

第 4 章 图形用户界面设计

本章要点

- Java 图形用户界面设计的基本知识
- 布局管理器的应用
- Java 常用图形用户界面设计组件的应用
- Java 常用组件事件处理的应用

4.1 认识 AWT 包和 Swing 包

用户界面是计算机用户与软件之间的交互接口。一个功能完善，使用方便的用户界面可以使软件的操作更加简单，使用户与程序之间的交互更加有效。因此图形用户界面（graphics user interface, GUI）的设计和开发已经成为软件开发中的一项重要的工作。

Java 语言提供的开发图形用户界面（GUI）的功能包括 AWT(Abstract Window Toolkit) 和 Swing 两部分。这两部分功能由 Java 的两个包来完成—awt 和 swing。虽然这两个包都是用于图形用户界面的开发，但是它们不是同时被开发出来了。awt 包是最早被开发出来的。但是使用 awt 包开发出来的图形用户界面并不完美，在使用上非常的不灵活。比如 awt 包所包含的组件，其外观是固定的，无法改变，这就使得开发出来的界面非常死板。这种设计是站在操作系统的角度开发图形用户界面，主要考虑的是程序与操作系统的兼容性。这样做的最大问题就是灵活性差，而且程序在运行时还会消耗很多系统资源。

由于 awt 包的不足表现，SUN 公司于 1998 年针对它存在的问题，对其进行了扩展，开发出了 Swing，即 swing 包。但是，SUN 公司并没有让 swing 包完成替代 awt 包，而是让这两个包共同存在，互取所需。awt 包虽然存在缺点，但是仍然有可用之处，比如在图形用户界面中用到的布局管理器、事件处理等依然采用的是 awt 包的内容。

Java 有两个主要类库分别是 Java 包和 Javax 包。在 Java 包中存放的是 Java 语言的核心包。Javax 包是 Sun 公司提供的扩展包，它是对原 Java 包的一些优化处理。

swing 包由于是对 awt 包的扩展和优化，所以是存放在 Javax 包下的，而 awt 包是存放在 Java 包下的。虽然 swing 是扩展包，但是，现在的图形用户界面基本都是基于 swing 包开发的。

swing 包的组件大部分是采用纯 Java 语言进行开发的，这就大大增加了组件的可操作性，尤其是组件的外观。通常情况下，只要通过改变所传递的参数的值，就可以改变组件的外观。而且 swing 包还提供 Look and Feel 功能，通过此功能可以动态改变外观。Swing 包中也有一些组件不是用纯 Java 语言编写的，这些组件一般用于直接和操作系统进行交互的。

4.2 布局管理器

在本节中将对图形用户界面中的布局管理器进行介绍。

【任务 1】认识图形用户界面。

1. 基础知识

先来看一个用 Java 语言编写的图形用户界面的例子，如图 4-1 所示。

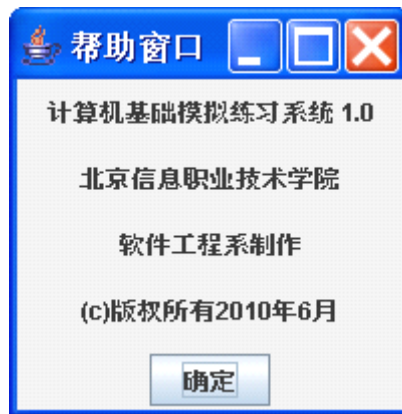


图 4-1 一个图形用户界面

图 4-1 是一个帮助文件的用户界面。通过这个界面，我们来介绍三个与图形用户界面有关的术语。

(1) 组件：构成图形用户界面的各种元素称为组件。图 4-1 中，放在“帮助窗口”中的每个一信息都是一个组件。例如“确定”按钮就是一种组件。

(2) 容器：是图形用户界面中容纳组件的部分，一个容器可容纳一个或多个组件，甚至可以容纳其他容器。窗口就是一种容器。例如图 4-1 中的组件全部都是放在“帮助窗口”这个容器中的。容器与组件的关系就像杯子和水的关系。需要说明的是，容器也可以被称为组件。

(3) 布局管理器：组件在被放到容器中时，要遵循一定的布局方式。在 Java 的图形用户界面中，有专门的类来管理组件的布局，称这些类为布局管理器。在“帮助窗口”中每一条信息都是一个组件，这些组件应该如何安排就是由布局管理器负责的。所谓的布局管理器，实际上就是能够对组件进行布局管理的类。

用吃饭的例子就能很好的说明这三个术语之间的关系。容器就是吃饭的桌子，组件就是一盘盘的菜。我们放菜的时候总要安排一下哪个菜应该放在什么位置，这就是布局管理器。

2. 任务实施

本任务的代码内容如下：

1	<code>import java.awt.*;</code>
2	<code>import javax.swing.*;</code>
3	<code>public class FrameEx</code>
4	<code>{</code>
5	<code> public void go()</code>
6	<code> {</code>
7	<code> JFrame win = new JFrame("帮助窗口");</code>
8	<code> Container contentPane = win.getContentPane();</code>
9	<code> contentPane.setLayout(new GridLayout(5,1));</code>
10	<code> JLabel labOne = new JLabel("计算机基础模拟练习系统的 1.0",JLabel.CENTER);</code>
11	<code> JLabel labTwo = new JLabel("北京信息职业技术学院",JLabel.CENTER);</code>
12	<code> JLabel labThree = new JLabel("软件工程系制作",JLabel.CENTER);</code>
13	<code> JLabel labFour = new JLabel("(c)版权所有 2010 年 6 月",JLabel.CENTER);</code>
14	<code> JButton queding = new JButton("确定");</code>

15	<code>contentPane.add(labOne);</code>
16	<code>contentPane.add(labTwo);</code>
17	<code>contentPane.add(labThree);</code>
18	<code>contentPane.add(labFour);</code>
19	<code>contentPane.add(queding);</code>
20	<code>win.setSize(200,200);</code>
21	<code>win.setVisible(true);</code>
22	<code>}</code>
23	<code>public static void main(String arg[]){</code>
24	<code>FrameEx fe = new FrameEx();</code>
25	<code>fe.go();</code>
26	<code>}</code>
27	<code>}</code>

3. 分析与提高

(1) 第 1 行和第 2 行的作用就是将 `awt` 包和 `swing` 包引入到该程序中，因为该程序中要用到 `awt` 包和 `swing` 包中的类。例如第 8 行用到的 `Container` 类和第 9 行的 `GridLayout` 类都属于 `awt` 包。其他以“J”开头的类都属于 `swing` 包。这样做的目的很容易理解。例如，`Container` 类是 Java 语言的开发者事先定义好的类。关于这个类的一些功能也是设定好的。作为使用者，只要将这个类用 `import` 的方式引入到自己的程序中就可以使用它了。这两个包的引用会在图形用户界面的程序中经常用到。

(2) 第 7 行定义的是一个窗口。第 8 行是开发图形用户界面必须要用到的类，这个类的作用会在后面的章节进行更详细的解释。

(3) 第 9 行是对窗口进行了布局管理器设置，通过 `GridLayout` 布局管理器，将窗口划分成 5 行，1 列。

(4) 第 10 行至第 13 行定义了窗口要显示的信息。

(5) 第 14 行定义了“确定”按钮。

(6) 第 15 至第 19 行是将定义好的组件放入到窗口中。

(7) 第 20 行定义了窗口的初始大小。

(8) 第 21 行定义了窗口的可显示性。

(9) 第 24 行和第 25 行是 `FrameEx` 类的对象，并通过这个对象调用 `go()` 方法，使这个帮助窗口运行起来。

在此只对程序进行粗略解释，目的只是让读者对图形用户界面有个初步的认识。在后面的任务中会逐个对各种组件和它们的应用进行详细介绍。

4.2.1 Flow Layout 布局管理器应用

【任务 2】认识 FlowLayout 布局管理器。

1. 基础知识

`FlowLayout` 是一个最简单的布局管理器，这个布局管理器的功能就是将容器中的组件从左到右，从上到下的排列。因此，它又被称为流布局管理器。例如图 4-1 中，帮助窗口共有五个组件。它们在窗口中的排列方式是 5 行，1 列。如果采用 `FlowLayout` 布局管理器，那

么窗口的执行结果应该是图 4-2 的样式。所有组件是按照添加时的顺序，按从左到右，从上到下的顺序排列的。默认情况下，第一个被添加的组件摆放在第 1 行居中位置，其后添加的组件摆放在第一个组件的后面。当第 1 行再也放不下组件的时候，其后的组件从第 2 行开始从左到右摆放。以此类推，直到添加完所有组件。

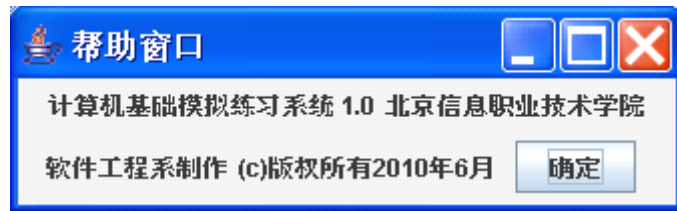


图 4-2 FlowLayout 布局管理器

2. 任务实施

任务 1 中的第 9 行就是对窗口布局的设置语句。语句如下：

```
contentPane.setLayout(new GridLayout(5,1));
```

只要对这一行进行更改，就可以达到应用 FlowLayout 布局管理器的目的。

contentPane 是一个对象，它指向的是 Container 类。关于这个类的含义和作用会在以后的章节中详细讲解。读者暂时只需要知道 contentPane 引用是指向所建立的“帮助窗口”就可以了。

从 contentPane 所调用的 setLayout()方法的方法名的含义就可以知道这个方法的作用——设置（set）布局（layout）。通过对这个方法参数赋予不同布局管理器的引用，达到对窗口进行布局设置的目的。

要实现将“帮助窗口”的布局管理器设置为 FlowLayout，只需要做如下的更改。

```
contentPane.setLayout(new FlowLayout());
```

其中的参数 new FlowLayout()实际上是 FlowLayout 类的一个引用。这条语句的含义就是：将 contentPane 所指向窗口的布局管理器设置成 FlowLayout。

还有另外一种设置布局管理器的方法。如下语句所示。

```
语句 1: FlowLayout flow= new FlowLayout();
```

```
语句 2: contentPane.setLayout(flow);
```

语句 1 创建了 FlowLayout 布局管理器的引用。语句 2 将语句 1 所创建的 flow 引用作为参数传递给 setLayout()方法。

flow 引用等价于 new FlowLayout()。因此，语句 contentPane.setLayout(new FlowLayout())实际上是语句 1 和语句 2 的组合写法。

3. 分析与提高

FlowLayout 布局管理器是 Panel 和 Applet 容器的默认布局管理器。如果不专门对 Panel 和 Applet 进行布局管理器设置的话，它们的组件的布局将按 FlowLayout 进行管理。Panel 和 Applet 都是一种容器。Panel 和窗口很类似，Applet 是嵌在浏览器中使用的一种客容器。这两个容器会在后面的章节中介绍。

FlowLayout 布局管理器对窗口中的组件的管理是与窗口的大小有关的。在任务 1 中有如下的语句：

```
win.setSize(200,200);
```

这条语句设置了窗口的初始值大小。第一个参数为窗口的宽度，第二个参数为窗口的高度。FlowLayout 布局管理器会根据这个大小来摆放组件。如果我们使用拖动的方法，改变窗口的大小，窗口中组件的排列方式也会跟着改变。

每个组件也有自己的初始大小。当窗口设置的宽度小于组件的初始宽度时，窗口宽度会自动变为组件的宽度，但窗口高度不会改变。

在摆放组件时，如果不设定组件的对齐方式，那么 `FlowLayout` 布局管理器是按居中对齐的方式摆放组件的。也可以通过向 `FlowLayout` 构造方法传递参数的方式设置所需的对齐方式。`FlowLayout` 布局管理器的对齐方式有：`FlowLayout.CENTER`、`FlowLaout.TRAILING` 和 `FlowLayout.LEADING`。分别表示居中对齐、右对齐和左对齐。

设置的方法如下面的语句所示：

```
contentPane.setLayout(new FlowLayout(FlowLayout.LEADING));
```

此语句可以将组件的对齐方式设置为左对齐

还可以设置组件之间的水平和垂直间距。

设置的方法如下面的语句所示：

```
contentPane.setLayout(new FlowLayout(FlowLayout.LEADING , 50 , 50));
```

这条语句的含义就是将组件间的水平和垂直间距都设置为 50。

4.2.2 BorderLayout 布局管理器应用

【任务 3】认识 BorderLayout 布局管理器。

1. 基础知识

`BorderLayout` 又称边界布局管理器，它将窗口划分为上北、下南、左西、右东和中央五个区域，分别用参数 `BorderLayout.NORTH`、`BorderLayout.SOUTH`、`BorderLayout.WEST`、`BorderLayout.EAST` 和 `BorderLayout.CENTER` 来表示。在窗口中添加组件时，系统会根据参数将组件摆放到窗口的相应位置。如图 4-3 所示。

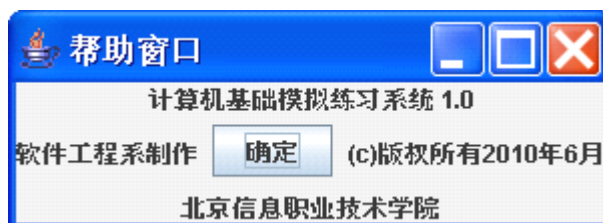


图 4-3 BorderLayout 布局管理器

如果某个区域没有摆放组件，则其他组件会占用此位置。规则如下：

当上北或下南没有摆放组件时，左西、右东和中央的组件会占用上北或下南的位置。

当左西或右东没有摆放组件时，上北、下南和中央的组件会占用左西或右东的位置。

当上北、下南、左西和右东都没有摆放组件时，中央的组件会占用这些位置。

当中央位置没有摆放组件时，位置被空缺，其他位置的组件不会占用中央的位置。

2. 任务实施

通过 `setLayout()` 方法可以完成 `BorderLayout` 布局管理器的设置，语句如下：

```
contentPane.setLayout(new BorderLayout());
```

在添加组件时，要对组件摆放的位置进行说明。语句如下：

```
contentPane.add(queding, BorderLayout.CENTER);
```

此语句的含义是将 `queding` 按钮摆放在窗口的中央位置。

如果在添加组件时不进行位置说明，系统则默认将组件放在中央位置的。当添加的组件大于一个时，只有最后添加的组件被显示。

3. 分析与提高

BorderLayout 是 Frame 容器的默认布局管理器。当 Frame 容器中没有指定布局管理器时，系统就会默认采用 BorderLayout 布局管理器。

采用 BorderLayout 布局管理器还可以设置组件之间的间距。格式如下：

```
new BorderLayout(int hGap,int vGap);
```

参数 hGap 和 vGap 分别表示组件之间的水平间距和垂直间距。如果不指定组件间的间距，则组件之间没有间距。

指定组件位置还有另外一组参数值，分别是 BorderLayout.PAGE_START、BorderLayout.PAGE_END、BorderLayout.LINE_START、BorderLayout.LINE_END 和 BorderLayout.CENTER。这两组参数的对应关系如表 4-1 表示。

表 4-1 两组位置参数对照表

参数组 1	参数组 2
NORTH	PAGE_START
SOUTH	PAGE_END
WEST	LINE_START
EAST	LINE_END
CENTER	CENTER

这两组参数在使用时没有区别，读者可以根据习惯使用。参数组 1 相对参数组 2 要好记忆一些。

4.2.3 GridLayout 布局管理器应用

【任务 4】认识 GridLayout 布局管理器。

1. 基础知识

GridLayout 又称网格布局管理器。相对于 FlowLayout 和 BorderLayout 来说，GridLayout 布局管理器是比较灵活的一种管理器。它可以通过行数和列数的设置，把窗口划分成若干个单元格，将组件放在这些单元格里。任务 1 中采用的就是这种布局管理器。

2. 任务实施

任务 1 将窗口设置为 5 行 1 列的表格，各个组件分别放在这个表格中的五个单元格里。这种布局的设置通过语句 `contentPane.setLayout(new GridLayout(5,1))` 实现的。

GridLayout 的括号中两个参数的含义就是（行数，列数）。通过改变这两个参数的值，将窗口划分成不同的单元格。

3. 分析与提高

GridLayout 布局管理器也可以不设置行数和列数，这时系统会默认行数和列数均为 1。GridLayout 布局管理器同样可以对组件之间的间距进行设置。格式如下：

```
new GridLayout(int rows,int columns,int hGap,int vGap)
```

rows 和 columns 分别表示行数和列数。hGap 和 vGap 分别表示组件之间的水平间距和垂直间距。

如果要摆放的组件大于 GridLayout 所设置的单元格个数，多出来的组件也会被摆放在窗口中，但是会破坏原来的单元格式样。

4.2.4 自定义布局管理器的应用

【任务 5】认识自定义布局管理器。

1. 基础知识

FlowLayout、BoderLayout 和 GridLayout 布局管理器有一个共同的特点就是布局是固定的。因此组件的摆放也就被固化了。比如 BorderLayout 只能把组件摆放在五个区域内，不可能有第六个区域。然后有很多界面中的组件摆放非常灵活。虽然 GridLayout 可以将窗口设置成不同的单元格，但是碰到组件摆放没有规律可寻的情况时，也会无能为力。有时候，为了达到窗口中组件的不规则摆放，需要对这三种布局管理器进行综合应用，实现起来非常麻烦。于是，Java 就提供了一种不使用布局管理器的方法，即自定义布局。

