单元七　Java GUI 应用程序开发

内容要点

1．AWT 及其图形界面组件。
2．AWT 布局管理器。
3．AWT 事件处理机制。

案例 7–1　登录窗口

案例任务

登录窗口是很多应用系统中不可缺少的组成部分。通过验证用户输入的用户名和密码，决定是否允许用户进入系统，在一定程度上保证系统的安全。本案例设计一个登录窗口，运行界面如图 7-1 所示。

图 7-1　案例 7-1 登录窗口

知识必备

众所周知，拥有图形用户界面的计算机应用程序生动、形象，更利于用户快速地掌握和使用。一些形象的图形甚至不用培训用户就能掌握其功能。那么，如何使用编程工具开发一个图形用户界面的应用程序呢？通过本单元对 Java 语言图形用户界面（GUI，Graphics User Interface）的学习，读者能较好地安排用户操作界面，使用户可以和程序之间方便地进行交互。读者还会学习 Java 的抽象窗口工具集（AWT，Abstract Window Toolkit）中包含的许多支持 GUI 的设计。

1. AWT 简介

Java 语言提供了两个处理图形用户界面的类库：java.awt 包和 javax.swing 包。本单元重点学习 AWT 包，Swing 包是在 AWT 包的基础上创建起来的程序包，所以要了解 Swing 包，必须先学好 AWT 包。

AWT 是抽象窗口工具集 Abstract Window Toolkit 的英文缩写。AWT 包中提供了大量进行 GUI 设计所使用的类和接口，包括绘制图形、设置字体和颜色、设置组件大小和位置、处理事件等内容。图 7-2 显示了 AWT 包中主要类及组件类的继承关系。

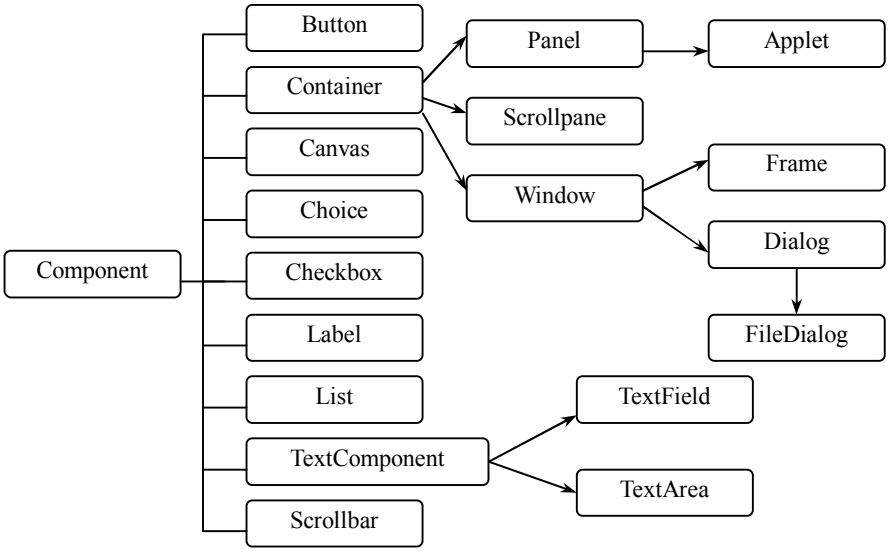


图 7-2　AWT 包中主要类及组件类的继承关系

从图 7-2 中我们可以看出，Component 类是 AWT 所有组件的最终抽象父类，它的子类分为两大部分：容器类与非容器类。

首先，学习一下 Component 类的主要方法，如表 7-1 所示。

表 7-1　组件类 Component 的常用方法

方法	说明
public void setVisible(Boolean b)	设置组件是否显示
public void setBounds(int x,int y,int w,int h)	设置组件的大小和显示位置
public void setBackground(Color c)	设置组件的背景颜色为 c
public void setSize(int width,int height)	设置组件的宽度和高度
public void setFont(Font f)	设置对象的字体样式为 f
public Color setForeground(Color c)	设置对象的前景色为 c
public void setLocation(int x,int y)	设置组件位置的左上角坐标（x,y）
public String getName()	返回对象的名称

接下来，我们认识一下容器类（Container）。本节开始程序中的 Frame 就是一个容器。Frame 类的构造方法和常用方法见表 7-2。那么，容器到底是什么？所谓容器就是可以包含其他组件的组件。容器类的层次关系请参见图 7-2。

表 7-2　Frame 类的构造方法和常用方法

构造方法	说明
public Frame()	创建一个没有标题的窗口框架
public Frame(String title)	创建一个标题为 title 的窗口框架
方法	**说明**
public void setMenuBar(MenuBar mb)	设置窗口的菜单条
public void setTitle(String title)	设置或修改窗口的标题
public String getTitle()	返回窗口的标题
public boolean isResizable()	判断窗口是否可以调整大小
public void remove(MenuComponent mc)	从窗口中删除给定的菜单项

若想向容器中添加组件，请使用 Container 类的 add()方法。反之，若想从中删除某个组件，则使用 remove()方法。容器类的主要方法如表 7-3 所示。

表 7-3　容器类 Container 的常用方法

方法	说明
public Component add(Component comp)	在容器中添加组件
public void setLayout(LayoutManager mgr)	设置组件容器的布局
public void remove(Component comp)	删除容器组件里指定的组件
public void paint(Graphics g)	重绘容器组件
public void paintComponent(Graphics g)	重绘容器组件里的所有组件

2. 组件（Components）

有了窗口之后，还要创建其他组件，然后将其添加到窗口中。本节将介绍 AWT 包中提供的基本组件，对于这些组件请参照图 7-2 所示的 AWT 类的层级图。

（1）标签（Label）

正如贴在饮料瓶上的商标一样，标签 Label 具有显示某段文本的功能。它可用 java.awt 类库里的 Label 类创建。表 7-4 列出了 Label 类的构造方法和常用方法。

表 7-4　Label 类的构造方法和常用方法

构造方法	说明
public Label()	创建一个没有文字的标签
public Label(String text)	创建标签，以 text 为标签上的文字
public Label(String text,int align)	创建标签，以 text 为标签上的文字，并以 align 的方式对齐，align 的值可为 Label.LEFT、Label.RIGHT、Label.CENTER，分别表示靠左、右和居中对齐
方法	**说明**
public int getAlignment()	返回标签上文字的对齐方式
public void setAlignment(int align)	设置标签上文字的对齐方式
public String getText()	返回标签上显示的文字
public void setText(String text)	设置标签上的文字为 text

（2）按钮（Button）

按钮用来响应用户的点击动作。当用户点击按钮的时候，与按钮相联系的程序将被执行，从而完成指定的功能。java.awt 类库提供了 Button 类来处理按钮控件的相关操作。表 7-5 给出了 Button 类的构造方法和常用方法。

表 7-5　Button 类的构造方法和常用方法

构造方法	说明
public Button()	创建一个没有标签的按钮
public Button(String label)	创建一个以 label 为标签的按钮
方法	**说明**
public void setLabel(String label)	设置按钮上的标签为 label
public String getLabel()	返回按钮上的标签

（3）单行文本框（TextField）

像输入密码和用户名时使用的输入框一样，用户可以向单行文本框中输入字符或修改其内容。TextField 类和 TextArea 类是 TextComponent 类的子类。表 7-6 列出了 TextComponent 类的常用方法。表 7-7 列出了 TextField 类的构造方法和常用方法。

表 7-6　TextComponent 类的常用方法

方法	说明
public void setEditable(boolean b)	设置文本组件是否可编辑
public void select(int selStart,int selEnd)	选择 selStart 与 selEnd 之间的文本
public void selectAll()	选择组件里的所有文本
public String getSelectText()	返回被选中的文本
Public String getText()	返回组件里的文本

表 7-7　TextField 类的构造方法和常用方法

构造方法	说明
public TextField()	创建文本框
public TextField(String text)	创建文本框，并以 text 为默认文本
public TextField(int columns)	创建文本框，并设置文本框的宽度可容纳 columns 个字符
public TextField(String text, int columns)	创建文本框，并以 text 为默认文本，设置文本框的宽度可容纳 columns 个字符
方法	**说明**
public void setEchoChar(char c)	设置文本框的回显字符
public void setText(String text)	设置文本框中的文本为 text
public void setColumns(int columns)	设置文本框的宽度为 columns 个字符
public int getColumns()	返回文本框默认的宽度，以字符为单位

（4）多行文本框（TextArea）

顾名思义，多行文本框就是可以输入多行文本的文本框，也称为文本区（如图 7-3 所示）。用户可以对其中的字符进行修改。表 7-8 列出了 TextArea 类的构造方法和常用方法。表 7-9 给出了 TextArea 类的数据成员。

