸设为 0×0，需要重新设置大小。

11. layer 及 Layer

内部窗格所在的桌面窗格是一个分层窗格，所以 Layout 组的 Layer 属性以表 5.2 所列的预定义常量指定该内部窗格所处的层。Other Properties 组的 layer 属性则用一个整数值(int)指定该内部窗格所处的层。这两个属性发生冲突时，Other Properties 组的 layer 属性值优先。

12. maximum

该属性设置该内部窗格在程序运行的初始界面中是否以最大化方式显示。但只有在它

所在的桌面窗格采用空值布局时才有确定效果。

13. icon

此属性设置该内部窗格在程序运行的初始界面中是否以最小化方式(图标)显示。但只有在它所在的桌面窗格采用空值布局时才有确定效果。

5.6.3 多文档界面的设计方法

使用 Java Swing 类库和 NetBeans IDE 设计一个多文档界面时，需要处理一些较为复杂的问题。例如，需要处理内部框架窗口的级联与平铺布局，处理属性设置的否决问题，拖曳操作，以及内部框架中的对话框使用等。

1. 设计多文档界面的一般步骤

(1) 设计应用程序中的常规 JFrame 窗体。

(2) 在 JFrame 中添加和设计桌面窗格。

(3) 构建和设计若干个内部框架。

(4) 确定和调整内部框架的大小。

(5) 设计和设置内部框架的显示属性。

(6) 向内部框架中添加所需要的组件。

(7) 将内部框架添加到桌面窗格中。

(8) 确定和设置默认选定的内部框架。

(9) 调整各内部框架的位置，使它们互相有合适的距离。一般内部框架之间的合适距离是标题栏的高度，计算方法是：

```
int frameDistance = iframe.getHeight() - iframe.geContentPane().getHeight();
```

其中，iframe 是内部框架组件名。

(10) 重新定位各内部框架的位置。计算方法是：

```
nextFrameX += frameDistance;
nextFrameY += frameDistance;
if(nextFrameX + width > desktop.getWidth())
    nextFrameX = 0;
if(nextFrameY + height > desktop.getHeight())
    nextFrameY = 0;
```

其中，width 和 height 是当前内部框架的宽度和高度，desktop 是内部框架所在的桌面窗格组件名。

2. 级联窗口

Windows 环境有用于窗口级联与平铺的标准命令，但是 Swing 的桌面窗格和内部框架却没有提供任何有关级联和平铺窗口的内部支持，需要编写程序解决这个问题。

为了级联所有窗口，应该将它们绘制成相同大小，并交错排列位置。首先获取桌面窗格中的所有内部框架：

```
JInternalFrame[] frame = jDesktopPane1.getAllFrames();
```

还应注意排除已经最小化的内部框架，将最大化的内部框架设置为非最大化可缩放状态。

为级联一个桌面窗格中内部框架设计下列方法，其中，参数 jDesktopPane 是内部框架所加入的桌面窗格、jFrame 是桌面窗格 jDesktopPane 的父容器（一般是一个 JFrame）。该方法代码如下。

```
public void cascadeInerWindows(JFrame jFrame, JDesktopPane jDesktopPane) {
    jFrame.setExtendedState(JFrame.MAXIMIZED_BOTH);
    jFrame.validate();
    JInternalFrame[] frame = jDesktopPane.getAllFrames();
    if(frame == null || frame.length == 0)      //如果桌面窗格中不包含内部框架则直接返回
        return;
    int s = 0;                                  //内部框架的左边框和顶边框的间距
    int x = 10, y = 10 ;                        //内部框架的左上角坐标
    int w = 0, h = 0 ;                          //内部框架的宽度和高度
    for (int i = frame.length - 1; i >= 0; i--) {
        if (!frame[i].isIcon()) {
           if(s == 0) {//找到第一个非最小化的内部框架，初始化间距、宽度和高度
                s = frame[0].getHeight() - frame[0].getContentPane().getHeight();
                w = frame[0].getWidth();
                h = frame[0].getHeight();
           }
           try {
                frame[i].setMaximum(false);
                frame[i].reshape(x, y, w, h);
                x += s;
                y += s;
                if (x + w > jDesktopPane.getWidth()) {
                    x = 0;
                }
                if (y + h > jDesktopPane.getHeight()) {
                    y = 0;
                }
            } catch (PropertyVetoException e) {
                e.printStackTrace();
            }
        }
    }
}
```