这种方法的理念是这样的：所有图形用户界面都是平面的，界面上的每个点都可以用 x 和 y 两个坐标来确定。在一个界面上如果选取一个点，再确定好要摆放组件的宽度和高度，就可以确定出一个区域。如图 4-4 所示。

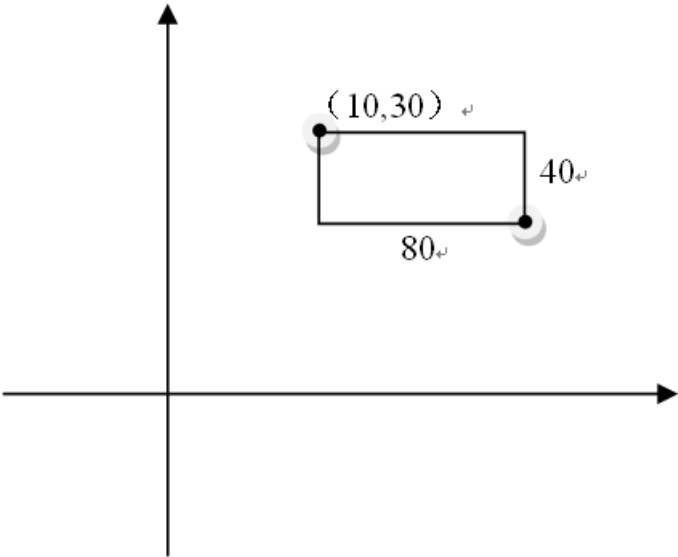


图 4-4 一个平面中两点坐标确定一个区域

(10, 30) 是起始坐标。80 和 40 是组件的宽度和高度，从而确定出一个区域，就可以把一个组件放在这个区域中。这样的布局方式就非常灵活，可以根据需要在窗口的任意位置摆放组件。事先只要给每个组件在窗口中确定一个摆放的位置就可以了。这时的布局不是针对窗口的了，而是针对每一个组件。例如，一个“确定”按钮。我们给它起个名字叫 button。下面的语句就可以在窗口中设置出用于摆放该按钮的区域。

`button.setBounds(10,30,80,40);`

`setBounds` 是设置 (set) 边界 (bound) 的意思。因此这条语句的含义就是将宽度为 80，高度为 40 的 button 放在以坐标 (10, 30) 为起始点的区域中。

2. 任务实施

本任务的代码内容如下：

1	<code>import java.awt.*;</code>
---	---------------------------------

2	import javax.swing.*;
3	public class FrameEx
4	{
5	public void go()
6	{
7	JFrame win = new JFrame("帮助窗口");
8	Container contentPane = win.getContentPane();
9	contentPane.setLayout(null);
10	JButton queding = new JButton("确定");
11	queding.setBounds(10, 30, 80, 40);
12	contentPane.add(queding);
13	win.setSize(200,200);
14	win.setVisible(true);
15	}
16	public static void main(String arg[]){
17	FrameEx fe = new FrameEx();
18	fe.go();
19	}
20	}

这段代码是对任务 1 的代码进行了部分修改得出的。只保留任务 1 中的“确定”按钮组件，并利用自定义布局方式将这个按钮摆放在窗口中。

3. 分析与提高

(1) 第 9 行将窗口的布局管理器设置为 `null`。意思就是这个窗口不使用布局管理器。既然要应用自定义的布局，就应该将容器原有的布局管理器去掉。虽然我们可以不进行布局管理器设置，但是每种容器都有默认的布局管理器，因此，必须将这种默认值去掉，自定义的布局才能发挥作用。比如这段程序采用是 `JFrame` 容器，这种容器属于 `Frame` 容器。`Frame` 容器的默认布局管理器是 `BorderLayout`。必须通过第 9 行的方法，使 `BorderLayout` 布局管理器不起作用，才能达到自定义布局的效果。

(2) 第 11 行设置了“确定”按钮在窗口中的坐标位置。其中前两组参数表示按钮左上角的坐标，后两组参数确定了组件的宽度和高度。

(3) 第 12 行将“确定”按钮添加到窗口时，会按照第 11 行确定的区域进行摆放。需要注意的是，组件的宽度和高度要设计得合理，否则，可能会使组件不能完全显示或者出现显示变形。

4.3 常用组件和事件处理

这一节将介绍一些在设计图形用户界面时常用的组件，以及与这些组件有关的事件处理。

4.3.1 学习 `JFrame` 和 `JPanel` 的使用

`JFrame` 和 `JPanel` 都属于 `swing` 包下的类，它们都是容器组件。`Swing` 对 `AWT` 进行了扩展，增加了 `awt` 包下组件的功能，为了与原来组件进行区别，在 `swing` 包下的所有组件名称

都在原来名字的前面加了一个“J”。因此，在 Java 的图形用户界面中，以“J”为首字母的组件都归属于 swing 包。在 awt 包下有 Frame 和 Panel 组件，JFrame 和 JPanel 就是对它们进行扩展得到的。

【任务 6】利用 JFrame 组件建立一个窗口。

1. 基础知识

JFrame 是 Java 图形用户界面中最底层的容器之一。前面的章节中，曾用吃饭的例子来比喻容器，组件和布局管理器之间的关系。JFrame 就相当于“桌子”，它可以用来容纳用户界面中的所有组件。

JFrame 在图形用户界面中的表现形式就是窗口。

通过 JFrame 创建窗口的常用方式有两种：

方式一：JFrame 对象名 = new JFrame();

方式二：JFrame 对象名 = new JFrame(String s);

窗口在标题栏部分都会显示其提示性信息，以表明这个窗口的作用。方法二可以实现这一功能。其中 String s 就是用于存放标题栏要显示的信息的参数。

语句 JFrame win = new JFrame(“帮助窗口”)的含义是：创建名字为 win 的 JFrame 窗口，标题栏要显示的信息为“帮助窗口”。

用方式一创建的窗口，标题栏内容为空。

JFrame 类包括一些方法，这些方法帮助我们完成与窗口有关的操作。

(1) 设置窗口大小的方法

pack()和 setSize(w,h)都可以用于控制窗口的大小。pack()方法会根据所容纳的组件自定义窗口的大小。setSize(w,h)方法设置的窗口大小是固定的，不受组件的影响。如果设置的窗口小于组件大小，会出现部分组件不能显示的情况。这种情况在使用 pack()方法时不会出现。

以上两种方法设置的窗口默认显示在屏幕的左上角。如果要改变初始显示的位置就要用 setBounds(x,y,w,h)方法。这个方法不但可以设置窗口的大小，而且可以设置窗口在屏幕上显示的位置。其中 x 和 y 是设置窗口在屏幕上显示的起始坐标，w 和 h 是设置窗口的大小。

(2) 标题设置方法

通过 setTitle(String s)方法可以设置窗口的标题。例如，win.setTitle(“提示信息”)，这条语句可以将 win 所指向的窗口标题设置为“提示信息”。

(3) 设置窗口前景色和背景色方法

方法 setForeground(Color c)用于设置窗口前景色。方法 setBackground(Color c)用于设置窗口背景色。这两个方法的参数都是颜色参数，Color 是专门用于处理颜色的类。

下面的语句是将 win 指向的窗口背景色设置为蓝色。

win.setBackground(Color.blue);

(4) 显示窗口方法

对窗口进行各种设置后，需要执行 setVisible(boolean b)方法，将窗口显示在屏幕上。其中的参数是布尔值。当参数值为 true 时，显示窗口；当参数值为 false 时，不显示窗口。show()方法也可以实现显示窗口的功能，它与 setVisible(boolean b)方法在应用上没有区别。

2. 任务实施

以下程序创建了一个不含任何组件的窗口。

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

3	public class FrameEx
4	{
5	public void go()
6	{
7	JFrame win = new JFrame("帮助窗口");
8	win.setSize(200,200);
9	//win.pack();
10	//win.setBounds(50,50,200,200);
11	win.setVisible(true);
12	//win.show();
13	}
14	public static void main(String arg[]){
15	FrameEx fe = new FrameEx();
16	fe.go();
17	}
18	}

这段程序的执行结果如图 4-5 所示。



图 4-5 不包含组件的窗口

3. 分析与提高

- (1) 第 1 行和第 2 行是引用语句。
- (2) 第 7 行是最关键的。JFrame win = new JFrame("帮助窗口"); 它创建了一个名字为 win，标题为“帮助窗口”的窗口。
- (3) 第 8 行 win.setSize(200,200)的作用就是设置这个窗口有多大，参数的含义是（宽度，高度）。这句话的含义就是将 win 这个窗口的大小（size）设置（set）为宽 200，高 200。
- (4) 第 9 行和第 10 行被注释掉的。这两条语句同样也是用于设置窗口大小。读者自行实验，尝试采用三种窗口大小设置方法的不同。
- (5) 第 11 行 win.setVisible(true)中，setVisible 的含义就是设置（set）显示（visible）状态。显示状态只有两种：显示（true）和不显示（false）。所以第 11 行的含义就是将 win 窗口的显示状态设置为显示。只有参数是 true 的时候，才会看到图 4-5 的窗口。
- (6) 第 12 行被注释掉的语句，这条语句同第 11 行的作用一样，用于显示窗口。

【任务 7】在窗口中加入 JPanel 容器，将窗口划分成两个区域。

1. 基础知识

(1) JPanel 的应用与 JFrame 几乎完全一样, JFrame 能容纳的组件都能摆放在 JPanel 容器中。在布局管理器的应用上也是一样的。但是不同的是 JFrame 可以独立应用, 而 JPanel 只能依附于 JFrame 才能应用。也就是说, 如果想应用 JPanel 的话, 之前必须有一个 JFrame 建立的容器存在, 才可以在这个已经存在的容器上再应用 JPanel。

(2) JPanel 是隐性显示的。它并不像 JFrame 建立的窗口那样可以显示出来。利用这一特点, 就可以在 JFrame 建立的窗口上应用若干个 JPanel, 把 JFrame 制作出来的容器划分成若干个区域, 每个区域都可以独立应用。这样做的目的显而易见。如果在 JFrame 制作出来的窗口中只用到一种布局管理器, 是没有必要再用 JPanel 的, 但是如果要在一个窗口中以不同的布局方式摆放组件, 就要用到 JPanel 了。因为 JPanel 可以独立应用, 就可以让一个窗口具有更多的布局方式, 使窗口的组件摆放更加灵活。

(3) 创建 JPanel 的常用方式有两种。

(4) 方式一: JPanel 对象名 = new JPanel();

(5) 方式二: JPanel 对象名 = new JPanel(布局管理器);

(6) 应用方式二创建 JPanel 时, 同时就可以设置它的布局管理器。如下面的语句:

```
JPanel jp = new JPanel(new FlowLayout());
```

2. 任务实施

本任务的代码内容如下:

1	import java.awt.*;
2	import javax.swing.*;
3	public class FrameEx
4	{
5	public void go()
6	{
7	JFrame win = new JFrame("帮助窗口");
8	JPanel jpOne = new JPanel();
9	JPanel jpTwo = new JPanel();
10	win.setLayout(new GridLayout(2,1));
11	win.add(jpOne);
12	win.add(jpTwo);
13	win.setSize(200,200);
14	win.setVisible(true);
15	}
16	public static void main(String arg[]){
17	FrameEx fe = new FrameEx();
18	fe.go();
19	}
20	}

程序的执行结果同图 4-5 一样。但是, 这个窗口已经被分为上下两个区域, 分别被 jpOne 和 jpTwo 容器占用。之所以看不出变化, 是因为 JPanel 是隐性显示的。

图 4-6 更加详尽的说明了 JPanel 的应用情况。窗口应用了两个 JPanel，划分出两个区域，上面的区域应用了 GridLayout 布局管理器，把这个区域划分成了两行两列。下面的区域应用了 BorderLayout 布局处理器。

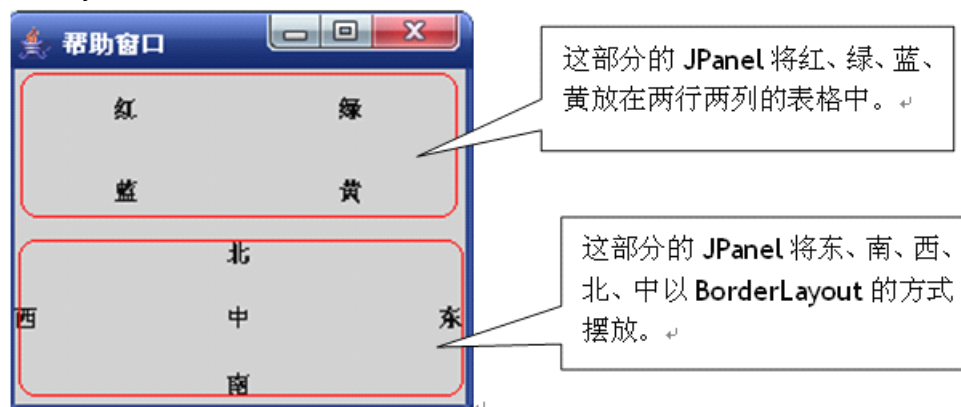


图 4-6 应用了 JPanel 的窗口

3. 分析与提高

- (1) 第 8 行和第 9 行创建了两个 JPanel 容器。
- (2) 第 10 行利用 GridLayout 布局管理器将窗口划分成 2 行 1 列。准备摆放两个 JPanel 容器。
- (3) 第 11 行和第 12 行将两个 JPanel 容器放入窗口中。

4.3.2 使用标签、按钮、复选框和单选框表组件制作调查问卷

在本节中上，在前面任务所学知识的基础上，结合本节的新知识，通过 4 个任务来完成一个调查问卷程序。

【任务 8】 设计调查问卷的布局，并加入一个醒目的标题。

1. 基础知识

在窗口中经常会出现一些说明或提示性的信息，这些信息大多是文字，有时候也可能是图片。Java 不允许将文字或图片直接显示在容器中。这些文字或图片性质的信息要借助组件显示在窗口中。能够实现这一功能的组件被称做标签。任务中提到的“标题”实际上就是调查问卷的“卷头”，它就可以用标签来实现。

JLabel 是 swing 包中专门创建标签组件的类。

创建标签的方式有几种，这里只介绍常用的四种。

方式一：JLabel 对象名 = new JLabel(String s);

方式二：JLabel 对象名 = new JLabel(String s,int hAlignment);

方式三：JLabel 对象名 = new JLabel(ImageIcon image);

方式四：JLabel 对象名 = new JLabel(ImageIcon image,int hAlignment);

方式一和方式二是显示文字性信息时应用的标签。参数 s 用于存放要显示的信息，参数 hAlignment 是设置所显示的信息的对齐方式。共有五种对齐方式：JLabel.BOTTOM、JLabel.LEFT、JLabel.RIGHT、JLabel.TOP 和 JLabel.CENTER，分别表示底端对齐、左对齐、右对齐、顶端对齐和居中对齐。采用方式一创建的标签，文字的对齐方式默认为左对齐。

方式三和方式四是显示图片信息时应用的标签。参数 `image` 是图片对象的引用。`ImageIcon` 是 Java 处理图片的类。`hAlignment` 是设置所显示的图片的对齐方式。也有五种，同文字采用的对齐式一样。采用方式三创建的标签，图片的对齐方式默认为左对齐。

2. 任务实施

首先要为这个调查问题设计一个布局。如图 4-7 所示。



图 4-7 调查问卷布局

这种布局设计与 `BorderLayout` 布局管理器所对应的布局形式有些接近。如图 4-8 所示。

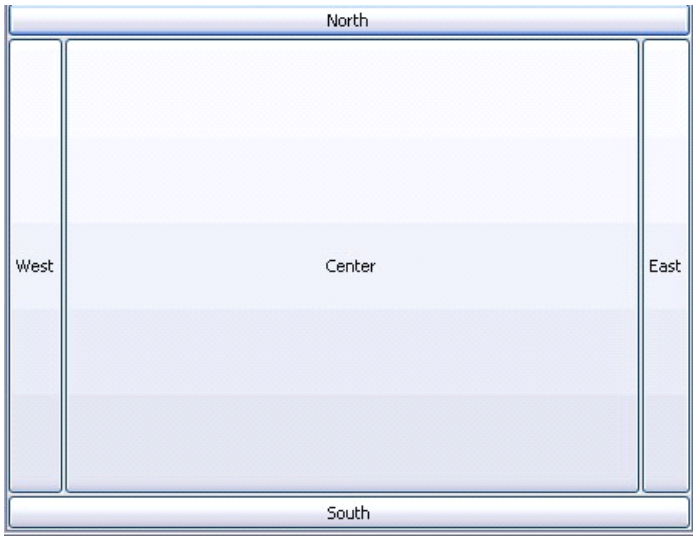


图 4-8 `BorderLayout` 布局管理器布局形式

调查问卷的标题部分可以放在“North”，调查问卷内容部分放在“Center”，提交按钮可以放在“South”。“West”和“East”舍弃不用。有的读者可能会想到用 `GridLayout` 布局管理器。从调查问卷的布局设计来看，是典型的 n 行 1 列。但是，用 `BorderLayout` 布局管理器的显示效果要好于 `GridLayout` 布局管理器。