图 7-3　多行文本框

表 7-8　TextArea 类的构造方法和常用方法

构造方法	说明
public TextArea()	创建一个 TextArea 对象
public TextArea(String text)	创建一个 TextArea 对象，text 为初始字符
public TextArea(int cols,int columns)	创建一个 cols 行、columns 列的 TextArea 对象
public TextArea(String text, int cols,int columns)	创建一个 TextArea 对象，text 为初始字符，同时设置行数和列数
public TextArea(String text, int cols,int columns, int scrollbars)	创建 TextArea 对象，text 为初始字符，设置行数和列数，同时加上滚动条的显示方式（参见表 7-9）
方法	**说明**
public void append(String str)	将字符串 str 加在文本末尾
public void insert(String str,int pos)	将字符串 str 插入 pos 位置
public void setColumns(String str,int start,int end)	将 start 到 end 之间的字符串替换为 str

表 7-9　TextArea 类的数据成员

数据成员	说明	代表数值
public SCROLLBARS_BOTH	具有水平与垂直滚动条	0
public SCROLLBARS_VERTICAL_ONLY	只有垂直滚动条	1
public SCROLLBARS_HORIZONTAL_ONLY	只有水平滚动条	2
public SCROLLBARS_NONE	没有滚动条	3

（5）复选框（Checkbox）

复选框是让用户选取项目的一种组件。当用户点击复选框时，若 Checkbox 的状态为 true（选中），则其状态变为 false（未选中）；若 Checkbox 的状态为 false，则其状态变为 true。

复选框分为复选（如图 7-4 所示）和单选（如图 7-5 所示）两种。表 7-10 列出了 Checkbox 类的构造方法和常用方法。

图 7-4　复选框

图 7-5　单选框

表 7-10　Checkbox 类的构造方法和常用方法

构造方法	说明
public Checkbox()	创建一个没有文字显示的复选框
public Checkbox(String label)	创建一个以 label 为标签的复选框
public Checkbox(String label,boolean state)	创建一个以 label 为标签的复选框，并设置复选框的初始状态为 state
public Checkbox(String label,boolean state, CheckboxGroup grp)	创建一个以 label 为标签的复选框，设置复选框的初始状态为 state，并将其加入到复选框组 grp 中
Public Checkbox(String label, CheckboxGroup gr, boolean state p)	同上，只是参数位置不同
方法	**说明**
public void setCheckboxGroup(CheckboxGroup grp)	设置复选框属于 grp 组
public void setState(boolean state)	设置复选框是否被选中

创建单选框需要 Checkbox 类与 CheckboxGroup 类一起使用。CheckboxGroup 类将 Checkbox 分成不同的组。用户在同一个 Checkbox 组中只能选择一个 Checkbox。

（6）列表框（List）和下拉列表框（Choice）

列表框（List）以目录的形式显示多个选项，用户可以从中选择一项或多项。我们把构成目录的元素称为列表项（item）。表 7-11 列出了 List 类的构造方法和常用方法。

表 7-11　List 类的构造方法

构造方法	说明
public List()	创建一个列表框，默认行数为 4 行
public List(int rows)	创建一个可以显示 rows 行的列表框
public List(int rows,boolean multipleMode)	创建一个可以显示 rows 行的列表框，multipleMode 为 true 表示多选，为 false 表示单选
方法	**说明**
public void add(String item)	将列表项 item 添加到列表框的末尾
public void add(String item,int index)	将列表项 item 添加到 index 位置。index 从 0 开始
public String getItem(int index)	返回 index 位置的列表项名称
public String getSelectedItem()	返回被选中的列表项的名称

下拉列表框（Choice）与列表框相似，它也是一个有许多选项的选择组件，如图 7-6 所示。但是，它只会将用户选择的选项显示出来，所以不会占据太大空间。表 7-12 列出了下拉列表框类 Choice 的构造方法和常用方法。

图 7-6　（下拉）列表框

表 7-12　Choice 类的构造方法和常用方法

构造方法	说明
public Choice()	创建一个下拉列表框
方法	**说明**
public void add(String item)	将列表项 item 添加到列表框的末尾
public void insert(String item,int index)	将列表项 item 插入到 index 位置
public String getItem(int index)	返回 index 位置的列表项名称
public String getSelectedItem()	返回被选中的列表项名称


```
1．import java.awt.*;
2．class AwtLabelButtonTextField{
3．        public static void main(String args[]){
4．            Frame frm=new Frame("登录窗口");                    //创建窗口 frm
5．            Label lab1=new Label("ID",Label.CENTER);            //创建标签
6．            Label lab2=new Label("密码",Label.CENTER);
7．            Button bun1=new Button("登录");                     //创建按钮
8．            Button bun2 =new Button("取消");
9．            TextField tf1=new TextField(10);                    //创建文本框
10．           TextField tf2=new TextField(10);
11．           frm.add(lab1);                                      //向窗口 frm 中添加标签 lab1
```

```
12.         frm.add(lab2);
13.         frm.add(bun1);                          //向窗口 frm 中添加按钮 bun1
14.         frm.add(bun2);
15.         frm.add(tf1);                           //向窗口 frm 中添加文本框 tf1
16.         frm.add(tf2);
17.         frm.setLayout(null);
18.         lab1.setBounds(20,50,100,30);           //设置标签的位置和大小
19.         lab2.setBounds(20,80,100,30);
20.         tf1.setBounds(120,50,100,30);           //设置文本框的位置和大小
21.         tf2.setBounds(120,80,100,30);
22.         bun1.setBounds(20,110,100,30);          //设置按钮的位置和大小
23.         bun2.setBounds(120,110,100,30);
24.         frm.setBounds(350,200,300,150);         //设置窗口的位置和大小
25.         frm.setVisible(true);                   //设置窗口可见
26.     }
27. }
```

本案例中使用 Frame 类创建了一个窗口，并在此窗口中创建了两个标签（Label）、两个按钮（Button）和两个文本框（TextField）。

小常识：即便您按了 Frame 上的关闭按钮，Frame 也不会关闭。在 DOS 窗口中，您可以通过按组合键 Ctrl+C 来强制关闭该 Frame。

说明：程序中的 Frame 类显然是 AWT 包中的类。setBounds()方法和 setVisible()方法是 Frame 类的方法或者其父类的方法。您认为我分析的对吗？实际上，只要我们掌握了 AWT 包中的类，就可以编写出图形用户界面程序。还等什么呢？赶快进入 AWT 世界吧！

现场演练

1．创建如图 7-7 所示的标签。

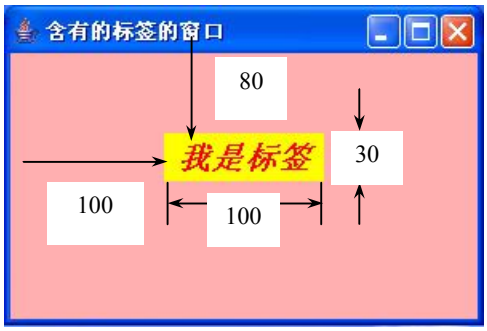


图 7-7　标签

2．学习相关组件的属性和方法，创建如图 7-8 所示的填写个人资料窗口。

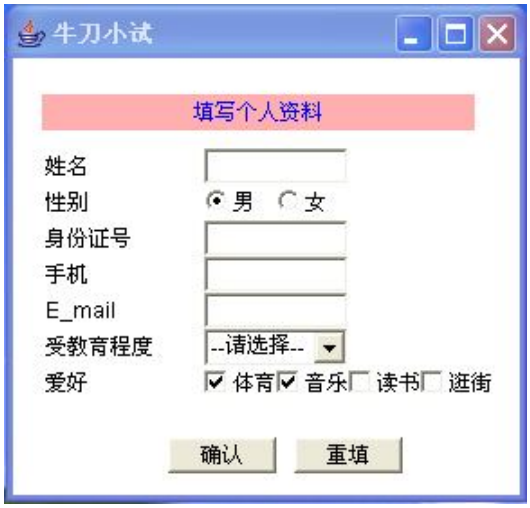


图 7-8　填写个人资料窗口

案例 7–2　计算器界面

案例任务

使用布局管理器实现计算器界面设计。如图 7-9 所示。

图 7-9　计算器界面

知识必备

在前面的学习中，都是先使用 setLayout(null)方法取消容器的默认布局管理器，然后使用 setBounds()方法设置组件的大小和位置。显然，使用该方法合理有序地摆放组件非常麻烦。而学习了布局管理器之后，这将是件非常轻松的事情。那么，布局管理器有什么作用呢？布局管理器能自动设置容器中组件的大小和位置，当容器大小改变时，它能自动调整其中组件的大小和位置。

容器可以通过调用 setLayout(布局管理器对象) 方法设置其布局管理器。向设置了布局管理的容器中添加组件都使用方法 add(参数)，参数随布局管理器的不同而不同。