3. 平铺窗口

窗口的平铺比级联更复杂一些。首先，计算出非最小化的内部框架数目，然后计算桌面窗格中平铺内部框架时所需行数及每行列数。循环中先在各行第一列平铺内部框架，一列排满后排列下一列。当排满不满员的一行后，重新计算窗格高度，以便最大限度地利用桌面窗格高度。该方法代码如下。

```
public void tileInerWindows(JFrame jFrame, JDesktopPane jDesktopPane) {
    jFrame.setExtendedState(JFrame.MAXIMIZED_BOTH);
    jFrame.validate();
    JInternalFrame[] frame = jDesktopPane.getAllFrames();
```

```
        if(frame == null || frame.length == 0)
            return;
        int frameCount = frame.length;                    //桌面窗格包含的内部框架个数
        int rows = (int) Math.sqrt(frameCount);           //平铺窗格所需要的行数
        int cols = frameCount / rows;                     //平铺窗格每行的列数
        int extra = frameCount % rows;                    //最后一行包含的窗格数
        int r = 0, c = 0;                                 //行号与列号
        int w = jDesktopPane.getWidth() / cols;           //桌面窗格(布局空间)宽度
        int h = jDesktopPane.getHeight() / rows;          //桌面窗格(布局空间)高度
        for (JInternalFrame fr : frame) {
            if (!fr.isIcon()) {
                try {
                    fr.setMaximum(false);
                    fr.reshape(c * w, r * h, w, h);
                    r++;
                    if (r == rows) {
                        r = 0;
                        c++;
                        if (c == cols - extra) {
                            rows++;
                            h = jDesktopPane.getHeight() / rows;
                        }
                    }
                } catch (PropertyVetoException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

4. 否决属性设置

在具有多文档界面的程序中，有时需要程序监视用户对内部框架窗口的一些属性改变的操作，以便防止用户做了程序所不希望的操作。如用户对窗口中的数据做了修改，还没有存盘，但是用户单击了窗口的“关闭”按钮，此时后台还有一些计算没有完成，即不能立即存盘(数据不完整)，那么就需要否决用户关闭窗口的操作。可以通过给该属性设计并注册可否决的更改监听器(VetoableChangeListener)而否决属性改变。

通过以下几个步骤可以实现这样一个**监视并通知**机制。

(1) 为每个内部框架注册一个 VetoableChangeEvent 事件监听器。

(2) 实现 VetoableChangeListener 接口的 vetoableChange()方法。使用该方法的参数(VetoableChangeEvent evt)evt 的 getName()方法查找用户想要更改的属性名称，还可以调用 getNewValue()方法获取用户提供的该属性新值。

(3) 可以通过抛出一个 PropertyVetoException 异常阻止该属性的更改。当然也可以给出具体原因及建议。

5. 内部框架中的对话框

有时在内部框架中可能需要使用对话框与用户交互。但是，不能使用 JDialog 类型的

对话框，因为 JDialog 对话框会在视窗系统中创建一个新窗口，并且不知道它与内部框架窗口之间的相对位置。相应地，应该是 JOptionPane 类型的 showIntenalXxxDialog 方法创建一个轻型对话框。如果要求复杂，则可以用 JInternalFrame 来自己设计。

5.6.4 应用举例

例 5.5 修改例 5.2 设计的文本阅读器，每当在左窗格选择一个文本文件时，就在右边窗格显示这个文件内容。即，使右边窗格能够同时显示多个文本文件内容。

分析：要使拆分窗格的右边窗格同时显示多个文件内容，可以将原例 5.2 中右边窗格的组件替换为桌面窗格，然后在这个桌面窗格中添加内部框架。每新打开一个文本文件就添加一个内部窗格。这可以在左边窗格中的 ListSelectionEvent 事件监听器中动态添加。为此，需要先创建一个独立的内部框架，并在其中添加文本区域，并设置有关属性，然后在需要时创建该内部框架的实例，并添加到桌面窗格中。

解：按照以下步骤设计。

(1) 在 Projects 窗口中右击 TextFileReader0.1 项目名称，在快捷菜单中单击 Copy 菜单项，在 Copy Project 对话框中修改项目名称为 TextFileReader0.2，单击 Copy 按钮。

(2) 打开 TextFileReader0.2 项目中的 MyFileReader.java 文件，在 Navigator 窗口中右击 jSplitPanefr 节点下的 jScrollPane2 节点，在快捷菜单中单击 Delete 菜单项。

(3) 在 Palette 中单击 Swing Containers→Desktop Pane 组件，然后在 Design 视图的拆分窗格右面的“右键”占位符按钮上单击，创建桌面窗格组件 jDesktopPane1。

(4) 右击 jDesktopPane1 组件，在快捷菜单中单击 Set layout→Null Layout 菜单项。

(5) 在 Projects 窗口中右击该项目 Source Packages→book.filereader 节点，在快捷菜单中单击 New→Other→Swing GUI Forms→JInternalFrame Form 命令，输入类名“InternalFrameText”，单击 Finish 按钮。即创建了一个独立的内部框架 InternalFrameText 窗体。

(6) 设置 InternalFrameText 内部框架的 closable、iconifiable、maximizable、resizable、selected 和 visible 为选取状态。

(7) 在 Navigator 窗口中右击 JInternalFrame 节点，在快捷菜单中单击 Set Layout→Border Layout 菜单项。

(8) 在 Palette 中单击 Swing Controls→Text Area 组件，然后在 Design 视图内部框架 InternalFrameText 窗体的中央单击，创建 jTextArea1 组件。

(9) 设置 jTextArea1 组件的 editable 属性为非选取状态，lineWrap 属性为选取状态。

(10) 切换到 InternalFrameText 窗体的 Source 视图，在类体中添加字段变量“String fileName=null;”并为 fileName 添加 getter()和 setter()代码。为 jTextArea1 添加 getter()方法。

(11) 切换到 MyFileReader 窗体的 Source 视图，在类体中添加以下字段变量。