布局设计好后，就要解决如何加入一个标题的问题。将调查问卷的标题设计成“基本情况调查表”的字符串。加入到窗口后的效果如图 4-9 所示。

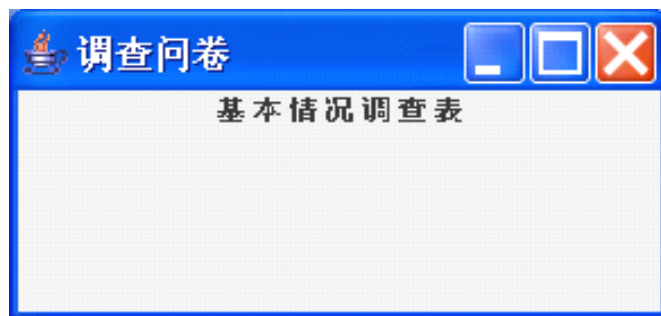


图 4-9 添加了标签的调查问卷

本任务的代码内容如下：

1	<code>import java.awt.*;</code>
2	<code>import javax.swing.*;</code>
3	<code>public class Question</code>
4	<code>{ JFrame win = new JFrame("调查问卷");</code>
5	<code> Container contentPane = win.getContentPane();</code>
6	<code> JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);</code>
7	<code>public void go(){</code>
8	<code> contentPane.setLayout(new BorderLayout());</code>
9	<code> contentPane.add(questionTest,BorderLayout.NORTH);</code>
10	<code> win.setSize(400,300);</code>
11	<code> win.setVisible(true);</code>
12	<code>}</code>
13	<code>public static void main(String arg[]){</code>
14	<code> Question be = new Question ();</code>
15	<code> be.go(); }</code>
16	<code>}</code>

3. 分析与提高

(1) 第 5 行要特别说明一下，这条语句将在以后的大部分程序中出现。Java 在处理图形用户界面时，并不是将组件直接摆放在 JFrame 容器中，而是在 JFrame 容器上再加一个隐性层，组件是放在这个隐性层上的。这个隐性层是由 Container 类创建的。这个类归属于 awt 包。这个隐性层采用第 5 行的形式创建，其中 contentPane 就是这个隐性层的引用。

前面的章节曾用吃饭的例子来说明容器、组件和布局管理器三者之间的关系，却没有提到隐性层。这个隐性层相当于“桌布”的作用。为桌子加桌布的原因是要保护桌子，而为窗口加“桌布”的原因是为了要实现更多的功能。因为 Container 包含很多功能，这些功能 JFrame 不具备。

这样一来，JFrame 创建了一个窗口（桌子），然后，通过第 5 行来为这个窗口加上隐性层（桌布）。所有的组件（菜）都是放在这张“桌布”上的。布局管理器也是针对这张“桌布”设置的。

那么，窗口的大小和窗口显示属性是不是也要对这张“桌布”进行操作呢？这实际上不是这样的。

这两个设置还是直接对窗口来操作。这用吃饭的例子还是可以讲得通的。设置“桌布”

的大小有什么意义呢？桌子（窗口）的大小才是决定能放下多少菜（组件）的关键。因为“桌布”只是一个附属品，它随着桌子（窗口）的存在而存在。

(2) 第 6 行就是这个任务的关键所在。这一行创建了一个标签，标签的名字为 questionTest。将“基本情况调查表”作为标签的内容，并居中对齐。

(3) 第 8 行将 containPane 指向的隐性层，即“桌布”的布局管理器设置为 BorderLayout。

(4) 第 9 行是将 questionTest 标签加入到窗口的上北位置。

【任务 9】 在调查问卷中加入个人体育爱好的调查项。

1. 基础知识

一个人的体育爱好可能有一种，也可能是多种，因此，这部分调查内容应该是可以多选的。复选框组件可以帮助我们完成这一任务。复选框的主要功能就是使用户可以进行多项选择。负责复选框功能的类是 JCheckBox。

常用的创建复选框方式有如下几种：

方式一：JCheckBox 对象名 = new JCheckBox(String s);


方式二：JCheckBox 对象名 = new JCheckBox(String s, boolean b);

方式三：JCheckBox 对象名 = new JCheckBox(ImageIcon image);

方式四：JCheckBox 对象名 = new JCheckBox(ImageIcon image, boolean b);

方式一和方式二创建的复选框显示的信息是字符串。其中参数 s 用于存放显示的字符串。方式二中的参数 b 是说明创建的复选框是否为默认选中。如下面语句：

```
JCheckBox c1 = new JCheckBox("足球",true)
```

这种方式的显示结果为  足球，参数中的“true”是表明这个选项是默认选项。如果去

掉这个参数，即 JCheckBox c1 = new JCheckBox("足球");，那么显示的结果为  足球。

方式三和方式四创建的复选框显示的信息是图片。其中参数 image 是图片的引用。方式四中的参数 b 同方式三中的参数 b 作用相同。

2. 任务实施

我们要将这部分调查内容放在窗口的“Center”部分。因为，调查项不只一项，所以，要事先安排“Center”部分的布局格式，才能将这些调查项逐一摆放。

如果要对“Center”部分进行布局设置，就必须在此处加入一个容器。这时就用到任务 7 中讲到的 JPanel 了。将这个用 JPanel 建立的容器命名为“p”，并将其布局管理局设置为 GridLayout。这样，只要根据调查项的条数来设置 GridLayout 的行数就可以了，而列数肯定是 1。

因为每一个调查项都由调查的问题和复选框组成，所以也存在布局的问题。处理方法就是为每一个调查项也创建一个 JPanel 容器，布局管理器同样设置为 GridLayout 型。调查项容器命名为“p1”，其他调查项容器依次命名为“p2”，“p3”……

布局设计好后，就要将调查项加入到窗口的“Center”部分。添加复选框后的窗口效果如图 4-10 所示。

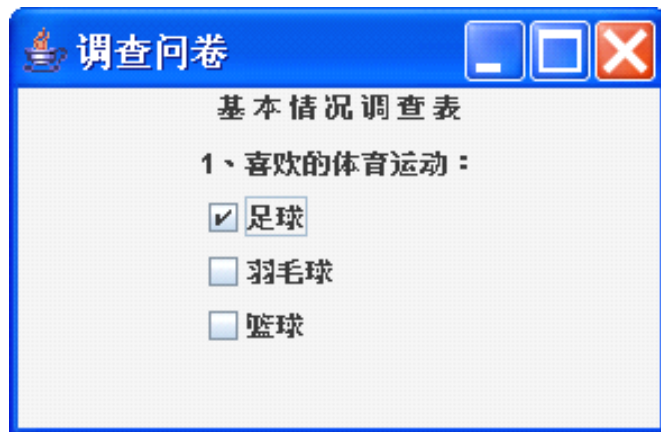


图 4-10 加入调查项后的窗口

从图 4-10 可以看出，关于体育运动的调查项实际由一个标签和三个复选框组成。程序源代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Question
4	{ JFrame win = new JFrame("调查问卷");
5	Container contentPane = win.getContentPane();
6	JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);
7	JLabel JLabel1= new JLabel("1、喜欢的体育运动：",JLabel.LEFT);
8	JCheckBox c1 = new JCheckBox("足球",true);
9	JCheckBox c2 = new JCheckBox("羽毛球");
10	JCheckBox c3 = new JCheckBox("篮球");
11	JPanel p = new JPanel();
12	JPanel p1 = new JPanel();
13	public void go(){
14	contentPane.setLayout(new BorderLayout());
15	//p.setLayout(new GridLayout(?,1));
16	p1.setLayout(new GridLayout(4,1));
17	p1.add(JLabel1);
18	p1.add(c1);p1.add(c2);p1.add(c3);
19	p.add(p1);
20	contentPane.add(questionTest,BorderLayout.NORTH);
21	contentPane.add(p,BorderLayout.CENTER);
22	win.setSize(400,300);
23	win.setVisible(true);
24	}
25	public static void main(String arg[]){
26	Question be = new Question ();
27	be.go(); }

3. 分析与提高

- (1) 第 7 行创建的标签是调查项的问题
- (2) 第 8 行至第 10 行创建了三个复选框，其中“足球”项为默认选项。
- (3) 第 11 行和第 12 行分别创建了用于“Center”部分的 JPanel 容器 p 和用于摆放调查项的 JPanel 容器 p1。
- (4) 第 15 行是注释行，这一行的作用是设置 p 的布局管理器为 GridLayout。目前只有一个调查项，还不能确定最终会有几个调查项，所以不能确定到底要用到几行，因此，这行被注释掉，等确定好共有几个调查项后，再把注释取消。
- (5) 第 16 行设置了“p1”的布局管理器为 GridLayout，并设置为 4 行 1 列。第 1 行用于摆放调查项问题，其他 3 行用于摆放复选框。
- (6) 第 17 行和第 18 行是把内容为“1、喜欢的体育运动：”的标签和三个复选框加入到 p1 容器中。
- (7) 第 19 行是将含有调查项内容的 p1 容器加入到 p 容器中。
- (8) 第 21 行是将 p 容器加入到窗口的“Center”部分。

【任务 10】 在调查问卷中，加入关于被调查人性别的调查项。

1. 基础知识

单选框就是只能选择选项中的一个。性别选项恰恰就可以用单选框来实现。JRadioButton 是专门处理单选框的类。

常用的创建单选框方式有如下几种：

方式一：JRadioButton 对象名 = new JRadioButton(String s);

方式二：JRadioButton 对象名 = new JRadioButton(String s, boolean b);

方式三：JRadioButton 对象名 = new JRadioButton(ImageIcon image);


方式四：JRadioButton 对象名 = new JRadioButton(ImageIcon image, boolean b);

创建单选框的这四种方式与创建复选框的方式含义基本一样，所以不再进行解释了。

下面的语句是采用方式二创建一个单选框。

```
JRadioButton r1 = new JRadioButton("男",true);
```

参数中的“true”是表明这个选项是默认选项。这句话的结果显示为  男，如果去掉

“true”参数，显示结果为  男。

与复选框不同的是，无论有多少选项，只能有一个选项被选中。因此，单选框要有另外一个类的配合，才能达到“单选”目的。这个类是 ButtonGroup。

对于单选框只能选择其中的一项的特性，Java 实际上是这样操作的：若干个选项组成一组，在这一组中只能选择一个选项。这个将若干选项划定为一组的工作就由 ButtonGroup 类来完成。它的创建方式为 ButtonGroup 对象名 = new ButtonGroup()。

比如有 r1 和 r2 两个单选框，利用下面的语句，就可以将这两个单选框划定为一组。

```
ButtonGroup 对象名.add(r1);
```

```
ButtonGroup 对象名.add(r2);
```

这样一来，系统就认定 r1 和 r2 是属于一组，在进行操作时，只能有一个选项被选中。

2. 任务实施

在布局的设计方面，任务 9 已经说的很清楚。新的调查项的布局方式按照上一个调查项的布局进行设计就可以了。因为要加入新的调查项，所以容器 p 的布局应该为 2 行 1 列。如图 4-11 所示。

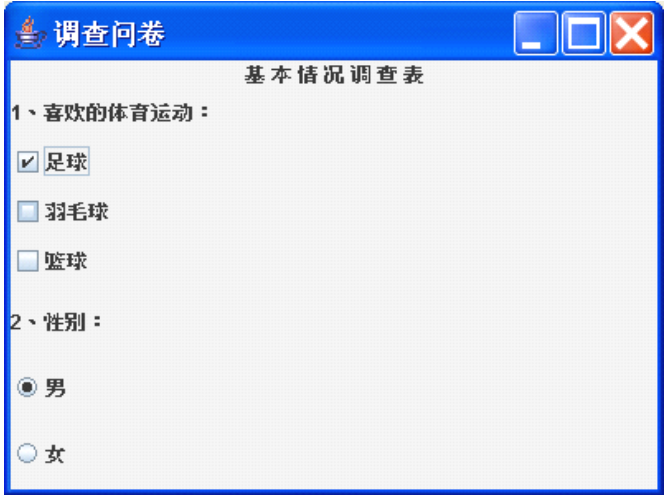


图 4-11 加入“性别”调查项后的窗口

本任务的代码内容如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Question
4	{ JFrame win = new JFrame("调查问卷");
5	Container contentPane = win.getContentPane();
6	JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);
7	JLabel JLabel1= new JLabel("1、喜欢的体育运动： ",JLabel.LEFT);
8	JCheckBox c1 = new JCheckBox("足球",true);
9	JCheckBox c2 = new JCheckBox("羽毛球");
10	JCheckBox c3 = new JCheckBox("篮球");
11	JLabel JLabel2= new JLabel("2、性别： ",JLabel.LEFT);
12	ButtonGroup bgroup = new ButtonGroup();
13	JRadioButton r1 = new JRadioButton("男",true);
14	JRadioButton r2 = new JRadioButton("女");
15	JPanel p = new JPanel();
16	JPanel p1 = new JPanel();
17	JPanel p2 = new JPanel();
18	public void go(){
19	contentPane.setLayout(new BorderLayout());
20	p.setLayout(new GridLayout(2,1));
21	p1.setLayout(new GridLayout(4,1));
22	p2.setLayout(new GridLayout(3,1));
23	bgroup.add(r1);bgroup.add(r2);
24	p1.add(JLabel1);

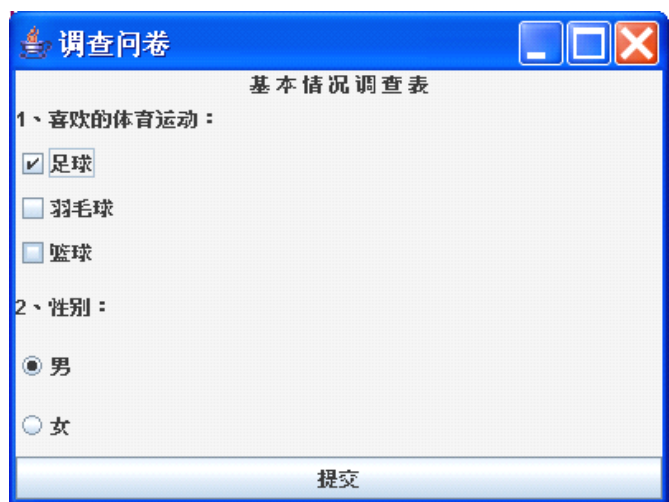


图 4-12 直接摆放“提交”按钮的窗口

显然，直接摆放的做法使得显示效果很不理想。如果想改变这种效果就必须创建一个 JPanel 容器，将按钮放入容器中，然后将容器放到窗口的“South”部分。JPanel 的默认布局管理器是 FlowLayout, 这个布局管理器默认的对齐方式是居中对齐。所以，只要采用 JPanel 的默认布局管理器，就可以使“提交”按钮在窗口下端居中显示。添加“提交”按钮后的窗口效果如图 4-13 所示。

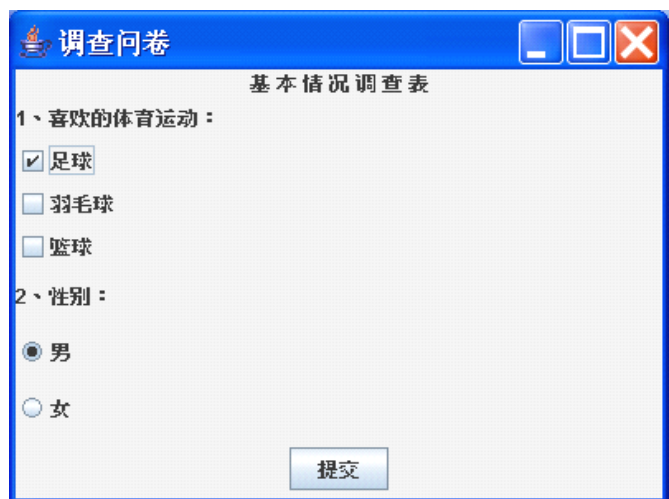


图 4-13 加入“提交”按钮后的窗口

本任务的代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Question
4	{ JFrame win = new JFrame("调查问卷");
5	Container contentPane = win.getContentPane();
6	JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);
7	JLabel JLabel1= new JLabel("1、喜欢的体育运动：",JLabel.LEFT);
8	JCheckBox c1 = new JCheckBox("足球",true);
9	JCheckBox c2 = new JCheckBox("羽毛球");
10	JCheckBox c3 = new JCheckBox("篮球");