下面介绍几种常用的布局管理器类。

1. BorderLayout 类

BorderLayout 是一种简单的布局策略，也称为边界式布局管理器（如图 7-10 所示）。BorderLayout 的布局策略是：

1）将容器空间分为北（North）、南（South）、西（West）、东（East）、中（Center）5 个区域。

2）每个组件都可占据某个区域，若没有指定区域，默认为中间（Center）。

3）若组件加入已被占用位置时，将取代原先的组件。

图 7-10　BorderLayout 布局管理器界面布局方式

BorderLayout 是容器 Frame 和对话框 Dialog 默认的布局管理器。

表 7-13 列出了 BorderLayout 类的构造方法，表 7-14 列出了表示组件摆放位置的数据成员。BorderLayout 类的常用方法请参阅相关读物。

表 7-13　BorderLayout 类的构造方法

构造方法	说明
public BorderLayout()	创建 BorderLayout 布局管理器，容器中的各组件之间没有间距
public BorderLayout(int hgap,int vgap)	创建 BorderLayout 布局管理器，容器中的各组件之间水平间距为 hgap，垂直间距为 vgap

表 7-14　BorderLayout 类中表示位置的数据成员

数据成员	说明	代表字符串
BorderLayout.EAST	组件放在容器右边	"East"
BorderLayout.WEST	组件放在容器左边	"West"
BorderLayout.NORTH	组件放在容器上边	"North"
BorderLayout.SOUTH	组件放在容器下边	"South"
BorderLayout.CENTER	组件放在容器中间	"Center"

向某个区域中添加组件的方法如下（以 EAST 为例）：add(组件对象,BorderLayout.EAST) 或 add(组件对象,"East")。

提示：Frame 默认的布局管理器就是 BorderLayout，因此可以不设置其布局管理器。但是，若使用其他的布局管理器则不能省略，必须使用 setLayout()方法设置布局管理器。

2. FlowLayout 类

FlowLayout 称为流式布局管理器，布局策略非常简单（如图 7-11、图 7-12 所示）。它的布局策略是：

1）组件按照加入容器的顺序从左向右排列。

2）一行排满以后就自动地转到下一行继续从左到右排列。

3）每一行中的组件都居中排列。

图 7-11　FlowLayout 布局 1

图 7-12　FlowLayout 布局 2

FlowLayout 是 Panel、Applet 小程序默认的布局管理器。

表 7-15 列出了 FlowLayout 类的构造方法，表 7-16 列出了 FlowLayout 类的主要数据成员。

表 7-15　FlowLayout 类的构造方法

构造方法	说明
public FlowLayout()	创建 FlowLayout 布局管理器，容器中的组件居中对齐，组件的水平和垂直间距默认为 5 个单位
public FlowLayout(int align)	创建同上的 FlowLayout 布局管理器，使用指定对齐方式（参见表 7-16）
public FlowLayout(int align,int hgap,int vgap)	创建同上的 FlowLayout 布局管理器，但组件间的水平间距为 hgap，垂直间距为 vgap

表 7-16　FlowLayout 类中表示对齐方式的数据成员

数据成员	说明	代表数值
FlowLayout.LEFT	每行组件靠左对齐	0
FlowLayout.CENTER	每行组件居中对齐	1
FlowLayout.RIGHT	每行组件靠右对齐	2

3．GridLayout 类

网格式布局管理器 GridLayout（如图 7-13 所示）的布局策略是：

1）将窗口划分成 n 行×m 列的网格区域。

2）组件按照加入容器的顺序从左到右、从上到下添加到网格区域中。

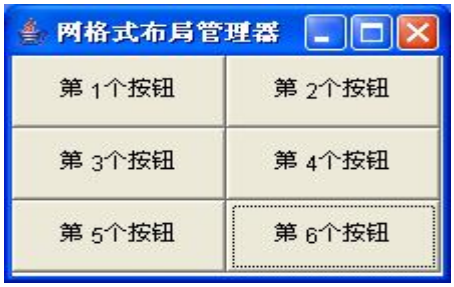


图 7-13　GridLayout 布局

表 7-17 列出了 GridLayout 类的构造方法。

表 7-17　GridLayout 类的构造方法

构造方法	说明
public GridLayout()	创建 GridLayout 布局管理器，使用默认值，每行只有一个组件
public GridLayout(int cows,int cols)	创建一个 cows 行、cols 列的 GridLayout 布局管理器
public GridLayout(int cows,int cols, int hgap,int vgap)	创建一个 cows 行、cols 列的 GridLayout 布局管理器，组件间的水平间距为 hgap，垂直间距为 vgap

布局管理器除了上面介绍的 BorderLayout、FlowLayout、GridLayout 之外，还有卡片式布局管理器 CardLayout 和网袋布局管理器 GridBagLayout，请参阅其他读物。


```
1．import java.awt.*;
2．public class TestLayout extends Frame{
3．static Panel pan=new Panel();                          //创建面板 pan
4．static Label lab=new Label("0.",Label.RIGHT);          //创建标签 lab
5．static Button b0,b1,b2,b3,b4,b5,b6,b7,b8,b9,bp,ba,bs,bm,bd,be;   //定义按钮
6．public static void main(String args[]){
7．    TestLayout frm=new TestLayout();
8．    b0=new Button("0");b1=new Button("1");b2=new Button("2");
9．    b3=new Button("3");b4=new Button("4");b5=new Button("5");
10．   b6=new Button("6");b7=new Button("7");b8=new Button("8");
11．   b9=new Button("9");bp=new Button(".");ba=new Button("+");
12．   bs=new Button("-");bm=new Button("*");bd=new Button("/");
13．   be=new Button("=");                                //初始化各按钮
```

```
14.  frm.setTitle("计算器界面");
15.  frm.setLayout(null);                                    //取消 frm 布局管理器
16.  frm.setSize(200,200);
17.  frm.setResizable(false);
18.  GridLayout grid=new GridLayout(4,4);
19.  pan.setLayout(grid);
20.  pan.setBounds(20,60,150,120);
21.  lab.setBounds(20,35,150,20);
22.  lab.setBackground(Color.cyan);
23.  pan.add(b7);pan.add(b8);pan.add(b9);pan.add(bd);
24.  pan.add(b4);pan.add(b5);pan.add(b6);pan.add(bm);
25.  pan.add(b1);pan.add(b2);pan.add(b3);pan.add(bs);
26.  pan.add(b0);pan.add(bp);pan.add(be);pan.add(ba);        //向面板中添加按钮
27.  frm.add(lab);
28.  frm.add(pan);                                          //向窗口 frm 中添加面板 pan
29.  frm.setVisible(true);                                  //设置窗口可见
30.  }
31. }
```

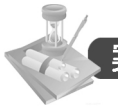
计算器界面由一个标签和若干按钮组成，按钮放在容器（Panel 类）pan 中，pan 采用 GridLayout 布局管理器，若干按钮按照 4 行 4 列摆放在面板 pan 中。

现场演练

1．编程实现图 7-9、图 7-10 和图 7-12 所示的布局。

2．使用 GridLayout 布局案例 7-1 的登录窗口。

案例 7–3　学生信息管理系统界面



案例任务

编程实现带有菜单的学生信息管理系统界面，如图 7-14 所示。

图 7-14　学生信息管理系统界面

之前一直使用的 Frame 就是一个容器，容器类的层次关系请参见图 7-2。接下来我们学习三个容器类。

1. 面板（Panel）

面板类 Panel 是容器类（Container）的直接子类，是一种没有标题的容器。它和 Frame 类有所区别。Panel 类具有如下特点：

1）Panel 必须位于窗口或其他容器内。实例化后必须使用 add()方法添加到窗口中。

2）默认的布局管理器是 FlowLayout，可使用 setLayout()方法改变其默认布局管理器。

3）Panel 中可以添加其他组件，例如按钮、标签等。Panel 一般用于布局和定位。

Panel 类的构造方法见表 7-18。

表 7-18　Panel 类的构造方法

构造方法	说明
public Panel()	创建一个面板
public Panel(LayoutManager layout)	创建一个以 layout 为布局管理器的面板

提示：Panel 类和 Frame 类一样也是容器（Container）类的子类，所以可以使用 Component 类和 Container 类的方法。

如图 7-15 所示在面板（Panel）中添加两个按钮。

图 7-15　面板

2.（文件）对话框（Dialog）

对话框类（Dialog）和框架类（Frame）都是 Window 的子类，它们也具有相近的性质，如都可以放置按钮、文本框等组件。但对话框右上角仅有一个关闭按钮，没有最小化和最大化按钮。

对话框主要用于显示信息或接收用户输入。我们必须给对话框指定所有者（owner），其所有者可以是 Frame 对象，也可以是另一个 Dialog 对象。其所有者被最小化，对话框也随之最小化；所有者被关闭，对话框也随之关闭。另外，Dialog 有模式和非模式两种方式，模式（modal）对话框总在最前面，若不处理完对话框，则不能对其他窗口进行操作；而非模式对话框在显示时，可以操作其他对话框。