```
private InternalFrameText internalFrameText1;
private javax.swing.JTextArea jTextAreaText;
```

在构造方法中最后一行语句后面添加代码：

```
internalFrameText1 = new InternalFrameText();
this.jTextAreaText = this.internalFrameText1.getjTextArea1();
```


（12）在 MyFileReader 类体中编写向桌面窗格组件 jDesktopPane1 中添加内部框架窗口的方法，方法代码如下。

```
void newIFT(String fileName) {
    boolean hasFrame = false;
    for (JInternalFrame frame : jDesktopPane1.getAllFrames()) {
        if (fileName.equals(((InternalFrameText) frame).getFileName())) {
            internalFrameText1 = (InternalFrameText) frame;
            hasFrame = true;
            break;
        }
    }
  int s = internalFrameText1.getHeight() - internalFrameText1.getContentPane().getHeight();
    int x = internalFrameText1.getX() + s;
    int y = internalFrameText1.getY() + s;
    int w = internalFrameText1.getWidth();
    int h = internalFrameText1.getHeight();
    if (!hasFrame) {
        internalFrameText1 = new InternalFrameText();
        internalFrameText1.setFileName(fileName);
        jTextAreaText = internalFrameText1.getjTextArea1();
        jDesktopPane1.add(internalFrameText1);
        internalFrameText1.setBounds(x, y, (int)(jDesktopPane1.getSize().getWidth() * 0.7),
                                           (int)(jDesktopPane1.getHeight() * 0.7));
        internalFrameText1.setTitle(fileName);
    }
    try {
        internalFrameText1.setSelected(true);
    } catch (PropertyVetoException ex) {
        Logger.getLogger(MyFileReader.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

（13）修改文件列表 jListFile 的 ListSelectionEvent 事件处理方法，在检测到当前选择的列表项是文本文件时，在该代码块开头调用 newIFT 方法。

```
else if (file != null && (file.getAbsolutePath().endsWith(".txt") ||
                                        file.getAbsolutePath().endsWith(".TXT"))) {
    newIFT(file.getAbsolutePath());
    … //例 5.2 中该部分程序代码不变
}
```

完成上述设计后运行程序，基本符合要求（见图 5.26）。

图 5.26　例 5.5 设计的文本阅读器程序运行界面

5.7　工　具　栏

工具栏 JToolBar 是一个中间容器，可以添加按钮、组合框等组件，把常用的命令放在可以迅速发现的位置，并把它们以常用命令组的形式组合在一起。许多情况下，工具栏按钮在菜单栏中会有对应的命令。Swing 的工具栏组件具有“浮动”的能力，即也可以成为父窗口顶部独立的子窗口，在父容器采用边框式布局时，还可以拖动到四个边框旁边。

5.7.1　使用方法

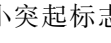
要使用工具栏，应该先有一个父容器，在 Palette 中单击 SwingContainers→Tool Bar 组件，在 Design 视图的父容器上单击，即可创建一个空的工具栏组件。在设置和调整好工具栏大小之后，可以向工具栏中添加按钮、组合框等组件形成工具栏上的工具按钮。放置到工具栏上的按钮外观与其他位置按钮有较明显的区别。

工具栏有以下重要属性。

1. orientation

该属性设置工具栏在父容器中的排列方向。在 Properties 窗口中单击该属性行右侧下三角按钮，下拉列表中有 HORIZONTAL 和 VETICAL 两个选项，分别设置工具栏水平或垂直排列。