11	JLabel JLabel2= new JLabel("2、性别: ",JLabel.LEFT);
12	ButtonGroup bgroup = new ButtonGroup();
13	JRadioButton r1 = new JRadioButton("男",true);
14	JRadioButton r2 = new JRadioButton("女");
15	JButton tiJiao = new JButton("提交");
16	JPanel p = new JPanel();
17	JPanel p1 = new JPanel();
18	JPanel p2 = new JPanel();
19	JPanel p3 = new JPanel();
20	public void go(){
21	contentPane.setLayout(new BorderLayout());
22	p.setLayout(new GridLayout(2,1));
23	p1.setLayout(new GridLayout(4,1));
24	p2.setLayout(new GridLayout(3,1));
25	bgroup.add(r1);bgroup.add(r2);
26	p1.add(JLabel1);
27	p1.add(c1);p1.add(c2);p1.add(c3);
28	p2.add(JLabel2);
29	p2.add(r1);p2.add(r2);
30	p3.add(tiJiao);
31	p.add(p1);p.add(p2);
32	contentPane.add(questionTest,BorderLayout.NORTH);
33	contentPane.add(p,BorderLayout.CENTER);
34	contentPane.add(p3,BorderLayout.SOUTH);
35	win.setSize(400,300);
36	win.setVisible(true);
37	}
38	public static void main(String arg[]){
39	Question be = new Question ();
40	be.go(); }
41	}

3. 分析与提高

- (1) 第 15 行创建了名字为 tiJiao 的按钮。按钮所显示的文字为“提交”。
- (2) 第 19 行创建了用于加载按钮的容器 p3。
- (3) 第 30 行将按钮加入以 p3 容器中。
- (4) 第 34 行将容器 p3 加入到窗口的“South”部分。

4.3.3 向调查问卷中加入下拉列表

本节将在已经创建的调查问卷中加入新的调查项，这个调查项的表现形式是下拉列表。

【任务 12】在调查问卷中增加调查者出生年份的调查项。

1. 基础知识

本调查问卷所调查的对象年龄在 12-32 岁之间，所以这个调查项应该具有 21 个选项。因为选项是固定的，所以采用下拉列表的方式让用户进行选择最为方便。

下拉列表就是其选项可以伸缩的列表，如 Microsoft Word 中字体、字号的选择就是下拉列表。

下面的语句是最常用的创建下拉列表的方法。

```
JComboBox 对象名 = new JComboBox(Object[]);
```

其中参数 Object[] 表示对象数组。因为下拉列表要显示是一组数据，所以采用数组的形式传递参数是再合适不过的了。但是，下拉列表要显示的数据的类型并不确定，所以采用 Object 作为数组的类型，使其可以接收任务数据类型的数据。一般情况下，采用的是 String 类型的数组来存储下拉列表所包含的内容。例如

```
String brand[] = {"雪佛兰","福特","大众","马自达","本田","丰田"};
```

```
JComboBox carBrand = new JComboBox(brand);
```

这两条语句可以建立以汽车品牌为内容的下拉列表。

2. 任务实施

新添加的年份调查项由调查项问题和下拉列表组成。同前面的任务一样，还是采用 JPanel 容器摆放调查项，然后再将容器摆放到窗口中的 p 容器中。年份调查项的容器布局管理器采用 FlowLayout，而不采用 GridLayout 布局管理器。这主要出于显示效果的考虑，对于本任务的情况来说，FlowLayout 比 GridLayout 布局管理器的显示效果要好。JPanel 的默认布局管理器就是 FlowLayout，但是，默认的对齐方式是居中对齐。这与前两项调查项的对齐方式不一致。因此，要把 FlowLayout 布局管理器的对齐方式设置成左对齐。

添加年份调查项后的窗口显示效果如图 4-14 所示。

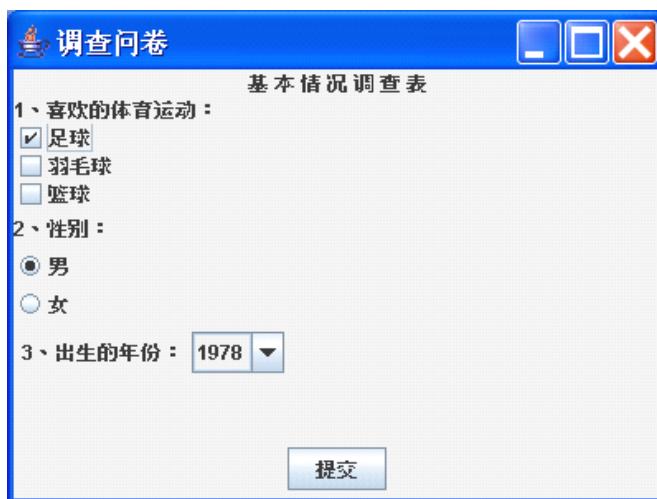


图 4-14 加入下拉表后的窗口

程序代码内容为：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Question
4	{ JFrame win = new JFrame("调查问卷");
5	Container contentPane = win.getContentPane();
6	JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);

7	JLabel JLabel1= new JLabel("1、喜欢的体育运动: ",JLabel.LEFT);
8	JCheckBox c1 = new JCheckBox("足球",true);
9	JCheckBox c2 = new JCheckBox("羽毛球");
10	JCheckBox c3 = new JCheckBox("篮球");
11	JLabel JLabel2= new JLabel("2、性别: ",JLabel.LEFT);
12	ButtonGroup bgroup = new ButtonGroup();
13	JRadioButton r1 = new JRadioButton("男",true);
14	JRadioButton r2 = new JRadioButton("女");
15	JLabel JLabel3 = new JLabel("3、出生的年份: ",JLabel.LEFT);
16	String year[] = {"1978","1979","1980","1981","1982","1983","1984",
17	"1985","1986","1987","1988","1989","1990","1991",
18	"1992","1993","1994","1995","1996","1997","1998"};
19	JComboBox jcmb = new JComboBox (year);
20	JButton tiJiao = new JButton("提交");
21	JPanel p = new JPanel();
22	JPanel p1 = new JPanel();
23	JPanel p2 = new JPanel();
24	JPanel p3 = new JPanel();
25	JPanel p4 = new JPanel();
26	public void go(){
27	contentPane.setLayout(new BorderLayout());
28	p.setLayout(new GridLayout(3,1));
29	p1.setLayout(new GridLayout(4,1));
30	p2.setLayout(new GridLayout(3,1));
31	p4.setLayout(new FlowLayout(FlowLayout.LEADING));
32	bgroup.add(r1);bgroup.add(r2);
33	p1.add(JLabel1);
34	p1.add(c1);p1.add(c2);p1.add(c3);
35	p2.add(JLabel2);
36	p2.add(r1);p2.add(r2);
37	p3.add(tiJiao);
38	p4.add(JLabel3);p4.add(jcmb);
39	p.add(p1);p.add(p2);p.add(p4);
40	contentPane.add(questionTest,BorderLayout.NORTH);
41	contentPane.add(p,BorderLayout.CENTER);
42	contentPane.add(p3,BorderLayout.SOUTH);
43	win.setSize(400,300);
44	win.setVisible(true);
45	}
46	public static void main(String arg[]){
47	Question be = new Question ();
48	be.go(); }
49	}

3. 分析与提高

- (1) 第 15 行创建调查问题标签 JLabel3，标签内容为“3、出生的年份：”，对齐方式的左对齐。
- (2) 第 16 行至第 18 行创建了 String 类型的数组 year，它是下拉表要包含的年份内容。
- (3) 第 19 行创建了以 year 数组为参数的下拉列表 jcmb。
- (4) 第 25 行创建了用于摆放年份调查项的 JPanel 容器 p4。
- (5) 由于添加了调查项，所以要把 p 窗口的行数增加到 3 行。见第 28 行。
- (6) 第 31 行设置了 p4 容器的布局管理器，并将对齐方式设置为左对齐。
- (7) 第 38 行将年份调查项问题和下拉列表添加到 p4 容器中。
- (8) 第 39 行将 p4 容器加入到 p 容器中。

4.3.4 在调查问卷中加入文本输入区域—文本框和文本域

在本节中将主要对文本框和文本域的使用进行介绍。

【任务 13】在调查问卷中加入调查者联系方式的调查项。

1. 基础知识

调查者的联系方式指的是电话或电子邮件。无论是哪一种，都需要用户进行输入。最常用的可供用户输入信息的就是文本框。比如，用户登录某系统时要输入用户名和密码等信息，所应用的就是文本框。swing 包中的 JTextField 类是用于创建文本框的类。

创建文本框的方式有以下几种：

方式一：JTextField 对象名 = new JTextField();

方式二：JTextField 对象名 = new JTextField(String s);

方式三：JTextField 对象名 = new JTextField(int i);

方式四：JTextField 对象名 = new JTextField(String s, int i);

参数 i 是文本框的宽度。这个宽度指输入的字段长度。参数 s 是文本框的初始内容，即在第一次显示时，文本框里的内容。如果采用方式一或方式三，则文本框的初始内容为空。采用方式一创建的文本框宽度为 0。采用方式二创建的文本框宽度即 s 字符串的字段长度。

其实，文本框的初始宽度与其所处的布局管理器有关。在 FlowLayout 布局管理器下，四种创建方式的文本框宽度都能如实表现出来。但是在 BorderLayout 和 GridLayout 布局管理器下，这些宽度设置就会失效。比如，用方式一创建的文本框的宽度应该是 0。但是，在 BorderLayout 布局管理器下，文本框会充满其所在的区域，宽度与其所在区域的宽度一致。所以，在应用时要特别注意文本框的这一特性。

2. 任务实施

联系方式调查项的同年份调查项的处理方式完全一样，布局管理器采用 FlowLayout，并设置为左对齐。

加入联系方式调查项的窗口显示效果如图 4-15 所示。

调查问卷

基本情况调查表

1、喜欢的体育运动：

☒ 足球

☐ 羽毛球

☐ 篮球

2、性别：

☒ 男

☐ 女

3、出生的年份： 1978 ▼

4、联系方式：

提交

图 4-15 加入文本框后的窗口

本任务的代码内容如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Question
4	{ JFrame win = new JFrame("调查问卷");
5	Container contentPane = win.getContentPane();
6	JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);
7	JLabel JLabel1= new JLabel("1、喜欢的体育运动： ",JLabel.LEFT);
8	JCheckBox c1 = new JCheckBox("足球",true);
9	JCheckBox c2 = new JCheckBox("羽毛球");
10	JCheckBox c3 = new JCheckBox("篮球");
11	JLabel JLabel2= new JLabel("2、性别： ",JLabel.LEFT);
12	ButtonGroup bgroup = new ButtonGroup();
13	JRadioButton r1 = new JRadioButton("男",true);
14	JRadioButton r2 = new JRadioButton("女");
15	JLabel JLabel3 = new JLabel("3、出生的年份： ",JLabel.LEFT);
16	String year[] = {"1978","1979","1980","1981","1982","1983","1984",
17	"1985","1986","1987","1988","1989","1990","1991",
18	"1992","1993","1994","1995","1996","1997","1998"};
19	JComboBox jcmb = new JComboBox (year);
20	JLabel JLabel4 = new JLabel("4、联系方式： ",JLabel.LEFT);
21	JTextField txt = new JTextField(10);
22	JButton tiJiao = new JButton("提交");
23	JPanel p = new JPanel();
24	JPanel p1 = new JPanel();
25	JPanel p2 = new JPanel();
26	JPanel p3 = new JPanel();
27	JPanel p4 = new JPanel();
28	JPanel p5 = new JPanel();

种情况的组件，即文本域 `JTextArea`。这种文本输入区域不同于文本框，它可以设置区域的行数，可显示更多的文本内容。

创建文本域的方式有以下几种：

方式一：`JTextArea` 对象名 = `new JTextField()`;

方式二：`JTextArea` 对象名 = `new JTextField(String s)`;

方式三：`JTextArea` 对象名 = `new JTextField(int rows,int columns)`;

方式四：`JTextArea` 对象名 = `new JTextField(Sting s,int rows,int columns)`;

参数 `s` 是文本域中要显示的初始内容。`rows` 和 `columns` 是文本域的行数和列数。同文本框一样，文本域的行数和列数与其所处的布局管理器有关。在特定的布局管理器中也会产生行数和列数失效的情况。

2. 任务实施

新加入的调查项仍然采用 `FlowLayout` 的布局方式，并且采用左对齐。目的是使文本域的行数和列数能够正常显示。

加入新调查项的窗口显示效果如图 4-16 所示。

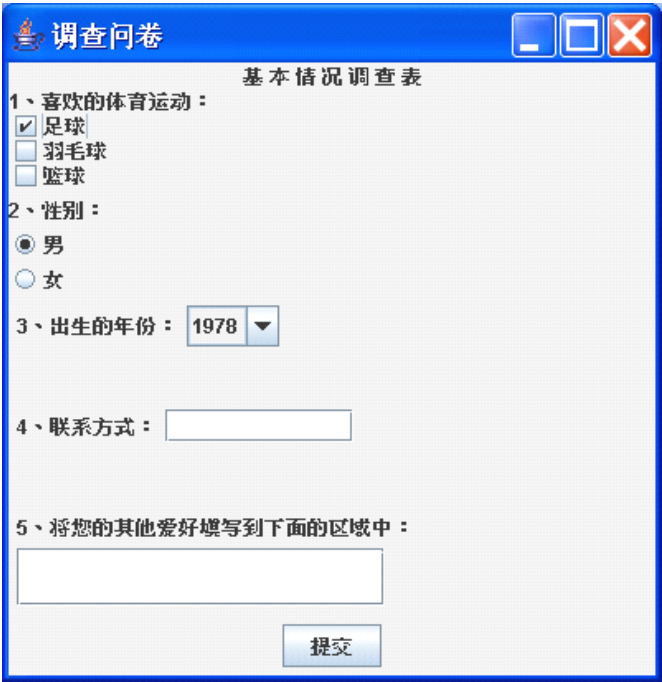


图 4-16 加入文本域后的窗口

本任务的代码内容如下：

1	<code>import java.awt.*;</code>
2	<code>import javax.swing.*;</code>
3	<code>public class Question</code>
4	<code>{ JFrame win = new JFrame("调查问卷");</code>
5	<code> Container contentPane = win.getContentPane();</code>
6	<code> JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);</code>
7	<code> JLabel JLabel1= new JLabel("1、喜欢的体育运动：",JLabel.LEFT);</code>
8	<code> JCheckBox c1 = new JCheckBox("足球",true);</code>
9	<code> JCheckBox c2 = new JCheckBox("羽毛球");</code>

10	JCheckBox c3 = new JCheckBox("篮球");
11	JLabel JLabel2= new JLabel("2、性别：",JLabel.LEFT);
12	ButtonGroup bgroup = new ButtonGroup();
13	JRadioButton r1 = new JRadioButton("男",true);
14	JRadioButton r2 = new JRadioButton("女");
15	JLabel JLabel3 = new JLabel("3、出生的年份：",JLabel.LEFT);
16	String year[] = {"1978","1979","1980","1981","1982","1983","1984",
17	"1985","1986","1987","1988","1989","1990","1991",
18	"1992","1993","1994","1995","1996","1997","1998"};
19	JComboBox jcmb = new JComboBox (year);
20	JLabel JLabel4 = new JLabel("4、联系方式：",JLabel.LEFT);
21	JTextField txt = new JTextField(10);
22	JLabel JLabel5 = new JLabel("5、将您的其他爱好填写到下面的区域中：",JLabel.LEFT);
23	JTextArea txtArea = new JTextArea(4,20);
24	JButton tiJiao = new JButton("提交");
25	JPanel p = new JPanel();
26	JPanel p1 = new JPanel();
27	JPanel p2 = new JPanel();
28	JPanel p3 = new JPanel();
29	JPanel p4 = new JPanel();
30	JPanel p5 = new JPanel();
31	JPanel p6 = new JPanel();
32	public void go(){
33	contentPane.setLayout(new BorderLayout());
34	p.setLayout(new GridLayout(5,1));
35	p1.setLayout(new GridLayout(4,1));
36	p2.setLayout(new GridLayout(3,1));
37	p4.setLayout(new FlowLayout(FlowLayout.LEADING));
38	p5.setLayout(new FlowLayout(FlowLayout.LEADING));
39	p6.setLayout(new FlowLayout(FlowLayout.LEADING));
40	bgroup.add(r1);bgroup.add(r2);
41	p1.add(JLabel1);
42	p1.add(c1);p1.add(c2);p1.add(c3);
43	p2.add(JLabel2);
44	p2.add(r1);p2.add(r2);
45	p3.add(tiJiao);
46	p4.add(JLabel3);p4.add(jcmb);
47	p5.add(JLabel4);p5.add(txt);
48	p6.add(JLabel5);p6.add(txtArea);
49	p.add(p1);p.add(p2);p.add(p4);p.add(p5);p.add(p6);
50	contentPane.add(questionTest,BorderLayout.NORTH);
51	contentPane.add(p,BorderLayout.CENTER);
52	contentPane.add(p3,BorderLayout.SOUTH);