表 7-19 给出了 Dialog 类的构造方法和常用方法。表 7-20 给出了 FileDialog 类的构造方法和常用方法。

表 7-19　Dialog 类的构造方法和常用方法

构造方法	说明
public Dialog(Dialog owner)	创建模式对话框，其所有者为另一对话框
public Dialog(Dialog owner,String title)	创建模式对话框，其所有者为另一对话框，同时设置标题为 title
public Dialog(Dialog owner, String title,boolean modal)	创建对话框，其所有者为另一对话框，标题为 title，模式由 modal 指定
public Dialog(Frame owner)	创建模式对话框，其所有者为一框架
public Dialog(Frame owner,String title)	创建模式对话框，其所有者为一框架，同时设置标题为 title
public Dialog(Frame owner,boolean modal)	创建对话框，其所有者为一框架，模式由 modal 指定
public Dialog(Frame owner, String title,boolean modal)	创建对话框，其所有者为一框架，标题为 title，模式由 modal 指定
方法	**说明**
public void show()	显示对话框
public void hide()	隐藏对话框
public void dispose()	销毁对话框对象
public void setModal(boolean b)	设置对话框模式
public void setResizable(boolean resizable)	设置对话框是否可以调整大小

表 7-20　FileDialog 类的构造方法和常用方法

构造方法	说明
public FileDialog(Frame parent)	创建所有者为 parent 的文件对话框
public FileDialog(Frame parent, String title)	创建文件对话框，所有者为 parent，标题为 title
public FileDialog(Frame parent, String title,int mode)	创建文件对话框，所有者为 parent，标题为 title，模式为 mode
方法	**说明**
public void setDirectory(String dir)	设置文件对话框默认打开的路径
public void setFile(String file)	设置文件对话框默认打开的文件
public void setMode(int mode)	设置文件对话框模式
public String getDirectory()	返回文件对话框内所取的路径
public String getFile()	返回文件对话框内所取的文件名
public int getMode()	返回文件对话框的模式：LOAD 或 SAVE

3. 菜单（Menu）组件

读者对菜单并不陌生，几乎所有应用软件都有菜单栏。Java 语言通过 MenuBar（菜单条）、Menu（菜单）、MenuItem（菜单项）三个类来实现菜单。一般情况下，建立菜单分为三个步骤：

1）向窗口中添加菜单条（MenuBar）。

2）在菜单条中添加菜单（Menu）。

3）向菜单中添加内容。内容一般是菜单项（MenuItem），也可以是子菜单（Menu）或菜单选项（CheckboxMenuItem）。

接下来学习一下三个类的构造方法和常用方法，分别见表 7-21、表 7-22 和表 7-23。

表 7-21　MenuBar 类的构造方法和常用方法

构造方法	说明
public MenuBar()	创建一个菜单条
方法	**说明**
public Menu add(Menu m)	向 MenuBar 中添加 Menu 对象 m
public void remove(int index)	删除位置为 index 的 Menu
public void remove(MenuComponent m)	删除 MenuComponent 对象 m
public Menu getMenu(int index)	返回位置为 index 的 Menu

表 7-22　Menu 类的构造方法和常用方法

构造方法	说明
public Menu()	创建一个 Menu
public Menu(String label)	创建一个标题为 label 的 Menu
方法	**说明**
public MenuItem add(MenuItem mi)	向 Menu 中添加 MenuItem 对象 mi
public void insert(MenuItem mi,int index)	在 index 位置插入 MenuItem 对象 mi
public void insert(String label,int index)	在 index 位置添加标题为 label 的 Menu
public void removeAll()	删除所有的 MenuItem

表 7-23　MenuItem 类的构造方法和常用方法

构造方法	说明
public MenuItem()	创建一个 MenuItem
public MenuItem(String label)	创建一个标题为 label 的 MenuItem
方法	**说明**
public void setLabel(String label)	设置 MenuItem 的标题为 label
public boolean isEnabled()	返回 MenuItem 是否可用
public void setEnabled(boolean b)	设置 MenuItem 可用

注意：窗口中添加菜单条（MenuBar）使用 setMenuBar(MenuBar mb)方法，而不是 add()方法。在菜单（Menu）中可以添加子菜单（Menu），请读者尝试。

案例透析

```
1．import java.awt.*;
2．class TestAwtMenu{
3．static Frame fr=new Frame("学生信息管理系统主界面");
4．static MenuBar mb=new MenuBar();                    //创建 1 个菜单条 mb
//创建 4 个菜单
5．static Menu    menuser=new Menu("用户管理");
6．static Menu    menuinfo=new Menu("信息管理");
7．static Menu    menuquery=new Menu("查询");
8．static Menu    menuhelp=new Menu("帮助");
//创建菜单 menuser 上的 4 个菜单项
9．static MenuItem    miadd=new MenuItem("增加用户");
10．static MenuItem    michange=new MenuItem("修改用户");
11．static MenuItem    midelete=new MenuItem("删除用户");
12．static MenuItem    miexit=new MenuItem("退出系统");
//创建菜单 menuinfo 上的 3 个菜单项
13．static MenuItem    miaddinfo=new MenuItem("增加信息");
14．static MenuItem    michangeinfo=new MenuItem("修改信息");
15．static MenuItem    mideleteinfo=new MenuItem("删除信息");
//创建菜单 menuquery 上的 2 个菜单项
16．static MenuItem    mibyno=new MenuItem("按学号");
17．static MenuItem    mibyname=new MenuItem("按姓名");
//创建菜单 menuhelp 上的 1 个菜单项
18．static MenuItem    mihelp=new MenuItem("帮助主题");
19．public static void main(String args[]){
20．      fr.setMenuBar(mb);                           //向窗口中添加菜单条
//向菜单条上添加 4 个菜单
21．      mb.add(menuser);
22．      mb.add(menuinfo);
23．      mb.add(menuquery);
24．      mb.add(menuhelp);
//向菜单 menuser 上添加菜单项
25．      menuser.add(miadd);
26．      menuser.add(michange);
27．      menuser.add(midelete);
28．      menuser.add(miexit);
//向菜单 menuinfo 上添加菜单项
29．      menuinfo.add(miaddinfo);
30．      menuinfo.add(michangeinfo);
31．      menuinfo.add(mideleteinfo);
//向菜单 menuquery 上添加菜单项
```

```
32.     menuquery.add(mibyno);
33.     menuquery.add(mibyname);
34.     menuhelp.add(mihelp);
//向菜单 menuhelp 上添加菜单项
35.     fr.setSize(300,300);
36.     fr.setVisible(true);
37. }
38. }
```

程序中先定义了菜单条 mb（第 4 行）、四个菜单（第 5～8 行）和 10 个菜单项（第 9～18 行）。然后将菜单条添加到窗口，将菜单添加到菜单条，将菜单项添加到相应的菜单（第 20～34 行）。

1．编程实现如图 7-15 所示窗口。

2．编程实现如图 7-16 所示窗口。

图 7-16　菜单

案例 7–4　简单“绘图板”

创建一个窗口，并能根据鼠标的移动轨迹在窗口中绘制曲线，如图 7-17 所示。

图 7-17　案例 7-4

知识必备

1. 基本概念

1）事件（event）：用户使用鼠标或键盘和窗口中的组件进行交互时所发生的事情。

2）事件源（event source）：就是能够产生事件的对象。

3）事件监听者：Java 程序把对事件进行处理的方法放在一个类对象中，这个类对象就是事件监听者。

4）事件处理接口：为了让监听者能对事件源发生的事件进行处理，创建该监听者对象的类必须实现相应的接口，即必须在类体中具体定义该接口中所有方法的方法体，以供监听者自动调用被类实现的某个接口的方法。

2. 委托事件模型

事件源产生某种事件时，Java 程序不会直接对该事件进行处理，它会将对该事件的处理工作委托给某个事件监听者。我们将这种事件处理模型称为委托事件模型（如图 7-18 所示）。

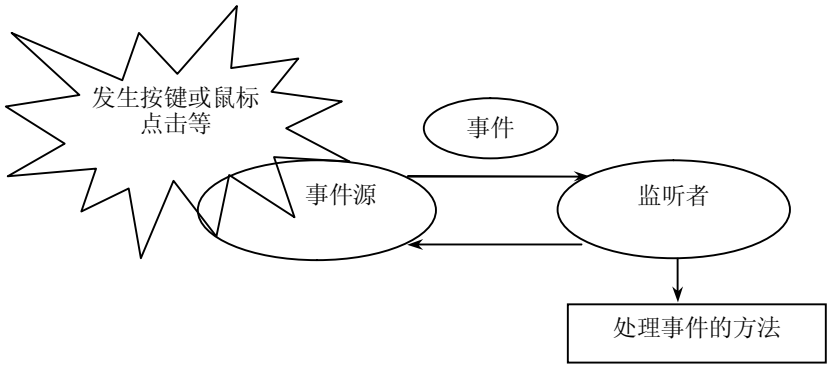


图 7-18　委托事件模型

在这种模型中，事件的产生者（事件源）和事件的处理者（监听者）是分离的。事件的处理者是一些实现了某个 Listeners 接口的类。某个事件必须先注册事件的监听者，即让实现了某个 Listeners 接口的类对象作为其处理者。当事件发生时，该处理者必须有相应的方法来接受这种类型的事件并对它进行处理。