2. floatable

该属性设置工具栏在父容器中可否浮动。在 Properties 窗口中单击该属性行右侧复选框，当该属性处于选取状态时，用户可以拖动工具栏到其他位置，也可以拖出原位置形成一个单独小窗口，且它的左端有一个小突起标志（  ）；否则，该工具栏固定于设置所定位之处，左端没有特殊标志（）。

3. toolTipText

该属性设置工具栏的工具提示文字。

通常，工具栏的工具按钮上显示图标、文字或二者都有。此外，通常也为工具按钮设置工具提示框，将鼠标停留在工具按钮上面的时候弹出来一个小“泡泡”（黄色方块，框内显示一些文字）。工具提示在应用程序中可能非常有用，可以为难用的项目提供帮助、扩展信息，甚至在拥挤的 UI 中显示某个项目的完整文本。事实上，Swing 的多数组件都可以设置 toolTipText 属性，并通过把鼠标放在组件上的特定时间来触发它；通常在鼠标处于不活动状态大约 1s 之后显示。只要鼠标还停留在那个组件上，它们就保持可见。

工具栏上的按钮等组件没有增加特殊属性。

5.7.2 应用举例

例 5.6 为例 5.5 设计的文本阅读器添加工具栏。工具栏中提供对打开的文档窗口层叠、平铺、全部关闭和退出按钮，并实现这些功能。

解：按照以下步骤设计。

（1）打开 TextFileReader0.2 项目的 MyfileReader 窗体，在 Navigator 窗口中右击 JFrame 节点，在快捷菜单中选择 Add From Palette→Swing Containers→Tool Bar 命令。

（2）在 Navigator 窗口中单击 jToolBar1 节点，在 Properties 窗口中修改首选大小的宽度为 100px、高度为 25px，最大尺寸和最小尺寸的宽度为 190px、高度为 27px。其他属性默认。

（3）在 Palette 中单击 Swing Controls→Button 组件，然后在 Design 视图中工具栏的中央单击，创建 jButton1 组件，重命名为 jButtonCascade。修改 text 属性为“层叠”。

（4）重复步骤（3）三次，分别创建三个按钮。组件名和 text 属性值分别为“jButtonTile”——“平铺”、“jButtonAllClose”——“全部关闭”及“jButtonExit”——“退出”。

（5）右击组件 jButtonCascade，在快捷菜单中选择 Events→Action→action Performed，在 Source 视图中编写窗口层叠事件处理方法。使用 5.6.3 节设计的方法，代码如下。

```
private void jButtonCascadeActionPerformed(java.awt.event.ActionEvent evt) {
    cascadeInerWindows(this, jDesktopPane1);
}
```

（6）右击组件 jButtonTile，在快捷菜单中选择 Events→Action→action Performed 菜单项，在 Source 视图中编写窗口平铺事件处理方法。使用 5.6.3 节设计的方法，代码如下。

```
private void jButtonTileActionPerformed(java.awt.event.ActionEvent evt) {
    tileInerWindows(this, jDesktopPane1) ;
}
```

(7) 右击组件 jButtonAllClose,在快捷菜单中选择 Events→Action→action Performed 菜单项,在 Source 视图中编写全部关闭窗口事件处理方法。代码如下。

```
private void jButtonAllCloseActionPerformed(java.awt.event.ActionEvent evt) {
    for (JInternalFrame frame : jDesktopPane1.getAllFrames()) {
        frame.dispose();
    }
}
```

(8) 右击组件 jButtonExit,在快捷菜单中选择 Events→Action→action Performed 菜单项,在 Source 视图中编写“退出”按钮事件处理方法。代码如下。

```
private void jButtonExitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

完成上述步骤后运行程序,满足设计要求。当然可以进一步改进,如对每个工具按钮设置图标、设置工具提示等,请读者自行完善。

习　　题

1. 什么是容器? 容器与 Swing Controls 组的组件是什么关系?

2. 试述滚动窗格的内部组成。

3. 如何给文本区域添加滚动条?

4. 如何设计一个左边和右边始终按 3∶7 比例显示的拆分窗格?

5. 一个选项卡是 Palette 中提供的特定组件吗? 如果是,它是哪个组中的哪个组件? 如果不是,如何创建一个选项卡?

6. 分层窗格有哪些预定义层次? 在预定义层次外可以创建新层次吗? 如果可以,怎么创建?

7. 桌面窗格与内部框架是什么关系? 桌面窗格与程序窗体是什么关系?

8. 什么是单文档界面? 什么是多文档界面? 如何使用 Java 语言创建多文档界面程序?

9. 工具栏必须定位于程序窗口的标题栏下面吗? 工具按钮必须水平排列吗?