53	<code>win.setSize(400,400);</code>
54	<code>win.setVisible(true);</code>
55	<code>}</code>
56	<code>public static void main(String arg[]){</code>
57	<code> Question be = new Question ();</code>
58	<code> be.go(); }</code>
59	<code>}</code>

3. 分析与提高

(1) 第 22 行创建包含调查项问题的标签 JLabel5，内容为“5、将您的其他爱好填写到下面的区域中：”，对齐方式为左对齐。

(2) 第 23 行创建了文本域 txtArea，文本域为 4 行，20 列。

(3) 第 31 行创建了用于摆放新调查项的 JPanel 容器 p6。

(4) 由于添加了新调查项，所以要把 p 窗口的行数增加到 5 行。见第 34 行。

(5) 第 39 行设置了 p6 容器的布局管理器，并将对齐方式设置为左对齐。

(6) 第 48 行将新调查项的问题和文本域添加到 p6 容器中。

(7) 第 49 行将 p6 容器加入到 p 容器中

通常情况下，文本域的创建并不是到此为止的。在应用中，如果用户输入的内容大于 4 行，20 列能显示的内容，文本域还是没法显示多出来的内容。有一个可以和文本域组件配合使用的类，它的名字是 JScrollPane。这个类的作用是将文本域加上滚动条功能。如果用户输入的内容大于文本域的行数或列数，则会自动在文本域的底部或右侧出现滚动条。

对于上面的程序可以做如下修改，就可以达到为文本域添加滚动条的功能。

创建 JScrollPane 对象：JScrollPane scrollPane = new JScrollPane(txtArea)。其中的 txtArea 就是第 23 行创建的文本域。参数的作用就是要让 JScrollPane 类知道到底为谁加滚动条功能。这句话可以加在第 23 行之后。

将第 48 行的 p6.add(txtArea) 更改为 p6.add(scrollPane)。

完成以上修改后，就可以实现文本域的滚动条功能了。

4.3.5 学习 JTable 组件的使用

当我们进行数据统计时，如果采用表格的话，可以将数据更有条理地显示在用户界面中。swing 包中的 JTable 类专门用于表格的创建。本节将重点介绍 JTable 组件的使用。

【任务 15】制作一张成绩单。

1. 基础知识

在进行数据处理的软件中，经常使用的数据显示方式就是表格。这种数据结构简洁，一目了然。swing 包中的 JTable 类可以完成表格的创建。常用创建表格的方式为：

JTable 对象名=new JTable(Object[][] rowData,Object[] columnNames);

两个参数均为数组。其中有 rowData 是二维数组，用于存放表格中的记录。columnNames 是一维数组，用于存放表头，即列的名称。

2. 任务实施

设计一个成绩单，其中表头为：姓名、语文、数学、总分、及格和作弊。其中及格和作弊用布尔值表示。

本任务的代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class TableEx {
4	JFrame win = new JFrame("成绩单");
5	Container contentPane = win.getContentPane();
6	public void go(){
7	Object[][] jilu = {
8	{"阿三", new Integer(66),new Integer(32), new Integer(98), new Boolean(false),new Boolean(false)},
9	{"阿四", new Integer(85), new Integer(69), new Integer(154), new Boolean(true),new Boolean(false)}
10	};
11	String[] lieming = {"姓名","语文","数学","总分","及格","作弊"};
12	JTable table = new JTable(jilu, lieming);
13	JScrollPane scrollPane = new JScrollPane(table);
14	contentPane.add(scrollPane, BorderLayout.CENTER);
15	win.pack();
16	win.setVisible(true);
17	}
18	public static void main(String args[]) {
19	TableEx be = new TableEx();
20	be.go();
21	}
22	}

程序的执行结果为如图 4-17 所示：



姓名	语文	数学	总分	及格	作弊
阿三	66	32	98	false	false
阿四	85	69	154	true	false

图 4-17 使用表格的成绩单

3. 分析与提高

- (1) 第 4 行和第 5 行创建了窗口 win 和它的隐性层。
- (2) 第 7 行至第 10 行定义了表格的记录数组 jilu。注意，这个数组的类型为 Object 型，所以，其中的元素必须使用包裹类类型。
- (3) 第 11 行定义了表格的列名数组 lieming。
- (4) 第 12 行创建了表格 table。将 jilu 和 lieming 作为其参数。
- (5) 第 13 行定义的是一个带滚动条的 Panel，将 table 放入其中。如果表格内容过多，窗口会显示不下，加入滚动条后，会使窗口更美观，实用。
- (6) 第 14 行将装载了表格的 scrollPane，以 BorderLayout 布局方式摆放在窗口的中央位置。
- (7) 窗口大小没有采用 setSize()方法。而是采用 pack()方法。这种方法会根据组件大小设置窗口大小。但是，如果使用了 JScrollPane，窗口高度会超出摆放的组件的高度。

4.3.5 制作计算器的菜单

本节中将介绍菜单的使用方法。

【任务 16】为计算器制作一个菜单。

1. 基础知识

菜单在图形用户界面中经常被使用。如 Microsoft word 的用户界面。菜单可以将各种操作集中起来，方便用户的操作。从显示效果来看，即美观又节省空间。

无论什么样的菜单，都是由三部分组成的，分别是菜单条、菜单和菜单项。如图 4-18 所示。

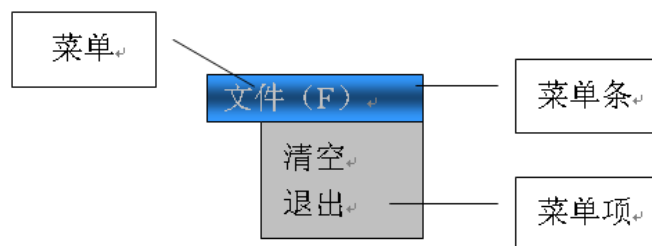


图 4-18 菜单的组成

菜单条的作用就是摆放各种功能菜单，对这些菜单进行整合。“文件(F)”、“编辑(E)”等被称为功能菜单，这些功能菜单其实是功能组。每个功能菜单下都有若干的功能，即菜单项。比如，通常“文件”菜单中会包含“打开”、“保存”、“另存为”等菜单项。在 Swing 中，菜单的三个组成部分分别对应三个类，即 JMenuBar、JMenu 和 JMenuItem。

(1) JMenuBar 类

JMenuBar 用于创建整合菜单的菜单条。下面的语句是最常用的创建菜单条的方式：

JMenuBar 对象名 = new JMenuBar();

JMenuBar 在添加功能菜单时采用的是 add(Component c)方法，参数 c 用于传递菜单对象。

菜单条组件与其他常用组件有一点不同。其他常用组件是摆放在由 Container 类创建的隐性层上的。而菜单条必须摆放在窗口中，而且不用 add(Component c)方法。JFrame 采用专用的添加菜单条方法摆放菜单条。这个方法是 setJMenuBar(JMenuBar mbar)。下面的语句是将 MBar 菜单条添加到 win 所指向的窗口中。

win. setJMenuBar(JMenuBar MBar);

(2) JMenu 类

JMenu 用于创建功能菜单。创建菜单的方式有如下两种：

方式一：JMenu 对象名 = new JMenu();

方式二：JMenu 对象名 = new JMenu(String s);

参数 s 是菜单要显示的信息。如“文件(F)”菜单的创建，其中的文字信息就要存放在参数 s 中。方式一所创建的菜单不显示任何文字信息。

JMenu 类拥有一些常用方法。

- add(Component c)

这个方法的作用是将菜单项加入到菜单中。

- addSeparator()

在菜单项中，有时会出现一些线条，将菜单项分隔成若干组。这个方法就可以在菜单项之间建立这样的分隔线。

(3) JMenuItem 类

JMenuItem 用于创建菜单项。可以通过如下几种方式创建菜单项。

方式一：JMenuItem 对象名 = new JMenuItem();

方式二：JMenuItem 对象名 = new JMenuItem(String s);

方式三：JMenuItem 对象名 = new JMenuItem(ImageIcon image);

方式四：JMenuItem 对象名 = new JMenuItem(String s,ImageIcon image);

参数 s 是菜单项要显示的文字信息。如创建“保存(S)”菜单项，其中的文字信息就要存放在参数 s 中。参数 image 可以为菜单项增加图片，使菜单项更加直观和美观。

采用方式一创建的菜单项为空菜单项。方式四创建的菜单项既包含图片又包含文字信息。

2. 任务实施

计算器由菜单、显示区和按钮区组成。本任务只制作出计算器的菜单和显示区。如图 4-19 所示。



图 4-19 添加了菜单的计算器窗口

本任务的代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class Calculator
4	{ JFrame win = new JFrame("计算器");
5	Container contentPane = win.getContentPane();
6	JTextField jTextField = new JTextField();
7	JMenuBar MBar = new JMenuBar();
8	JMenu file = new JMenu("文件(F)");
9	JMenu help = new JMenu("帮助(H)");
10	JMenuItem qingkong = new JMenuItem("清空");
11	JMenuItem exit = new JMenuItem("退出");
12	JMenuItem about = new JMenuItem("关于...");
13	public void go() {
14	file.add(qingkong);
15	file.add(exit);
16	help.add(about);
17	MBar.add(file);
18	MBar.add(help);
19	contentPane.add(jTextField);
20	win.setJMenuBar(MBar);
21	win.setSize(200,100);
22	win.setVisible(true);

23	}
24	public static void main(String arg[]) {
25	Calculator be = new Calculator ();
26	be.go(); }
27	}

3. 分析与提高

- (1) 第 4 行创建了计算器的窗口，标题为“计算器”。
- (2) 第 5 行创建了计算器窗口的隐性层 contentPane。
- (3) 第 6 行创建了计算器的显示区文本框 jTextField。
- (4) 第 7 行创建了计算器的菜单条 Mbar。
- (5) 第 8 行创建了文件菜单。
- (6) 第 9 行创建了帮助菜单。
- (7) 第 10 行创建了清空菜单项。
- (8) 第 11 行创建了退出菜单项。
- (9) 第 12 行创建了关于菜单项。
- (10) 第 14 行和第 15 行将“清空”菜单项和“退出”菜单项加入到文件菜单。
- (11) 第 16 行将“关于”菜单项加入到帮助菜单。
- (12) 第 17 行和第 18 行将 file 和 help 指向的菜单加入到 MBar 指向的菜单条中。
- (13) 第 19 行将显示区加入到窗口中。

(14) 第 20 行利用 setJMenuBar()方法，将 Mbar 菜单条加入到 win 指向的窗口中。注意，并不是加入到隐性层上。

读者是否注意到，本程序中并没有关于布局管理器的设置。而且，菜单条组件摆放在窗口中，显示区摆放在隐性层。是不是需要对这它们分别设置布局管理器呢？实际上，这样做对于本任务来说是不必要的。

窗口容器的默认布局管理器是 BorderLayout。但是，菜单条有其自身特性，它并不占用窗口的布局。Java 语言的开发者这样设计菜单条的原因可能是因为菜单条通常都是在窗口的上部，所以就将其位置固定下来，而不受布局管理器的控制。因此，在本任务中，关于窗口的布局管理器设置就省略了。

隐性层的布局管理器默认也是 BorderLayout。在 BorderLayout 布局管理器中摆放组件时，如果不对摆放位置进行说明的话，系统默认的位置是“CENTER”。因此，第 19 行语句 contentPane.add(jTextField)与语句 contentPane.add(jTextField, BorderLayout.CENTER)等价。

【任务 17】为计算器添加工具栏

1. 基础知识

工具栏一般出现在菜单条的下方。一些常用操作会出现在工具栏中，使用户操作更加方便。

Swing 采用 JToolBar 类创建工具栏组件。

最常用的创建工具栏的方法是：JToolBar 对象名 = new JToolBar();。

工具栏中整合的实际上是菜单项。这些菜单项是以按钮的形式展现的。例如文件菜单中有“清空”项，工具栏中也有“清空”项，它们完成的操作是一样的。但是，文件菜单中的“清空”是菜单项，是用 JMenuItem 类创建的。而工具栏中的“清空”是按钮，是用 JButton 类创建的。

创建按钮最常用的两种方式已经在任务 11 中讲解过了。如果工具栏采用方式一创建按钮，则工具栏中显示的是文字。如果工具栏采用方式二创建按钮，则工具栏中显示的是图片。本任务中，采用方式一创建工具栏中的按钮。

工具栏和菜单条不同，Java 将其作为普通组件处理。因此，它受布局管理器的控制。从显示效果来看，工具栏的宽度可以随着窗口的变化而变化，但是高度应该保持不变。而且，工具栏一般摆放在菜单条的下方。从这些特性来看，最适合工具栏的布局管理器就是 BorderLayout。可以将工具栏摆放在 BorderLayout.NORTH。

2. 任务实施

加入工具栏后的计算器如图 4-20 所示。

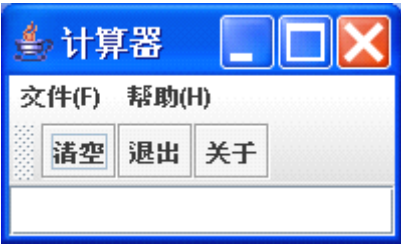


图 4-20 加入工具栏后的计算器窗口

本任务的代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class JToolBarEx
4	{ JFrame win = new JFrame("计算器");
5	Container contentPane = win.getContentPane();
6	JTextField jTextArea = new JTextField(10);
7	JMenuBar MBar = new JMenuBar();
8	JMenu file = new JMenu("文件(F)");
9	JMenuItem qingkong = new JMenuItem("清空");
10	JMenuItem exit = new JMenuItem("退出");
11	JMenu help = new JMenu("帮助(H)");
12	JMenuItem about = new JMenuItem("关于...");
13	JToolBar toolBar = new JToolBar();
14	JButton jb1 = new JButton("清空");
15	JButton jb2 = new JButton("退出");
16	JButton jb3 = new JButton("关于");
17	public void go() {
18	//contentPane.setLayout(new BorderLayout());
19	file.add(qingkong);
20	file.add(exit);
21	help.add(about);
22	MBar.add(file);
23	MBar.add(help);
24	toolBar.add(jb1);
25	toolBar.add(jb2);

26	toolBar.add(jb3);
27	contentPane.add(jTextArea, BorderLayout.CENTER);
28	contentPane.add(toolBar, BorderLayout.NORTH);
29	win.setJMenuBar(MBar);
30	win.setSize(200,120);
31	win.setVisible(true);
32	}
33	public static void main(String arg[]) {
34	JToolBarEx be = new JToolBarEx ();
35	be.go();}
36	}

3. 分析与提高

- (1) 第 13 行创建工具栏 toolBar。
- (2) 第 14 行至第 16 行创建了三个按钮。
- (3) 第 18 行被注释掉的语句是对隐性层 contentPane 进行布局管理器设置。因为，隐性层的默认布局管理器就是 BorderLayout，所以这句话是可有可无的。
- (4) 第 24 行至第 26 行将第 14 行至第 16 行创建的按钮添加到工具栏中。
- (5) 第 28 行将工具栏 toolBar 加入到 contentPane 的 NORTH 位置。

4.3.6 常用对话框

当用户对某系统进行操作时，经常会出现一些提示性的窗口。这些提示性窗口有时是指导用户如何进行下一步操作，有时是提醒用户出现了错误操作。比如某文本框中的内容要求是整数，位数不能大于 8 位，用户如果输入错误就要给出提示。这种提示性窗口被称为对话框。在这一节中就将介绍各种情况下使用的对话框。

Swing 有多个类支持对话框，在这里只介绍 JOptionPane 类。关于这个类，只需要读者了解可以提供用户所需对话框的方法就可以了。另外，这些方法都是静态的，在使用时不必创建 JOptionPane 类的对象，可以直接引用。这些方法的名字都以 showXxxxDialog 的方式出现。其中 Xxxx 可以是：Message、Confirm、Input 和 Option。分别对应信息对话框、验证对话框、输入对话框和自定义对话框。以下任务中，将逐一对这些方法进行介绍。

【任务 18】认识提示信息对话框

1. 基础知识

提示信息对话框在应用上最为广泛。可以通过方法 showMessageDialog()完成提示信息对话框的创建。提示信息对话框最常用的创建方法如下：

```
showMessageDialog(Component c,Object message,String title,int messageType);
```

其中参数 c 表示对话框的上一级窗口。对话框不可能单独存在，它是由用户在操作某系统时产生的，所以要确定上一级窗口。这样做还有一个目的就是：对话框窗口不关闭，用户则无法操作其上一级窗口。这种做法不仅可以保证用户操作系统时的顺序性，有时还可以避免错误操作的升级，造成系统瘫痪。