Java 中处理各组件事件的一般步骤是：

1）定义实现事件监听接口的类。

2）创建事件监听者对象。

3）向事件源注册监听者对象（addListener(监听者)）。

3. AWT 事件类（Event）与监听器接口（Listener）

Java 语言在 java.awt.event 包中定义了许多事件类用于处理各种用户操作所产生的事件，它们都继承自 java.awt 包中的 AWTEvent 类。

对于每一个事件类，几乎都有相应的事件监听者。Java 语言的事件监听者绝大多数都是以接口形式给出的，它们都是继承自 java.util.EventListener 接口。事件类、对应的事件监听者接口与事件监听者接口所提供的方法这三者之间的关系见表 7-24。

表 7-24　事件类、对应的事件监听者接口与事件监听者接口所提供的方法

事件类/监听者	监听者接口所提供的事件处理者	产生事件的用户操作
ComponentEvent ComponentListener	ComponentMoved (ComponentEvent e)	移动组件时
	ComponentResized(ComponentEvent e)	改变组件大小时
	ComponentHidden(ComponentEvent e)	隐藏组件时
	ComponentShown(ComponentEvent e)	显示组件时
ContainerEvent ContainerListener	componentAdded(ContainerEvent e)	添加组件时
	componentRemoved(ContainerEvent e)	删除组件时
ActionEvent ActionListener	actionPerformed(ActionEvent e)	单击按钮、在文本行中单击 Enter 键、双击列表框选择菜单项时
AdjustmentEvent AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)	当组件的值被改变时
ItemEvent ItemListener	itemStateChanged(ItemEvent e)	选择复选框、单选框、单击列表框、选中带复选框的菜单项时
TextEvent TextListener	textValueChanged(TextEvent e)	在文本行、文本区中修改内容时
MouseEvent MouseListener	mouseClicked(MouseEvent e)	单击鼠标时
	mouseEntered(MouseEvent e)	鼠标进入指定区域时
	mouseExited(MouseEvent e)	鼠标离开指定区域时
	mousePressed(MouseEvent e)	按下鼠标时
	mouseReleased(MouseEvent e)	松开鼠标时
MouseEvent MouseMotionListener	mouseDragged(MouseEvent e)	拖动鼠标时
	mouseMoved(MouseEvent e)	移动鼠标时
KeyEvent KeyListener	keyPressed(KeyEvent e)	按下键盘上的按键时
	keyReleased(KeyEvent e)	释放键盘上的按键时
	keyTyped(KeyEvent e)	按键盘按键时
WindowEvent WindowListener	windowActivated(WindowEvent e)	激活窗口时
	windowClosing(WindowEvent e)	关闭窗口时
	windowClosed(WindowEvent e)	关闭窗口后
	windowDeactivated(WindowEvent e)	窗口失去焦点时
	windowDeiconified(WindowEvent e)	窗口从最小还原为正常时
	windowIconified(WindowEvent e)	窗口最小化成图标时
	windowOpened(WindowEvent e)	打开窗口时
FocusEvent FocusListener	focusGained(FocusEvent e)	获得焦点时
	focusLost(FocusEvent e)	失去焦点时

每当在事件源上发生一个操作时，就会产生相应的事件对象。表 7-25 给了事件源与其产生的事件对象的对应关系。

表 7-25　AWT 组件可能产生事件的对应关系

事件源	产生事件的类类型
Button	ActionEvent
CheckBox	ActionEvent、ItemEvent
Component	ComponentEvent、FocusEvent、KeyEvent、MouseEvent
MenuItem	ActionEvent
Scrollbar	AdjustmentEvent
TextField	ActionEvent
TestArea	ActionEvent
Window	WindowEvent

4. 动作事件（ActionEvent）及处理

动作事件对应于组件的主要用途。点击按钮，选择菜单项目，或向单行文本框输入字符串并按 Enter 键时，都会发生动作事件。在 Java 中动作事件由 ActionEvent 类表示。ActionEvent 所对应的监听者是 ActionListener 接口，该接口所提供的事件处理方法是 actionPerformed (ActionEvent e)。ActionEvent 类的常用方法如表 7-26 所示。

表 7-26　ActionEvent 类的常用方法

方法	说明
public String getActionCommand()	获取事件源的动作命令（actionCommand）字符串
Public int getModifiers()	返回事件发生时所按下的修饰符

EventObject 类是 ActionEvent 类的祖先类，而 EventObject 类中的 getSource()方法可以返回事件源。

下面就以动作事件为例学习利用委托事件模型编写事件处理程序。

例：窗口中摆放一个按钮，当单击该按钮时，窗口的背景变为红色。

（1）让“事件源”所在类的对象作为监听者

```
1．import java.awt.*;
2．import java.awt.event.*;
3．public class aevent extends Frame implements ActionListener{
4．        static aevent fr=new aevent();
5．        static Button button=new Button("红色");
6．        public static void main(String args[]){
7．        button.addActionListener(fr);
8．        fr.setBounds(100,100,300,200);
9．        fr.setBackground(Color.pink);
10．       fr.setLayout(new FlowLayout());
11．       fr.add(button);
```

```
12.        fr.setVisible(true);
13.    }
14.    public void actionPerformed(ActionEvent e){
15.          Button bu=(Button)e.getSource();
16.          if(bu==button) fr.setBackground(Color.red);
17.     }
18. }
```

本代码选择由包含事件源（按钮）的类（aevent）的对象来担任监听者，因此 aevent 类必须实现（implements）ActionListener 接口。程序第 3 行正是这么做的。

确定了监听者，接下来就是向事件源（button）注册监听者 fr，如程序第 7 行。

最后，在 aevent 类内定义 ActionListener 接口的 actionPerformed()方法，把事件处理程序编写在里面即可，如程序第 14～17 行。

（2）定义内部类担任监听者

```
1. import java.awt.*;
2. import java.awt.event.*;
3. public class aevent2 extends Frame {
4.         static aevent2 fr=new aevent2();
5.         static Button button=new Button("红色");
6.         public static void main(String args[]){
7.         button.addActionListener(new MyActionListener());
8.         fr.setBounds(100,100,300,200);
9.         fr.setBackground(Color.pink);
10.          fr.setLayout(new FlowLayout());
11.          fr.add(button);
12.          fr.setVisible(true);
13.     }
14.    static class MyActionListener implements ActionListener{
15.     public void actionPerformed(ActionEvent e){
16.           Button bu=(Button)e.getSource();
17.           if(bu==button) fr.setBackground(Color.red);
18.      }
19.     }
20. }
```

程序第 14 行定义了 MyActionListener 类实现了 ActionListener 接口，并定义了 actionPerformed()方法。第 7 行将 MyActionListener 类的对象注册为 button 的监听者。

运行结果如图 7-19 所示。

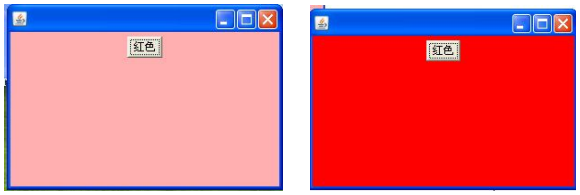


图 7-19　按钮改变窗口背景

5. 窗口事件（WindowEvent）及处理

窗口事件类 WindowEvent 是指对窗口本身进行操作时所产生的事件。窗口事件类的监听者是 WindowListener 接口，该接口中声明了 7 个用来处理不同事件的方法。WindowListener 接口的方法如表 7-27 所示。

表 7-27　WindowListener 接口的方法

处理事件的方法	事件说明
public void windowOpened(WindowEvent e)	窗口打开时
public void windowActivated(WindowEvent e)	窗口激活时
public void windowDeactivated(WindowEvent e)	窗口失活时，即窗口失去焦点时
public void windowClosing(WindowEvent e)	企图关闭窗口时（关闭前）
public void windowClosed(WindowEvent e)	窗口被关闭后
public void windowIconified(WindowEvent e)	窗口由正常大小变为最小化图标时
public void windowDeiconified(WindowEvent e)	窗口由最小化图标恢复成正常大小时

例：创建事件类 WindowEvent 的应用。建立窗口 frm 对象，并在其上添加一个标签对象，用来显示对窗口的各种不同操作。

```
1．import java.awt.*;
2．import java.awt.event.*;
3．public class WindowEventDemo extends Frame{
4．    static Label lab=new Label();
5．    static WindowEventDemo frm=new WindowEventDemo();
6．    static MyWinListener winlist=new MyWinListener();
7．    public static void main(String args[]){
8．        frm.setLayout(null);
9．        frm.setTitle("窗口事件");
10．       frm.setBounds(120,50,215,100);
11．       lab.setBounds(25,25,150,50);
12．       frm.add(lab);
13．       frm.addWindowListener(winlist);
14．       frm.setVisible(true);
15．       }
16．  static class MyWinListener implements WindowListener{
17．       public void windowOpened(WindowEvent e){
18．           lab.setText("打开新窗口");
19．       }
20．       public void windowActivated(WindowEvent e){
21．           lab.setText("窗口被激活");
22．       }
23．       public void windowIconified(WindowEvent e){
24．           frm.setTitle("窗口被最小化");
25．       }
```

```
26.         public void windowDeiconified(WindowEvent e){
27.             frm.setTitle("窗口被还原成正常大小");
28.         }
29.         public void windowClosing(WindowEvent e){
30.             frm.dispose();
31.             System.exit(0);
32.         }
33.         public void windowDeactivated(WindowEvent e){}
34.         public void windowClosed(WindowEvent e){}
35.         }
36.  }
```

运行结果如图 7-20 所示。

图 7-20　窗口事件

6. 适配器类（Adapter）

通过接口来完成事件处理时，尽管某些方法没有用到，但也要同时实现该接口的所有方法。但是有时只是需要对其中的某些方法进行处理，因此写上所有方法显得很麻烦。为了方便起见，Java 为某些监听者接口提供了适配器类，当需要对某种事件进行处理时，只需要让事件处理类继承事件所对应的适配器类，覆盖本次操作用到的事件处理方法即可，不必实现无关的事件处理方法。

适配器类是一个类而不是一个接口，因而处理事件的类只能继承一个适配器类。当该类需要处理多种事件时，通过继承适配器类的方法是不行的。但可以基于适配器类，用内部类的方法来处理这种情况。如果想用作事件监听者的类已经继承了其他类，就不能用适配器类，只能去实现事件监听者接口了。接口与其对应的适配器类如表 7-28 所示。

表 7-28　接口对应的适配器类

接口名称	适配器名称
ComponentListener	ComponentAdapter（组件监听适配器）
ContainerListener	ContainerAdapter（容器监听适配器）
FocusListener	FocusAdapter（焦点监听适配器）
KeyListener	KeyAdapter（键盘监听适配器）
MouseListener	MouseAdapter（鼠标监听适配器）
MouseMotionListener	MouseMotionAdapter（鼠标移动监听适配器）
WindowListener	WindowAdapter（窗口监听适配器）

7. 键盘事件（KeyEvent）及处理

键盘事件类 KeyEvent 继承自 InputEvent 类。当用户在键盘上输入字符时便会触发此类事件。KeyEvent 类的常用方法如表 7-29 所示。

表 7-29　KeyEvent 类的常用方法

方法	说明
public char getKeyChar()	返回按下的字符
public int getKeyCode()	返回按下字符的代码
public boolean isActionKey()	判断所按下的键是否为 Action Key，所谓 Action Key 是指方向键、PgUp、PgDn 与 F1～F12 等键

使用 KeyListener 接口处理 KeyEvent 事件时，必须以类实现 KeyListener 接口，该接口里声明了 3 个方法。KeyListener 接口的方法如表 7-30 所示。

表 7-30　KeyListener 接口的方法

处理事件的方法	事件说明
public void keyPressed(KeyEvent e)	按下按键时发生
public void keyReleased(KeyEvent e)	放开按键时发生
public void keyTyped(KeyEvent e)	单击字符时发生，即按下按键与放开按键这一整个事件，但不包括输入 Action Key

8. 鼠标事件（MouseEvent）及处理

鼠标事件类 MouseEvent 是一些常见的鼠标操作。如鼠标单击事件源、鼠标指针进入或离开事件源，移动、拖动鼠标等操作，均会触发鼠标事件。鼠标事件类 MouseEvent 的常用方法如表 7-31 所示。

表 7-31　MouseEvent 的常用方法

方法	说明
public Point getPoint()	返回鼠标按钮被按下时的点的坐标，返回的是 Point 类型
public int getX()	返回鼠标按钮被按下时的点的 x 坐标
public int getY()	返回鼠标按钮被按下时的点的 y 坐标

MouseEvent 类一般以 MouseListener 接口或 MouseMotionListener 接口作为监听者。MouseListener 接口的方法如表 7-32 所示。MouseMotionListener 接口的方法如表 7-33 所示。

表 7-32　MouseListener 接口的方法

处理事件的方法	事件说明
public void mouseClicked(MouseEvent e)	在事件源上单击鼠标按钮时发生
public void mouseEntered(MouseEvent e)	鼠标指针移入事件源上方时发生
public void mouseExited(MouseEvent e)	鼠标指针移出事件源上方时发生
public void mousePressed(MouseEvent e)	按下鼠标的任一按钮时发生
public void mouseReleased(MouseEvent e)	放开鼠标被按下的按钮时发生

注意：上述 5 个鼠标事件中有些事件几乎是同时发生。对于 mouseClicked、mousePressed 与 mouseReleased 这 3 个事件的触发顺序是：mouseReleased 事件先发生，然后 mouseClicked 发生，最后是 mousePressed 发生。

表 7-33　MouseMotionListener 接口的方法

处理事件的方法	事件说明
public void mouseDragged(MouseEvent e)	拖动鼠标时发生
public void mouseMoved(MouseEvent e)	移动鼠标时发生


```
1．import java.awt.*;
2．import java.awt.event.*;
3．public class TestMouseEvent extends Frame{
4．Graphics g;
5．int x,y,ox,oy;
6．public TestMouseEvent(String title){
7．        super(title);
8．        setSize(200,200);
9．        setVisible(true);
10．        g=this.getGraphics();
11．        g.setColor(Color.red);
12．        addMouseMotionListener(new MyMouseMotionAdapter());
13．        addMouseListener(new MyMouseAdapter());
14．}
15．public static void main(String[] args){
16．        TestMouseEvent f=new TestMouseEvent("绘制图形");
17．}
18．class MyMouseMotionAdapter extends MouseMotionAdapter{
19．          public void mouseDragged(MouseEvent e){
20．               x=e.getX();y=e.getY();
21．               g.drawLine(ox,oy,x,y);
22．               ox=x;oy=y;
23        }
24．}
25．  class MyMouseAdapter extends MouseAdapter{
26．          public void mousePressed(MouseEvent e){
27．               ox=e.getX();oy=e.getY();
28．         }
29．    }
30．}
```

程序通过继承适配器类实现事件处理。由于主类 TestMouseEvent 已经继承了其他类，所以又定义了两个内部类分别继承鼠标事件（MouseEvent）的监听接口 MouseListener 和 MouseMotionListener（如代码第 18～29 行）。第 20 行表示若移动鼠标获取当前坐标，第 21 行根据获得的坐标绘图，也就是鼠标移动的轨迹。第 12～13 行分别将内部类的对象注册为事件监听者。

1．简述事件机制和原理。

2．编写事件处理程序，窗口中包含两个组件，一个按钮，一个文本区，当单击按钮后，文本区中的字体颜色设置为红色。

Swing 组件

1．按钮

（1）基本概念

按钮是 Java 图形用户界面的基本组件之一，经常用到的按钮有 4 种形式：JButton、JToggleButton、JCheckBox、JradioButton，这些按钮类均是 AbstractButton 的子类或间接子类。它们之间的继承关系如图 7-21 所示。

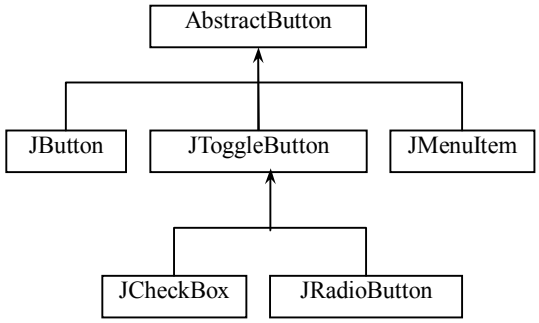


图 7-21　按钮类继承关系

（2）普通按钮

JButton 是最简单的按钮，其主要构造方法如表 7-34 所示。

表 7-34　JButton 类构造方法

构造方法	说明
JButton()	创建一个既没有显示文本也没有图标的按钮
JButton(Icon icon)	创建一个没有显示文本但有图标的按钮

续表

构造方法	说明
JButton(String text)	创建一个有显示文本但没有图标的按钮
JButton(String text,Icon icon)	创建一个既有显示文本又有图标的按钮