对话框的上一级窗口通常就是 JFrame 容器。c 即可以是上一级窗口的引用，也可以是上一级窗口的隐性层引用。

参数 message 是提示信息对话框中要显示的信息内容。通过是 String 型的字符串。

参数 title 通常也是 String 型的字符串,它要传递的是提示信息对话框窗口标题栏的内容。
参数 messageType 是指提示信息对话框的类型。这些类型如表 4-2 所示。

表 4-2 提示信息对话框类型表

提示框名称	类型
错误信息	ERROR_MESSAGE
资讯信息	INFORMATION_MESSAGE
警告信息	WARNING_MESSAGE
问题信息	QUESTION_MESSAGE
通用信息	PLAIN_MESSAGE

这些类型在显示时,会有与其类型对应的图片,用来加强信息对话框的可读性。只有通用信息类型的对话框没有图片。

提示信息对话采用的方法是 JOptionPane 类的静态方法。所以在使用时,可以直接采用下面的格式:

```
JOptionPane.showMessageDialog(Component c,Object message,String title,int messageType);
```

五种提示信息对话框如图 4-21、4-22、4-23、4-24、4-25 所示。

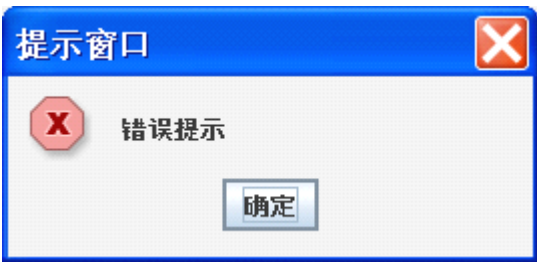


图 4-21 错误信息提示对话框

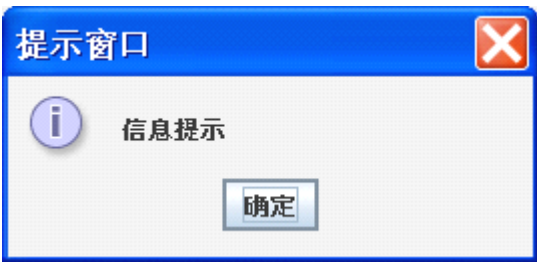


图 4-22 资讯信息提示对话框



图 4-23 警告信息提示对话框

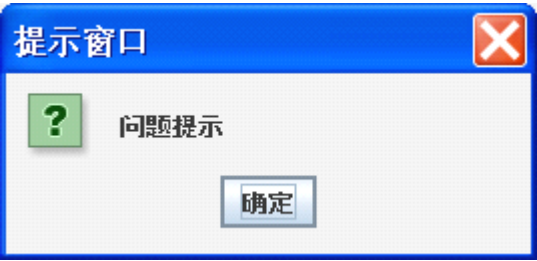


图 4-24 问题信息提示对话框



图 4-25 通用信息提示对话框

2. 任务实施

本任务的代码内容如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class MessageDialog {
4	void go(){
5	JFrame win = new JFrame("对话框上一级窗口");
6	Container contentPane = win.getContentPane();
7	win.setSize(400,200);
8	win.setVisible(true);
9	JOptionPane.showMessageDialog(win,"错误提示","提示窗口",JOptionPane.ERROR_MESSAGE);
10	JOptionPane.showMessageDialog(win,"信息提示","提示窗口",JOptionPane.INFORMATION_MESSAGE);
11	JOptionPane.showMessageDialog(win,"警告提示","提示窗口",JOptionPane.WARNING_MESSAGE);
12	JOptionPane.showMessageDialog(win,"问题提示","提示窗口",JOptionPane.QUESTION_MESSAGE);
13	JOptionPane.showMessageDialog(win,"简单提示","提示窗口",JOptionPane.PLAIN_MESSAGE);
14	}
15	public static void main(String arg[]){
16	MessageDialog fe = new MessageDialog();
17	fe.go();
18	}
19	}

程序在执行先显示 win 窗口，即对话框的上一级窗口。然后依次显示五种信息对话框。在这些对话框没有关闭时，win 窗口不可操作。

3. 分析与提高

(1) 第 5 行至第 8 行创建对话框的上一级窗口 win，设置窗口大小，并显示。其中第 7

行创建的隐性层在本程序中并没有用到。把这条代码加上是让读者尝试一下，将隐性层作为对话框上一级窗口的效果。只要把 `showMessageDialog()` 方法中的参数 `c` 的位置用 `contentPane` 代替即可。两种方式产生的效果是一样的。

(2) 第 9 行至第 13 行用于依次显示五种信息对话框。

【任务 19】认识验证对话框。

1. 基础知识

验证对话框与提示信息对话框很相似，不同之处在于，提示信息对话框只提供“确定”按钮，而验证对话框因为是要用户确认所做的操作是否正确，所以，提供的按钮内容要丰富一些。

验证对话框应用的是 `showConfirmDialog()`，格式如下：

`showConfirmDialog(Component c,Object message,String title,int optionType);`

参数 `optionType` 表示验证对话框的类型，共有 4 种。类型决定着对话框的按钮样式。对应关系如表 4-3 所示。

表 4-3 验证对话框类型对照表

类型	按钮样式
DEFAULT_OPTION	确定
YES_NO_OPTION	是 否
YES_NO_CANEL_OPTION	是 否 撤消
OK_CANCEL_OPTION	确定 撤消

其他参数的含义与 `showMessageDialog()` 方法相同。

4 种验证对话框如图 4-26、4-27、4-28、4-29 所示。

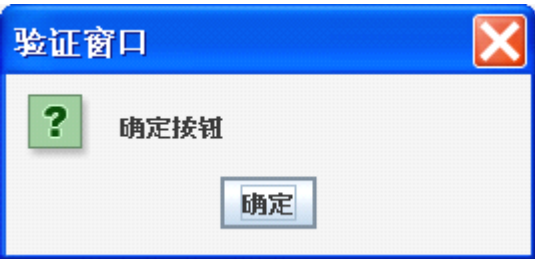


图 4-26 确定按钮的验证对话框



图 4-27 是与否按钮的验证对话框



图 4-28 是、否和撤消按钮的验证对话框



图 4-29 确定和撤消按钮的验证对话框

2. 任务实施

本任务代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class ConfirmDialog {
4	void go(){
5	JFrame win = new JFrame("对话框上一级窗口");
6	Container contentPane = win.getContentPane();
7	win.setSize(400,200);
8	win.setVisible(true);
9	JOptionPane.showConfirmDialog(win,"确定按钮","验证窗口",JOptionPane.DEFAULT_OPTION);
10	JOptionPane.showConfirmDialog(win,"是与否按钮","验证窗口",JOptionPane.YES_NO_OPTION);
11	JOptionPane.showConfirmDialog(win,"是、否和撤消按钮","验证窗口",
	JOptionPane.YES_NO_CANCEL_OPTION);
12	JOptionPane.showConfirmDialog(win,"确定和撤消按钮","验证窗口",JOptionPane.OK_CANCEL_OPTION);
13	}
14	public static void main(String arg[]){
15	ConfirmDialog fe = new ConfirmDialog();
16	fe.go();
17	}
18	}

3. 分析与提高

- (1) 第 9 行至第 12 行创建了 4 种类型的验证对话框。
- (2) 隐性层仍然可以作为验证对话框的上一级容器。

【任务 20】认识输入对话框。

1. 基础知识

输入对话框可以为用户提供与系统进行交互的功能,用户可以在对话框中选择或输入相应信息,系统会收集这些信息,为判断下一步操作提供数据。创建对话框的方法是 showInputDialog()方法。格式如下:

showInputDialog(Component c,Object message,String title,int messageType,ImageIcon image,Object[] selectionValues,Object initSelectionValue);

参数 selectionValues 是数组,用于传递用户可选择值。参数 image 用于传递对话框中要显示的图片。如果为空,则系统会使用输入对话框的默认图片。参数 initSelectionValue 是对话框初始化时显示的默认值。如果为 null,则默认参数 selectionValues 数组的第一个元素为初始值。

其它参数含义与提示信息对话框相同。

2. 任务实施

结合前面制作的调查问卷,设计一个输入对话框。用于调查本次调查问卷问题设置的全面程度。分为非常全面、全面、一般、不全面、非常不全面。参数 image 为空,以对话框采用系统默认图片。以“全面”作为对话框的默认值。

本任务的代码如下:

1	import java.awt.*;
2	import javax.swing.*;
3	public class InputDialog {
4	void go(){
5	JFrame win = new JFrame("对话框上一级窗口");
6	Container contentPane = win.getContentPane();
7	win.setSize(400,200);
8	win.setVisible(true);
9	Object[] selectionValues = {"非常全面","全面","一般","不全面","非常不全面"};
10	JOptionPane.showInputDialog(win,"您觉得本次调查所涉及的问题是否全面: ","调查窗口", JOptionPane.INFORMATION_MESSAGE,null,selectionValues,"全面");
11	}
12	public static void main(String arg[]){
13	InputDialog fe = new InputDialog();
14	fe.go();
15	}
16	}

程序的执行结果如图 4-30 所示。



图 4-30 问题全面性调查对话框

3. 分析与提高

(1) 第 9 行创建输入对话框要使用的调查信息数组 selectionValues。

(2) 第 10 行创建输入对话框，以“您觉得本次调查所涉及的问题是否全面：”作为对话框中显示的信息。以“调查窗口”作为对话框的标题名。采用 INFORMATION_MESSAGE 信息类型和系统默认图片。将“全面”作为对话框的初始默认值。

【任务 21】认识自定义对话框。

1. 基础知识

以上所讲的对话框都有一定的格式要求，如果想要设计出具有自身个性的对话框，就要采用于自定义对话框。其方法为 showOptionDialog()。格式如下：

showOptionDialog(Component c,Object message,String title,int optionType,int messageType,ImageIcon image,Object[] options,Object initValue);

- 参数 optionType 指按钮的类型，共有 4 种。这 4 种类型与验证对话框的按钮类型相同。
- 参数 options 是数组，用于存放对话框中按钮要显示的信息。
- 参数 initValue 是对话框初始化时按钮的默认值。

2. 任务实施

本任务设计的对话框用于调查的用户满意度。分为满意、一般和不满意。对话框采用系统默认图片。以满意作为对话框初始化的默认值。

本任务的代码如下：

1	import java.awt.*;
2	import javax.swing.*;
3	public class OptionDialog {
4	void go(){
5	JFrame win = new JFrame("对话框上一级窗口");
6	Container contentPane = win.getContentPane();
7	win.setSize(400,200);
8	win.setVisible(true);
9	Object[] optionValues = {"满意","一般","不满意"};
10	JOptionPane.showOptionDialog(win,"您对本次调查是否满意: ","调查窗口",JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.QUESTION_MESSAGE,null,optionValues,optionValues[0]);
11	}

12	<code>public static void main(String arg[]){</code>
13	<code> AlertDialog fe = new AlertDialog();</code>
14	<code> fe.go();</code>
15	<code> }</code>
16	<code>}</code>

程序执行结果如图 4-31 所示。



图 4-31 自定义的满意度调查对话框

3. 分析与提高

(1) 第 9 行定义 `optionValues` 数组，用于存放三种满意度。

(2) 第 10 行创建输入对话框，以“您对本次调查是否满意:”作为对话框中显示的信息。以“调查窗口”作为对话框的标题名。采用 `QUESTION_MESSAGE` 信息类型和系统默认图片。将“满意”按钮作为对话框的初始默认按钮。

对话框所显示的按钮信息和个数与参数 `options` 有直接关系。而不受参数 `optionType` 所设置的按钮类型的限制。比如在本任务中设置的 `optionType` 是 `YES_NO_CANCEL_OPTION`。这种类型的按钮是：是、否和撤消。按钮个数与 `options` 数组元素个数相符。在显示对话框时，按 `options` 数组的内容显示。如果将 `optionType` 的类型换成 `YES_NO_OPTION`，这种类型的按钮是：是和否。按钮个数小于 `options` 数组元素个数。但是，显示效果仍然如图 4-30 所示。这样看来，参数 `optionType` 似乎是没有用的，但实际上不是这样的。

窗口所展示出的效果只是表面现象，对于后台操作来说，仍然要依据参数 `optionType` 的值。比如本任务中，如果用户点击“满意”按钮，对话框向系统传递的实际上是“是”的信息。因为，这个对话框的“满意”、“一般”和“不满意”按钮，对于系统来说，实际上是“是”、“否”和“撤消”按钮。在后面常用组件事件处理的介绍中，会更加详细的讲解这部分内容。

4.3.7 常用组件的事件处理

在前面的章节中只介绍了常用组件的使用，在本小节中，将进一步介绍与这些组件有关的方法和事件处理。

在图形用户界面运行时，用户会通过键盘或鼠标对系统进行操作。每一次敲击键盘或点击鼠标都是用户对系统的一个请求，系统要对这个请求做出反应，以完成用户的请求。在这一过程中，就包括了事件和事件的处理。

所谓事件就是对组件做的一个动作。用户敲击键盘或点击鼠标所产生的动作都被称为事件。比如用户用鼠标点击了按钮，就产生了对按钮的“点击事件”。

所谓的事件处理就是产生事件后，系统做出反映的过程。即事件产生后，系统要根据事件的请求做出相应的响应，以完所用户要达到的目的过程。处理事件的任务当然由程序来完成，一般称这类程序为事件处理器。比如用户点击某个按钮后，窗口就关闭了。点击按钮就是一个事件，用户要通过这个事件达到“窗口的关闭”的目的，系统要根据这个点击事件调

用于关闭窗口的程序，执行完后，将窗口关闭，完成事件的处理。

在 Java 语言中，事件的处理并不是由组件本身来完成的。组件产生的事件和事件的处理过程是分开的。Java 将事件的处理委托给叫做监听器的事件处理实体来完成。这种事件处理模式被称为委托处理模式。不同的组件产生的事件不同，比如对键盘的敲击和对按钮的点击，同样的动作但是产生的事件却不同，与此相关事件处理实体（监听器）和处理方法也不同。

事件的处理实体，即监听器，实际上是一个类的实例。这些监听器的格式一般为 XxxListener。在这些监听器中都有如下格式方法：

```
addXxxListener(ListenerType listener);
```

其中，Xxx 是事件的类型。例如，如果是键盘事件，该监听器的名字是 KeyListener，方法的名字就是 addKeyListener。如果是鼠标事件，该监听器的名字是 MouseListener，方法的名字就是 addMouseListener。

事件的处理过程中，如果搞清楚以下三方面知识，就可以很好的应用 Java 语言的事件处理功能。

(1) 谁产生了事件？

也就是事件源是哪种组件。上面提到的“点击某个按钮后，窗口就关闭”的例子中的事件源是按钮。不同的事件源产生的事件是不同的，所以只有搞清楚了事件源是什么，才能确定产生的是什么事件。

(2) 什么样的事件要被处理。

不同的事件要由不同的监听器来处理。搞清楚所产生的事件后，才知道事件要交给哪个监听器来处理。将组件监听起来的格式如下：

```
对象名.addXXXListener(this);
```

(3) 处理事件对应的代码。

因为组件已经被监听，所以产生事件后，系统会根据采用的监听器，去调用相应的方法。而这些方法的方法体是空的，事件处理的对应代码要由程序设计员自己编写。比如“点击某个按钮后，窗口就关闭”。点击事件产生后，要去调用一段能够把窗口关闭的代码，这段代码要由程序设计员来编写。

【任务 22】对调查问题问卷的按钮加入事件处理。

1. 基础知识

对不同组件的操作会产生不同的事件，与之对应的事件处理也不同。点击按钮所产生的事件被称为 ActionEvent 事件。除了按钮能够产生 ActionEvent 事件外，点击菜单项、文本框中的回车和双击列表中的选项也可以产生 ActionEvent 事件。

ActionEvent 提供名字为 getActionCommand()的方法。这个方法可以返回产生事件的组件的字符串。如果产生事件的是按钮，那么这个方法可以返回按钮上的文字。比如“确定”按钮，可以通过这个方法返回“确定”字符串。这样做很有必要。因为在一个窗口中，有多个组件，必须通过这种方法让系统知道用户点击的是那个组件，系统才能做出与这个组件有关的事件处理。

ActionEvent 还包含名字为 getSource()的方法。这个方法的作用是获取产生事件组件的引用变量。在本任务中没有用到此方法。

当用户完成调查问题时，系统进行两次调查分别是问题完整性调查和满意度调查。这两种调查都是以对话框的形式展现。分别采用输入对话框（Input Dialog）和自定义对话框（Option Dialog）。

现在，已经基本搞清楚了事件处理的三件事：产生事件的组件是按钮；产生的是 `ActionEvent` 事件；进行的事件处理是产生两个对话框。

因为产生的是 `ActionEvent` 事件，所以应用的监听器应该是 `ActionListener`，采用的监听方法是 `addActionListener()`。当点击按钮时，触发监听器，由监听器去调用事件处理的代码。但是，事件处理的代码要放在什么位置呢？监听器怎么才能找到它呢？

每个监听器都有自己固定的方法，这些方法在不同的事件产生时被调用。比如 `ActionEvent` 事件的监听器的方法是 `actionPerformed(ActionEvent e)`。本任务中，对按钮的事件处理的代码就应该放在这个方法中。当监听器被触发时，会自动调用这个方法，于是事件就被处理了。