例：程序运行时，每当按动按钮时，就会在屏幕上交替显示出两条不同信息。

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4. class JButtonExample extends WindowAdapter implements ActionListener{
5.    JFrame f;
6.    JButton b;
7.    JTextField tf;
8.    int tag=0;
9.          public static void main(String[] args){
10.         JButtonExample be=new JButtonExample();
11.         be.go();
12.         }
13.   public void go(){
14.         f=new JFrame("JButton Example");
15.         b=new JButton("Sample");
16.         b.addActionListener(this);
17.         f.getContentPane().add(b,"South");
18.         tf=new JTextField();
19.         f.getContentPane().add(tf,"Center");
20.         f.addWindowListener(this);
21.         f.setSize(300,200);
22.         f.setVisible(true);
23.         }
24.   public void actionPerformed(ActionEvent e){
25.         String s1="你已经按过按钮！";
26.         String s2="请下次再试！";
27.         if(tag==0){
28.               tf.setText(s1);
29.               tag=1;
30.               }else{
31.                     tf.setText(s2);
32.                     tag=0;
33.                     }
34.         }
35.   public void windowClosing(WindowEvent e){
36.         System.exit(0);
37.         }
38.   }
```

运行结果如图 7-22 所示。

图 7-22　JButton 按钮

（3）切换按钮、复选按钮及单选按钮

- 切换按钮（JToggleButton）

JToggleButton 是具有两种状态的按钮，即选中和未选中状态。

JToggleButton 的构造方法如表 7-35 所示。

表 7-35　JToggleButton 的构造方法

构造方法	说明
JToggleButton()	创建一个既没有显示文本也没有图标的切换按钮
JToggleButton(Icon icon)	创建一个没有显示文本但有图标的切换按钮
JToggleButton(Icon icon,Boolean selected)	创建一个有图标和指定初始状态但没有显示文本的切换按钮
JToggleButton(String text)	创建一个有显示文本但没有图标的切换按钮
JToggleButton(String text, Boolean selected)	创建一个有显示文本和指定初始状态但没有图标的切换按钮
JToggleButton(String text, Icon icon)	创建一个既有显示文本又有图标的切换按钮
JToggleButton(String text, Icon icon ,Boolean selected)	创建一个既有显示文本又有图标和指定初始状态的切换按钮

- 复选按钮（JCheckBox）

JCheckBox 是具有两种状态的按钮，即选中和未选中状态。

JCheckBox 的构造方法如表 7-36 所示。

表 7-36　JCheckBox 的构造方法

构造方法	说明
JCheckBox()	创建一个既没有显示文本也没有图标的复选按钮
JCheckBox(Icon icon)	创建一个没有显示文本但有图标的复选按钮
JCheckBox(Icon icon,Boolean selected)	创建一个有图标和指定初始状态但没有显示文本的复选按钮
JCheckBox(String text)	创建一个有显示文本但没有图标的复选按钮
JCheckBox(String text, Boolean selected)	创建一个有显示文本和指定初始状态但没有图标的复选按钮
JCheckBox(String text, Icon icon)	创建一个既有显示文本又有图标的复选按钮
JCheckBox(String text, Icon icon, Boolean selected)	创建一个既有显示文本又有图标和指定初始状态的复选按钮

- 单选按钮（JRadioButton）

JRadioButton 是具有两种状态的按钮，即选中和未选中状态。

JRadioButton 的构造方法如表 7-37 所示。

表 7-37　JRadioButton 类构造方法

构造方法	说明
JRadioButton()	创建一个既没有显示文本也没有图标的单选按钮
JRadioButton(Icon icon)	创建一个没有显示文本但有图标的单选按钮
JRadioButton(Icon icon,Boolean selected)	创建一个有图标和指定初始状态但没有显示文本的单选按钮
JRadioButton(String text)	创建一个有显示文本但没有图标的单选按钮
JRadioButton(String text, Boolean selected)	创建一个有显示文本和指定初始状态但没有图标的单选按钮
JRadioButton(String text, Icon icon)	创建一个既有显示文本又有图标的单选按钮
JRadioButton(String text, Icon icon ,Boolean selected)	创建一个既有显示文本又有图标和指定初始状态的单选按钮

注意：在 JToggleButton 类中定义了一个 isSelected()方法，通过该方法可以获知按钮的当前状态：当返回值为真（true）时表示处于选中状态，而返回值为假（false）时则表示处于未选中状态。

2. 标签

标签对象通常用于显示提示性的文本信息或图标。

JLable 的构造方法如表 7-38 所示。

表 7-38　JLable 的构造方法

构造方法	说明
JLable()	构造一个既不显示文本信息也不显示图标的空标签
JLable(Icon image)	构造一个显示图标的标签
JLable(String text)	构造一个显示文本信息的标签
JLable(Icon image,int horizontalAlignment)	构造一个显示图标的标签，水平对齐方式由 int 型参数指定
JLable(String text,int horizontalAlignment)	构造一个显示文本信息的标签，水平对齐方式由 int 型参数指定
JLable(String text,Icon icon,int horizontalAlignment)	构造一个同时显示文本信息和图标的标签，水平对齐方式由 int 型参数指定

3. 组合框

组合框是一个下拉式菜单，它有两种形式：不可编辑的和可编辑的。

JComboBox 的构造方法和常用方法如表 7-39 所示。

表 7-39　JComboBox 常用的构造方法和方法

构造方法	说明
JComboBox()	创建一个没有任何可选项的组合框
JComboBox(Object[] items)	根据 Object 数组创建组合框，Object 数组的元素即为组合框中的可选项
方法	**说明**
public void addItem(Object anObject)	在末尾位置添加新的可选项
public void insertItemAt(Object anObject,int index)	在 index 指定的位置添加新的可选项 anObject
public void removeAllItems()	删除所有可选项
public void removeItem(Object anObject)	删除由 anObject 指定的可选项
public void removeItemAt(int anIndex)	删除指定位置（由 anIndex 指定）的可选项

4. 列表

列表是可供用户进行选择的一系列选项。

JList 有 4 种构造方法，如表 7-40 所示。

表 7-40　JList 构造方法

构造方法	说明
JList()	构造一个空列表
JList(ListModel dataModel)	构造一个列表，列表的可选项由 ListModel 型参数 dataModel 指定
JList(Object[] listData)	构造一个列表，列表的可选项由对象数组 listData 指定
JList(Vector listData)	构造一个列表，列表的可选项由 Vector 型参数 ListData 指定

5. 文本组件

（1）基本概念

文本组件可用于显示信息和提供用户输入功能，在 Swing 中提供了文本域（JTextField）、口令输入域（JPasswordField）、文本区（JTextArea）等多个文本组件，这些文本组件都有一个共同的基类——JTextComponent。

在 JTextComponent 中定义的主要方法如表 7-41 所示。

表 7-41　JTextComponent 类主要方法

方法	说明
getSelectedText()	从文本组件中提取被选中的文本内容
getText()	从文本组件中提取所有文本内容
getText(int offs,int len)	从文本组件中提取指定范围的文本内容
select(int start,int end)	在文本组件中选中指定范围的文本内容
selectAll()	在文本组件中选中所有文本内容