2. 任务实施

只需要对任务 14 的代码进行部分修改就可以实现本任务的功能。

首先要在程序中引入监听器。方法是 `import java.awt.event.*;`，所有监听器都包含在 `awt` 包的 `event` 包下，因此，只要涉及到事件处理的程序都应该引用这个包。

第二步要使 `Question` 类实现监听器 `ActionListener`。为什么是“实现”，而不是“继承”呢？因为，所有监听器都是接口，而不是类。

第三步要使按钮被监听器监听，格式为 `tiJiao.addActionListener(this)`。

第四步添加 `actionPerformed(ActionEvent e)` 方法，方法内容为创建问题完整性调查对话框和满意度调查对话框。这部分参考任务 20 和任务 21。

本任务的主要代码如下：

1	<code>import java.awt.*;</code>
2	<code>import java.awt.event.*;</code>
3	<code>import javax.swing.*;</code>
4	<code>public class QuestionActionEvent implements ActionListener</code>
5	<code>{</code>
6	<code>public void go(){</code>
7	<code>tiJiao.addActionListener(this);</code>
8	<code>.....</code>
9	<code>}</code>
10	<code>public void actionPerformed(ActionEvent e) {</code>
11	<code>if(e.getActionCommand()=="提交"){</code>
12	<code>Object[] selectionValues = {"非常全面","全面","一般","不全面","非常不全面"};</code>
13	<code>JOptionPane.showInputDialog(win,"您觉得本次调查所设及的问题是否全面: ","调查窗口",</code> <code>JOptionPane.INFORMATION_MESSAGE,null,selectionValues,"全面");</code>
14	<code>Object[] optionValues = {"满意","一般","不满意"};</code>
15	<code>JOptionPane.showOptionDialog(win,"您对本次调查是否满意: ","调查窗口</code> <code>",JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.QUESTION_MESSAGE,null,optionValues,optionValues[0]);</code>
16	<code>else{ JOptionPane.showMessageDialog(win,"您的操作不正确! ","提示窗口", JOptionPane.ERROR_MESSAGE); }</code>
17	<code>}</code>
18	<code>public static void main(String arg[]){</code>
19	<code>QuestionActionEvent be = new QuestionActionEvent();</code>
20	<code>be.go();</code>
21	<code>}</code>

代码中的省略部分请参考任务 14。

3. 分析与提高

- (1) 第 2 行引入了与监听器有关的包。
- (2) 第 4 行实现了监听器 `ActionListener` 接口。
- (3) 第 5 行省略了任务 14 的第 4 行至第 31 行。
- (4) 第 7 行对 `tijiao` 按钮进行了监听。这种监听就相当于有一个监视器在随时注意着“提交”按钮，当它产生 `ActionEvent` 事件，监听器马上就可以做出反应，去调用事件对应的方法。
- (5) 第 8 行省略了任务 14 的第 33 行至第 54 行。
- (6) 第 10 行至第 17 行为事件处理的代码部分，由 `ActionListener` 接口的方法 `actionPerformed` 完成。方法中首先利用 `getActionCommand()` 方法判断用户点击的是否为“提交”按钮（第 11 行），如果是，则依次创建问题完整性调查对话框（第 12 和第 13 行）和调查满意度对话框（第 14 行和第 15 行）。否则，创建错误对话框，提示用户操作不正确（第 16 行）。

【任务 23】认识与复选框和单选框有关的事件。

1. 基础知识

复选框和单选框所对应的事件者是 `ItemEvent` 事件。另外当选中列表中的一项或多项时，也产生 `ItemEvent` 事件。当双击列表中的某项时产生的是 `ActionEvent` 事件。

`ItemEvent` 事件对应的监听器是 `ItemListener`，其监听方法是 `addItemListener()`。`ItemListener` 监听器用于处理事件的方法是 `itemStateChanged(ItemEvent e)`。

`ItemEvent` 事件有两个常用方法，分别是 `getStateChange()` 和 `getSource()`。

`getStateChange()` 方法用于判断组件的状态。复选框和单选框都有两种状态：选中和没选中，即 `SELECTED` 和 `DESELECTED`。

`getSource()` 方法与 `ActionEvent` 中的 `getSource()` 方法作用相同，都是用于获得组件的引用变量。例如有如下语句：

```
JCheckBox c1 = new JCheckBox("足球",true);
```

当 `c1` 复选框被选中，利用 `getSource()` 方法可以获取这个复选框的引用变量 `c1`。

复选框和单选框有一个共同的方法 `getText()`，这个方法会在任务实施的代码中用到。它用于获取复选框和单选框的方字信息。

标签也包含 `getText()` 方法，用于获取标签的方字信息。同时标签还包含 `setText(String s)` 方法，这个方法用于设置标签的信息。在本任务中会用到这标签的这两个方法。

2. 任务实施

因为复选框和单选框的事件处理完全一样，所以本任务只对复选框的事件处理进行实验。在任务 9 代码的基础上，实现以下操作：在窗口的“South”位置加入标签，当选中某复选框时，复选框的文字信息显示在这个标签中。

事件处理代码的算法是：当 `ItemEvent` 事件产生时，利用复选框的 `getText()` 方法，获取被选中复选框的文字信息，再利用标签的 `setText(String s)` 方法，将所获取的文字信息传递给标签进行显示。复选框可能同时选取多项，因此，要有一个变量用于收集所有被选复选框的文字信息。

需要注意的是，复选框分为选中和取消两种操作，这两种操作都会触发 `ItemListener` 监

听器，在两种情况下都会执行上述算法。这就会造成如下情况：当选中复选框时，复选框的文字信息被显示在标签中，当取消选中时，文字信息会被再一次显示在标签上。

为了解决这个问题，实现只在复选框被选中时文字信息才会显示，就必须使用 `getStateChange()` 方法，只有当这个方法的值为 `SELECTED` 时，才执行上述算法。

对任务 9 的代码应做如下修改：

首先引入 `java.awt.event.*`。

第二步，使 `Question` 类实现 `ItemListener` 监听器。

第三步，建立标签，并将其摆放在窗口的“South”位置。

第四步，对复选框 `c1`、`c2` 和 `c3` 进行监听，采用的方法是 `addItemListener()`。

第五步，添加 `itemStateChanged(ItemEvent e)` 方法，按上面的算法编写事件处理代码。

本任务的代码如下：

1	<code>import java.awt.*;</code>
2	<code>import java.awt.event.*;</code>
3	<code>import javax.swing.*;</code>
4	<code>public class Question implements ItemListener</code>
5	<code>{</code> <code>JFrame win = new JFrame("调查问卷");</code>
6	<code>Container contentPane = win.getContentPane();</code>
7	<code>JLabel result = new JLabel("您选择的体育运动是: ",JLabel.LEFT);</code>
8	<code>JLabel questionTest = new JLabel("基 本 情 况 调 查 表",JLabel.CENTER);</code>
9	<code>JLabel JLabel1= new JLabel("1、喜欢的体育运动: ",JLabel.LEFT);</code>
10	<code>JCheckBox c1 = new JCheckBox("足球",true);</code>
11	<code>JCheckBox c2 = new JCheckBox("羽毛球");</code>
12	<code>JCheckBox c3 = new JCheckBox("篮球");</code>
13	<code>JPanel p = new JPanel();</code>
14	<code>JPanel p1 = new JPanel();</code>
15	<code>public void go(){</code>
16	<code>c1.addItemListener(this);</code>
17	<code>c2.addItemListener(this);</code>
18	<code>c3.addItemListener(this);</code>
19	<code>contentPane.setLayout(new BorderLayout());</code>
20	<code>//p.setLayout(new GridLayout(?,1));</code>
21	<code>p1.setLayout(new GridLayout(4,1));</code>
22	<code>p1.add(JLabel1);</code>
23	<code>p1.add(c1);p1.add(c2);p1.add(c3);</code>
24	<code>p.add(p1);</code>
25	<code>contentPane.add(questionTest,BorderLayout.NORTH);</code>
26	<code>contentPane.add(p,BorderLayout.CENTER);</code>
27	<code>contentPane.add(result,BorderLayout.SOUTH);</code>
28	<code>win.setSize(400,300);</code>
29	<code>win.setVisible(true);</code>
30	<code>}</code>
31	<code>public void itemStateChanged(ItemEvent e) {</code>
32	<code>String str="";</code>

33	if(e.getStateChange()==e.SELECTED)
34	{if(e.getSource()==c1)
35	{str=result.getText()+c1.getText();
36	result.setText(str);}
37	if(e.getSource()==c2)
38	{str=result.getText()+c2.getText();
39	result.setText(str);}
40	if(e.getSource()==c3)
41	{str=result.getText()+c3.getText();
42	result.setText(str);}
43	}
44	}
45	public static void main(String arg[]){
46	Question be = new Question ();
47	be.go(); }
48	}

程序执行结果如图 4-32 所示。

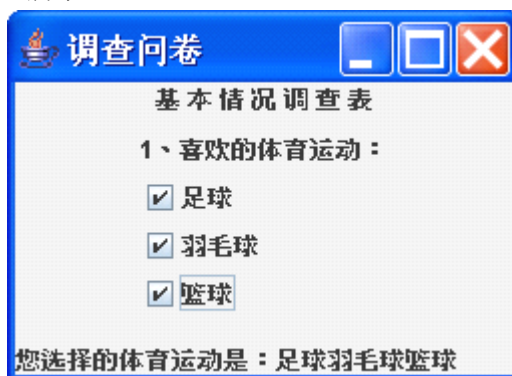


图 4-32 复选框架事件处理

3. 分析与提高

(1) 第 4 行实现了 ItemListener 监听器接口。

(2) 第 7 行创建了用于显示复选框文字信息的标签 result。第 27 行将 result 标签摆放在窗口的“South”位置。

(3) 第 16 行至第 18 行分别对 c1、c2 和 c3 复选框进行监听。

(4) 第 31 行至第 44 行是事件处理代码。当复选框被选中，产生 ItemEvent 事件，触发了 ItemListener 监听器后，监听器就会自动调用 itemStateChanged(ItemEvent e)方法，这部分代码就被执行了。

(5) 第 32 行的 str 字符串变量是用于收集被选中的复选框的文字信息的。同时选中多个复选框后，它们的文字信息会连接成字符串，存放在 str 变量中。

(6) 第 33 行利用 getStateChange()方法，判断复选框是不是被选中，即是否为 SELECTED 状态，如果是则执行第 34 行至第 43 行的代码，否则就什么都不做。这一行的作用就是解决前面提到的“取消选中时，仍然重写文字信息”的问题。

(7) 第 34 行至第 36 行的作用是：当 c1 复选框被选中时，则在标签中显示“足球”的文字信息。

(8) 第 34 行是利用 `getSource()`方法判断被选中的对象是不是 `c1`。

(9) 第 35 行中的 `result.getText()`用于获取标签当前的文字信息，`c1.getText()`用于获取复选框的文字信息。然后将两个文字信息相连形成新的字符串，存放到变量 `str` 中。之所以要获取标签当前的文字信息，是因为我们要将所有被选中的复选框的文字信息显示在标签中，那么，在选中 `c1` 前，可能已经有其它复选框被选中了，因此，标签就可能已经存在了文字信息。必须获取标签已经存在的文字信息，然后将新文字信息与之相连，形成新的字符串，才能实现本任务的功能，否则就会产生丢失文字信息的情况。

(10) 第 36 行是利用标签的 `setText(String s)`方法，将 `str` 字符串中存放的被选中的复选框的文字信息显示在标签 `result` 中。

(11) 第 37 行至第 39 行是处理 `c2` 复选框的代码。第 40 行至第 42 行是处理 `c3` 复选框的代码。处理方法与 `c1` 复选框相同。

【任务 24】认识与文件框和文本域有关的事件。

1. 基础知识

与文本框和文本域有关的事件有三个，分别是 `ActionEvent`、`TextEvent`、`KeyEvent`。`ActionEvent` 是在回车时产生的事件；`TextEvent` 是当文本框或文本域中有文字变化时产生的事件；`KeyEvent` 又称键盘事件，当对文本框或文本域进行文字输入时，产生此事件。

`ActionEvent` 在任务 22 已经介绍过。`TextEvent` 事件的处理大部分可以由 `KeyEvent` 事件处理来代替，所以在本任务中只介绍 `KeyEvent` 事件。

键盘产生的事件都是 `KeyEvent` 事件，这个事件是图形用户界面应用最多的事件之一。

键盘的动作分为三种：按下、松开和敲击。实际上敲击是按下和松开的组合操作过程。这三个动作可以产生 `KeyEvent` 事件。键盘中不同按键产生 `KeyEvent` 事件的情况有所不同。当对键盘中的 26 个英文字母进行敲击时，产生 `KeyEvent` 事件。其他按钮在进行按下和松开时，分别产生 `KeyEvent` 事件。

与 `KeyEvent` 对应的监听器是 `KeyListener` 接口。采用的监听方法是 `addKeyListener()`。

由于不同按键产生事件的情况下不同，所以 `KeyListener` 接口用 `keyPress(KeyEvent e)`、`keyReleased(KeyEvent e)`和 `keyTyped(KeyEvent e)`三个方法，分别响应用户按下、松开和敲击按键时的操作。

`KeyEvent` 有两个常用方法，分别是：`getKeyChar()`和 `getKeyCode()`。

`getKeyChar()`方法用于返回用户所按的按键对应的字符。用户按下字母和数字时，多用这个方法获取数据。`getKeyCode()`返回的是按钮的数字编码。系统为某些按钮分配的是数字编码，如方向键中的向上键，在系统中对应 `KeyEvent.VK_UP` 常量。这个常量存放着“向上键”的数字编码。

利用这两个方法，就可以获取键盘上所有按键的信息了。

2. 任务实施

本任务要在任务 14 的代码的基础上进行修改。对联系方式调查项加入事件处理。假设联系方式指用户的电话，那么，当用户输入 0-9 以外的字符时，系统就要报错。

用户输入数字属于敲击事件。要使用 `KeyListener` 接口中的 `keyTyped(KeyEvent e)`方法。算法如下：

当用户输入信息敲击键盘按键时，产生 `KeyEvent` 事件。触发 `KeyListener` 监听器。监听器调用 `keyTyped(KeyEvent e)`方法，进行事件处理。`keyTyped(KeyEvent e)`方法利用 `KeyEvent` 类的 `getKeyChar()`方法获取用户输入的字符，判断是否为‘0’ - ‘9’之间的字符。如果判

断结果为 false，则弹出错误提示对话框，如果是 true，则不做任何操作，让用户继续输入。

需要注意的是 getKeyChar()方法返回值的数据类型是 char，所以在进行判断时，要将返回值与字符 ‘0’ - ‘9’ 进行比较，而不是数值 0-9。

对任务 14 代码做如下修改：

首先，引入 java.awt.event.*。

第二步，实现 KeyListener 监听器。

第三步，对联系方式文本框 txt 进行监听，格式为 txt.addKeyListener(this)。

第四步，添加方法 keyPressed(KeyEvent e)、keyReleased(KeyEvent e) 和 keyTyped(KeyEvent e)。但在本任务中只应用了 keyTyped(KeyEvent e)。

第五步，按上述算法，编写事件处理代码。

本任务的代码如下：

1	import java.awt.*;
2	import java.awt.event.*;
3	import javax.swing.*;
4	public class Question implements KeyListener
5	{
6	JLabel Jlabel4 = new JLabel("4、联系方式：",JLabel.LEFT);
7	JTextField txt = new JTextField(10);
8
9	public void go(){
10	txt.addKeyListener(this);
11
12	}
13	public void keyPressed(KeyEvent e) { }
14	public void keyReleased(KeyEvent e) { }
15	public void keyTyped(KeyEvent e) {
16	if(!(e.getKeyChar()>='0'&&e.getKeyChar()<='9')){
17	JOptionPane.showMessageDialog(win,"请输入0-9之间的数字","提示窗口",JOptionPane.ERROR_MESSAGE);
18	}
19	}
20	public static void main(String arg[]){
21	Question be = new Question ();
22	be.go();
23	}