setEditable(Boolean b)	设置为可编辑或不可编辑状态
setText(String t)	设置文本组件中的文本内容
setDocument(Document doc)	设置文本组件的文档

续表

方法	说明
Copy()	复制选中的文本到剪贴板
Cut()	剪切选中的文本到剪贴板
Paste()	将剪贴版的内容粘贴到当前位置

（2）文本域

文本域是一个单行的文本输入框，可用于输入少量文本。

文本域有 5 种构造方法，如表 7-42 所示。

表 7-42　JTextField 类构造方法

构造方法	说明
JTextField()	构造一个空文本域
JTextField(int columns)	构造一个具有指定列数的空文本域，int 型参数 columns 指定文本域的列数
JTextField(String text)	构造一个显示指定初始字符串的文本域，String 型参数 text 指定要显示的初始字符串
JTextField(String text,int columns)	构造一个具有指定列数、并显示指定初始字符串的文本域，String 型参数 text 指定要显示的初始字符串，int 型参数 columns 指定文本域的列数
JTextField(Document doc,String text,int columns)	构造一个使用指定文档、具有指定列数、并显示指定初始字符串的文本域，Document 型参数 doc 指定使用的文档，String 型参数 text 指定要显示的初始字符串，int 型参数 columns 指定文本域的列数

（3）文本区

文本区是一个多行多列的文本输入框。

JTextArea 的构造方法如表 7-43 所示。

表 7-43　JTextArea 的构造方法

构造方法	说明
JTextArea()	构造一个空文本区
JTextArea(String text)	构造一个显示指定初始字符串的文本区，String 型参数 text 指定要显示的初始字符串
JTextArea(int rows,int columns)	构造一个具有指定行数和列数的空文本区，int 型参数 rows 和 columns 分别指定文本区的行数和列数
JTextArea(String text,int rows,int columns)	构造一个具有指定行数和列数，并显示指定初始字符串的文本区，String 型参数 text 指定要显示的初始字符串，int 型参数 rows 和 columns 分别指定文本区的行数和列数
JTextArea(Document doc)	构造一个使用指定文档的文本区
JTextArea(Document doc,String text,int rows,int columns)	构造一个使用指定文档的文本区，具有指定的行数和列数，并显示指定初始字符串

6. 菜单组件

（1）基本概念

菜单也是最常用的 GUI 之一，Swing 包中提供了多种菜单组件，它们的继承关系如图 7-23 所示。

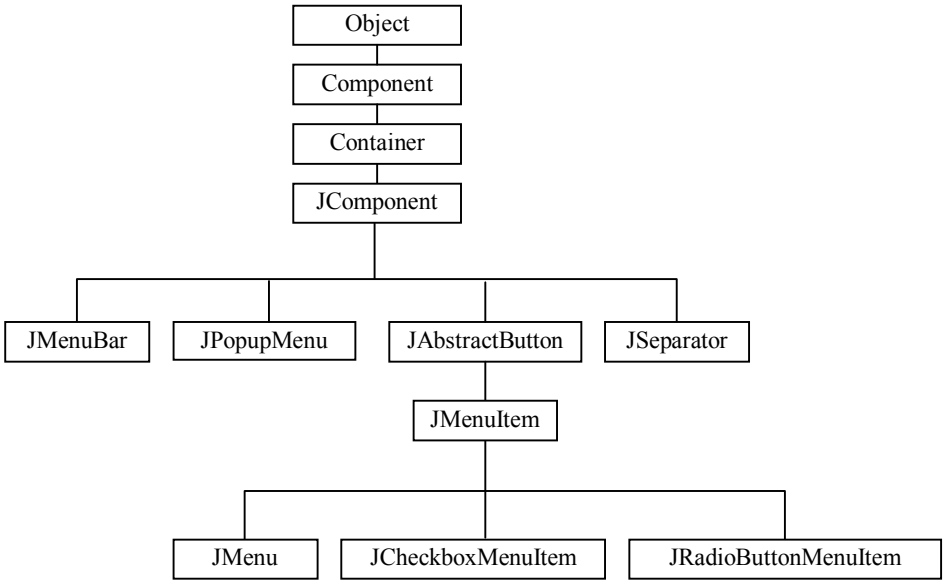


图 7-23　菜单类继承关系

（2）菜单栏

菜单栏是窗口中的主菜单，用来包容一组菜单。

JMenuBar 的构造方法见表 7-44。

表 7-44　JMenuBar 的构造方法

构造方法	说明
JMenuBar()	创建一个菜单栏

（3）菜单

菜单是最基本的下拉菜单，用来包容一组菜单项或子菜单。

JMenu 的构造方法见表 7-45。

表 7-45　JMenu 的构造方法

构造方法	说明
public JMenu(String label)	创建一个下拉菜单，其中 String 型参数 label 指定了菜单上的文本

（4）菜单项

如果将整个菜单系统看作是一棵树，那么菜单项就是这棵树的叶子，是菜单系统最下一级。

JMenuItem 的构造方法见表 7-46。

表 7-46　JMenuItem 的构造方法

构造方法	说明
JMenuItem(Icon icon)	构造一个只显示图标的菜单项
JMenuItem(String text)	构造一个只显示文本的菜单项
JMenuItem(String text,Icon icon)	构造一个同时显示文本和图标的菜单项
JMenuItem(String text,int mnemonic)	构造一个显示文本并且有快捷键的菜单项

（5）复选菜单项和单选菜单项

复选菜单项和单选菜单项是两种特殊的菜单项。可以对其做选中和不选中操作。

JCheckBoxMenuItem 的构造方法见表 7-47。

表 7-47　JCheckBoxMenuItem 的构造方法

构造方法	说明
JCheckBoxMenuItem(Icon icon)	构造一个只显示图标的复选菜单项
JCheckBoxMenuItem(String text)	构造一个只显示文本的复选菜单项
JCheckBoxMenuItem(String text,boolean b)	构造一个只显示文本的复选菜单项，并且可设定选中状态
JCheckBoxMenuItem(String text, Icon icon)	构造一个既显示图标又显示文本的复选菜单项
JCheckBoxMenuItem(String text, Icon icon, boolean b)	构造一个既显示图标又显示文本的复选菜单项，并且可以设定选中状态

（6）弹出式菜单

弹出式菜单是一种比较特殊的独立菜单，可以根据需要显示在指定位置。

JPopupMenu 的两种构造方法和常用方法见表 7-48。

表 7-48　JPopupMenu 的构造方法和常用方法

构造方法	说明
public JPopupMenu()	构造一个没有名称的弹出式菜单
public JPopupMenu(String label)	构造一个有指定名称的弹出式菜单
方法	**说明**
public void show(Component invoker,int x,int y)	在这个方法中需要一个组件作为参数，该组件的位置将作为显示弹出式菜单的参考原点

注意：显示弹出式菜单必须使用 show()方法。

7. 对话框

（1）对话框

对话框是与框架类似的可移动窗口，它与框架的区别在于：对话框具有较少的修饰并且能够被设置为“模式”窗口，即在该窗口被关闭之前，其他窗口无法接收任何形式的输入。

JDialog 的构造方法见表 7-49。

表 7-49　JDialog 的构造方法

构造方法	说明
JDialog(Frame owner)	构造一个没有标题的非模式对话框
JDialog(Frame owner,boolean modal)	构造一个没有标题的对话框，boolean 型参数 modal 指定对话框是否为模式窗口
JDialog(Frame owner,String title)	构造一个有标题的非模式对话框，String 型参数 title 指定对话框的标题
JDialog(Frame owner,String title ,boolean modal)	构造一个有标题的对话框，String 型参数 title 指定对话框的标题，boolean 型参数 modal 指定对话框是否为模式窗口

注意：应当将对话框视为一种可以反复使用的资源，即当某个对话框不需要显示时，不要立即将其清除，而是继续保留它，等待以后再用。

（2）标准对话框

JDialog 通常用于创建自定义的对话框，除此之外，在 Swing 中还提供了用于显示标准对话框的 JOptionPane 类。

在 JOptionPane 类中定义了多个 showXxxDialog 形式的静态方法，如表 7-50 所示。

表 7-50　JOptionPane 类中 showXxxDialog 形式的静态方法

方法	说明
showConfirmDialog	确认对话框，显示问题，要求用户确认
showMessageDialog	信息对话框，显示信息，告知用户发生了什么情况
showOptionDialog	选项对话框，显示选项，要求用户选择
showInputDialog	输入对话框，提示用户输入

（3）文件对话框

文件对话框是专门用于对文件进行浏览和选择的对话框。

JFileChooser 的三种构造方法和常用方法，如表 7-51 所示。

表 7-51　JFileChooser 构造方法和常用方法

构造方法	说明
JFileChooser()	根据用户的默认目录创建文件对话框
JFileChooser(File currentDirectory)	根据 File 型参数 currentDirectory 指定的目录创建文件对话框
JFileChooser(String currentDirectoryPath)	根据 String 型参数指定的目录创建文件对话框
方法	**说明**
showOpenDialog(Component parent)	将显示一个“打开”文件的对话框
showSaveDialog(Component parent)	将显示一个“保存”文件的对话框
showDialog(Component parent,String s)	显示一个“自定义”文件的对话框

注意：刚创建的对话框是不可见的，可以调用方法将其显示出来。

1．简述图形用户界面的构成成分以及各自的作用？

2．叙述 AWT 和 Swing 组件的关系？

3．什么是事件源？什么是监听者？Java 语言的图形用户界面中，谁可以充当事件源？谁可以充当监听者？

4．动作事件的事件源可以有哪些？如何响应动作事件？

5．简述 Java 的事件处理机制？

6．组件与容器有何异同？

7．Frame 和 Dialog 有何异同？

8．编程实现如图 7-24 所示的窗口。当点击窗口的关闭按钮时，会弹出对话框，对话框包含三个组件，一个标签和两个按钮。当点击“是”按钮时，窗口被关闭；当点击“取消”按钮时，对话框被隐藏，窗口不动。

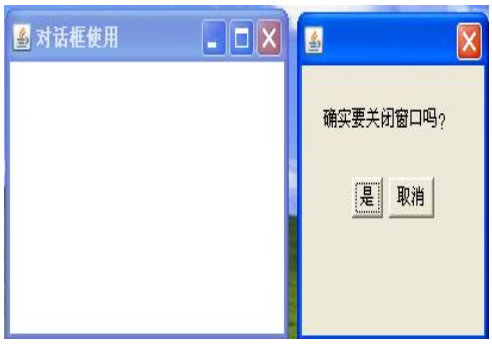


图 7-24　练习 8

9．编写如图 7-25 所示的事件处理程序。窗口中有两个菜单（颜色和窗口），颜色菜单包含四个菜单项（红色、绿色、蓝色和斜体），窗口菜单下有一个菜单项（关闭）。当点击菜单项时，文本区中字体的格式会发生相应的变化；当点击“关闭”菜单项时，窗口被关闭。

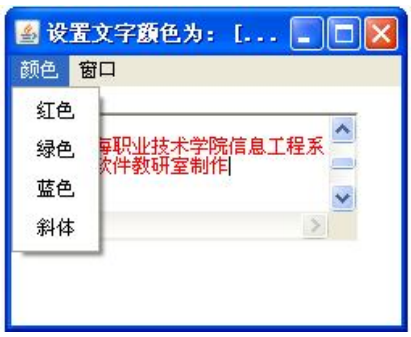


图 7-25　练习 9