程序执行结果如图 4-33 所示。

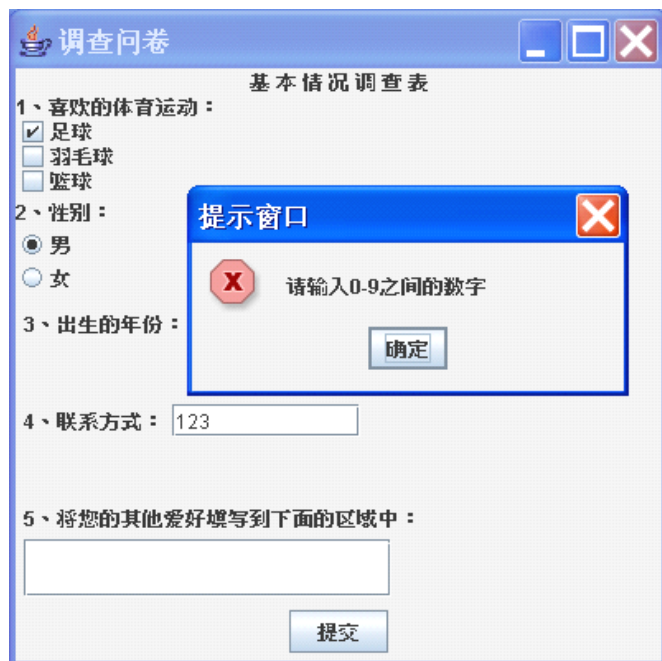


图 4-33 文本框的事件处理

3. 分析与提高

- (1) 第 4 行实现 `KeyListener` 监听器接口。
- (2) 第 5 行省略任务 14 的第 4 行至第 19 行。
- (3) 第 8 行省略任务 14 的第 22 行至第 31 行。
- (4) 第 10 行对 `txt` 文本框进行监听。
- (5) 第 11 行省略任务 14 的第 33 行至第 54 行。
- (6) 第 13 行和第 14 行分别是 `KeyListener` 监听器接口的两个方法。这两个方法在本程序中没有发挥作用。但是，这两个方法必须要添加。因为，接口中的方法都是抽象方法，实现接口就必须实现其所有的抽象方法。即使不使用也要实现。这是抽象方法的特性。
- (7) 第 15 行至第 19 行是 `KeyListener` 监听器的另外一个方法 `keyTyped(KeyEvent e)`。它用于敲击键盘的事件处理。
- (8) 第 16 行利用 `getKeyChar()` 方法获取用户输入的按键字符，并判断用户输入的字符是否为 '0' - '9' 之间的字符。如果不是，则执行第 17 行，创建错误对话框。

【任务 25】实现任务 16 中菜单项的事件处理。

1. 基础知识

系统将菜单项和工具栏的菜单作为按钮处理，它们产生的都是 `ActionEvent` 事件，事件的处理方式也同按钮一样。因此，关于 `ActionEvent` 事件处理的知识参考任务 22。这里不再冗述。

这里要介绍与文本框有关的两个方法，分别是 `setText(String s)` 和 `getText()`。`setText(String s)` 方法用于向文本框传递文字。`getText()` 方法用于取得文本框中的当前文字内容。本任务要用到 `setText(String s)` 方法。

2. 任务实施

任务 16 中共两个菜单，分别是“文件(F)”和“帮助(H)”。

“文件(F)”菜单中包括“清空”和“退出”两个菜单项。点击“清空”项后，应该将显示区（jTextField）中的内容清空。点击“退出”项后，关闭计算器窗口，退出系统。

“帮助(H)”菜单中只有“关于...”菜单项。点击该项后，弹出关于计算器版本信息的窗口。

算法如下：

当点击菜单项时，产生 `ActionEvent` 事件，触发 `ActionListener` 触发器，调用 `actionPerformed(ActionEvent e)` 方法。在该方法中，首先要判断用户点击了哪个菜单项。采用的方法是 `getSource()`。根据 `getSource()` 返回值决定执行什么样的操作。

如果用户点击的是“清空”项，则利用文本框的 `setText(String s)` 方法，将文本框内容设置为空。格式为：`jTextField.setText("")`。

如果用户点击的是“退出”项，则关闭窗口，退出当前执行的程序。其实，只要退出当前执行的程序，窗口就自然会关闭。`System` 类有一个 `exit(int i)` 的方法，专门用于结束当前执行的程序。格式为：`System.exit(0)`。

如果用户点击“关于...”项，则弹出关于计算器版本的窗口。

需要对任务 16 做如下修改：

首先，引入 `java.awt.event.*`。

第二步，实现 `ActionListener` 监听器接口。

第三步，对所有菜单项进行监听，采用 `addActionListener()` 方法。

第四步，添加 `actionPerformed(ActionEvent e)` 方法。

第五步，按上述算法，编写事件处理代码。

本任务代码如下：

1	<code>import java.awt.*;</code>
2	<code>import java.awt.event.*;</code>
3	<code>import javax.swing.*;</code>
4	<code>public class Calculator implements ActionListener</code>
5	<code>{</code> <code>JFrame win = new JFrame("计算器");</code>
6	<code>Container contentPane = win.getContentPane();</code>
7	<code>JTextField jTextField = new JTextField();</code>
8	<code>JMenuBar MBar = new JMenuBar();</code>
9	<code>JMenu file = new JMenu("文件(F)");</code>
10	<code>JMenu help = new JMenu("帮助(H)");</code>
11	<code>JMenuItem qingkong = new JMenuItem("清空");</code>
12	<code>JMenuItem exit = new JMenuItem("退出");</code>
13	<code>JMenuItem about = new JMenuItem("关于...");</code>
14	<code>public void go() {</code>
15	<code>qingkong.addActionListener(this);</code>
16	<code>exit.addActionListener(this);</code>
17	<code>about.addActionListener(this);</code>
18	<code>file.add(qingkong);</code>
19	<code>file.add(exit);</code>
20	<code>help.add(about);</code>
21	<code>MBar.add(file);</code>
22	<code>MBar.add(help);</code>

23	contentPane.add(jTextField);
24	win.setJMenuBar(MBar);
25	win.setSize(200,100);
26	win.setVisible(true);
27	}
28	public void actionPerformed(ActionEvent e) {
29	if(e.getSource()==qingkong)
30	jTextField.setText("");
31	if(e.getSource()==exit)
32	System.exit(0);
33	if(e.getSource()==about)
34	{JFrame newF = new JFrame("帮助窗口");
35	JLabel banben = new JLabel("计算器版本5.1 版权所有(c) 2010 BITC");
36	newF.add(banben);
37	newF.setSize(260,100);
38	newF.setVisible(true);}
39	}
40	public static void main(String arg[]) {
41	Calculator be = new Calculator();
42	be.go(); }
43	}

3. 分析与提高

- (1) 第 4 行实现 ActionListener 监听器接口。
- (2) 第 15 行至第 17 行分别对 qingkong、exit 和 about 菜单项进行监听。
- (3) 第 28 行至第 39 行为 actionPerformed(ActionEvent e)方法，用于处理(ActionEvent 事件。
- (4) 第 29 行利用 getSource()方法获取组件引用变量，并判断是否为 qingkong，如果是，则执行第 30 行，利用文本框的 setText(String s)方法将文本框内容设置为空。
- (5) 第 31 行利用 getSource()方法获取组件引用变量，并判断是否为 exit，如果是，则执行第 32 行，退出该程序的执行，窗口关闭。
- (6) 第 33 行利用 getSource()方法获取组件引用变量，并判断是否为 about，如果是，则执行第 34 行至第 39 行的代码，弹出包含版本信息的窗口。第 34 行至第 39 行的代码也可以用自定义对话框来代替，这样程序代码就更加精练了。

在本任务程序的基础上，可以完成工具栏事件的处理。需要做提修改如下：

首先，参考任务 17，将工具栏添加到本任务的程序中。

第二步，对工具栏中的按钮进行监听。

第三步，在 actionPerformed(ActionEvent e)方法中添加对工具栏事件进行处理的代码。

因为工具栏中的按钮功能与菜单中的菜单项一样，所以只需要在本任务的第 29 行、第 31 行和第 33 行添加判定条件即可。参考代码如下：

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource()==qingkong||e.getSource()==jb1)
        jTextField.setText("");
    if(e.getSource()==exit||e.getSource()==jb2)
```

```

        System.exit(0);
    if(e.getSource()==about||e.getSource()==jb3)
        {JFrame newF = new JFrame("新窗口");
        newF.setSize(100,100);
        newF.setVisible(true);}
    }

```

【任务 26】认识与鼠标有关的事件。

1. 基础知识

用户使用鼠标进行操作进，产生的就是 `MouseEvent` 事件。`MouseEvent` 类有两个监听器接口，分别为 `MouseListener` 和 `MouseMotionListener`。

这两个接口所处理的鼠标操作不同。

(1) `MouseListener` 监听器接口的方法

`MouseListener` 监听器主要响应鼠标的点击、按下、释放和进入或退出某被监听区域时的操作。进行监听时采用方法为 `addMouseListener()`。

- `mouseClicked(MouseEvent e)`方法

该方法用于响应用户点击鼠标时的动作。所谓点击就是按下和释放鼠标的过程。

- `mouseEntered(MouseEvent e)`方法

当鼠标进入某被监听的区域时，该方法被调用。

- `mouseExited(MouseEvent e)`方法

当鼠标退出某被监听的区域时，该方法被调用。

- `mousePressed(MouseEvent e)`方法和 `mouseReleased(MouseEvent e)`方法

当用户按下鼠标时，调用 `mousePressed(MouseEvent e)` 方法，当鼠标被释放时，`mouseReleased(MouseEvent e)` 方法被调用。当点击鼠标时，必须是一个按下和释放的过程，因此，这两个方法往往先后被调用。

(2) `MouseMotionListener` 监听器接口的方法

`MouseMotionListener` 监听器主要响应鼠标的拖曳和移动操作。进行监听时采用的方法为 `addMouseMotionListener()`。

- `mouseDragged(MouseEvent e)`

当用户拖曳鼠标时，该方法被调用。

- `mouseMoved(MouseEvent e)`

当用户移动鼠标时，该方法被调用。

2. 任务实施

采用如下算法演示鼠标事件处理的过程。

如果将一个窗口的隐性层监听起来，那么在这个隐性层上所有鼠标操作都会触发 `MouseEvent` 事件。另外，可以利用标签展示出鼠标当前所处的操作，以展现不同方法被调用的过程。因此，只要创建包含一个标签的窗口，就可以完成本任务的演示。

分别用两段程序来演示 `MouseListener` 监听器和 `MouseMotionListener` 监听器对鼠标事件处理的情况。

`MouseListener` 监听器进行事件处理的代码如下：

1	import java.awt.*;
---	--------------------

2	import java.awt.event.*;
3	import javax.swing.*;
4	public class MouseAction implements MouseListener
5	{
6	JFrame win = new JFrame("鼠标事件演示窗口");
7	Container contentPane = win.getContentPane();
8	JLabel label = new JLabel("起始状态， 还没鼠标事件",JLabel.CENTER);
9	public void go()
10	{ contentPane.addMouseListener(this);
11	contentPane.add(label);
12	win.setSize(280,180);
13	win.setVisible(true); }
14	public void mousePressed(MouseEvent e) {
15	label.setText("您已经按下鼠标按钮"); }
16	public void mouseReleased(MouseEvent e){
17	label.setText("您已经释放鼠标按钮"); }
18	public void mouseEntered(MouseEvent e) {
19	label.setText("鼠标光标进入按钮"); }
20	public void mouseExited(MouseEvent e) {
21	label.setText("鼠标光标离开按钮"); }
22	public void mouseClicked(MouseEvent e) {
23	label.setText("您已经点击了鼠标"); }
24	public static void main(String args[])
25	{
26	MouseAction be = new MouseAction();
27	be.go();
28	}
29	}

MouseMotionListener 监听器进行事件处理的代码如下：

1	import java.awt.*;
2	import java.awt.event.*;
3	import javax.swing.*;
4	public class MouseMotionAction implements MouseMotionListener
5	{
6	JFrame win = new JFrame("鼠标事件演示窗口");
7	Container contentPane = win.getContentPane();
8	JLabel label = new JLabel("起始状态， 还没鼠标事件",JLabel.CENTER);
9	public void go()
10	{ contentPane.addMouseMotionListener(this);
11	contentPane.add(label);
12	win.setSize(280,180);
13	win.setVisible(true); }
14	public void mouseDragged(MouseEvent arg0) {

15	label.setText("您正在拖动鼠标");	}
16	public void mouseMoved(MouseEvent arg0) {	
17	label.setText("您正在移动鼠标");	}
18	public static void main(String args[])	
19	{	
20	MouseMotionAction be = new MouseMotionAction();	
21	be.go();	
22	}	
23	}	

3. 分析与提高

MouseListener 监听器进行事件处理的代码：

- (1) 第 4 行实现了 MouseListener 监听器接口。
- (2) 第 8 行创建用于显示鼠标当前操作的标签。
- (3) 第 10 行对隐性层 contentPane 进行监听。
- (4) 第 14 行和第 15 行是处理鼠标按下操作的代码。当用户在窗口中按下鼠标时，调用 mousePressed (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“您已经按下鼠标按钮”的提示信息。
- (5) 第 16 行和第 17 行是处理鼠标释放操作的代码。当用户在窗口中释放鼠标时，调用 mouseReleased (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“您已经释放鼠标按钮”的提示信息。
- (6) 第 18 行和第 19 行是处理鼠标进入监听区域时的代码。当鼠标进入被监听区域时，调用 mouseEntered (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“鼠标光标进入按钮”的提示信息。
- (7) 第 20 行和第 21 行是处理鼠标离开监听区域时的代码。当鼠标离开被监听区域时，调用 mouseExited (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“鼠标光标离开按钮”的提示信息。
- (8) 第 22 行和第 23 行是处理鼠标点击操作的代码。当用户点击鼠标时，调用 mouseClicked (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“您已经点击了鼠标”的提示信息。

(9) 需要提醒的是：程序第 16 和第 17 行似乎永远没有被执行，因为，在标签中没有显示“您已经释放鼠标按钮”的提示信息。其实这两行代码确实被执行了。当用户按下鼠标时，第 14 行和第 15 行被调用。当用户释放鼠标时，第 16 行和第 17 行被调用。但是，由于用户完成了按下和释放的过程，就相当于做了一次点击鼠标的操作。因此，还没等我们看到第 16 行和第 17 行的显示结果时，第 22 行和第 23 行就被调用了。可以将第 23 行的语句注释掉，即让 mouseClicked (MouseEvent arg0)成为空方法，就可以看到第 16 行和第 17 行的执行结果了。

MouseMotionListener 监听器进行事件处理的代码：

- (1) 第 4 行实现了 MouseMotionListener 监听器接口。
- (2) 第 14 行和第 15 行是处理鼠标拖动操作的代码。当用户在窗口中拖动鼠标时，调用 mouseDragged(MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“您正在拖动鼠标”的提示信息。
- (3) 第 16 行和第 17 行是处理鼠标移动操作的代码。当用户在窗口中移动鼠标时，调用 mouseMoved (MouseEvent arg0)方法，利用标签的 setText(String s)方法，在窗口中显示“您

正在移动鼠标”的提示信息

4.4 上机练习

1. 设计一个新的调查项，并尝试将新的调查项加入到任务 14 的代码中。
2. 使用 JTable 组件制作一张课程表。
3. 参考图 4-34 制作用户登录界面。

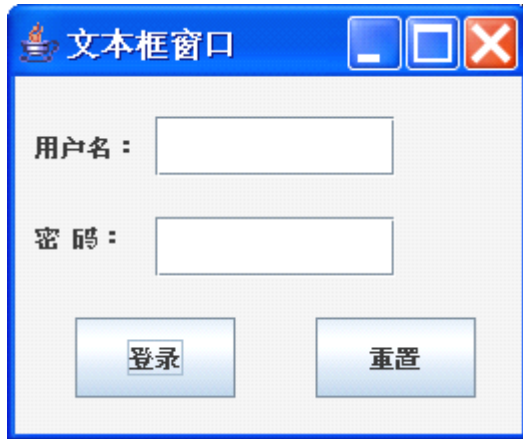


图 4-34 用户登录界面

提示：

“用户名:”和“密码:”可能通过标签组件实现。用户名和密码的输入可以通过文本框实现。文本框组件除了用 JTextField 类实现外，还有一个名字为 JPasswordField 的类。这个类专门用于创建密码文本框。用户所输入的数据会以“*”代替，以提高安全性。JPasswordField 使用与 JTextField 基本相同。

登录和重置是两个按钮，可以通过 JButton 类创建。

制作本窗口时，可以尝试用多种布局管理器或直接用自定义布局的方法。

4. 在第 3 题的基础上，为窗口增加事件处理功能。要求如下：

- (1) 当点击“登录”按钮时，弹出“欢迎进行本系统”的对话框。
- (2) 当点击“重置”按钮时，将用户名和密码对应的文本框清空。

提示：

“登录”和“重置”按钮产生的都是(ActionEvent)事件，其监听器为 ActionListener。“登录”产生的对话框，可以利用提示信息对话框实现。“重置”按钮产生的是清空操作，可以利用文本框的 setText(String s)方法实现。

5. 将任务 17 的工具栏的菜单项修改为以图片形式显示的菜单项。

提示：

工具栏中的菜单实际上就是按钮，只要在创建按钮时，采用 JButton 对象名 = new JButton(ImageIcon image)的方式，就可以创建出图片形式的按钮。

6. 对任务 22 进行修改，使用 getSource()方法进行事件源判断，完成事件处理。

提示：

任务 22 使用 getActionCommand()判断事件源，判断条件是按钮的文字信息。getSource()将组件的引用作为事件源的判断条件。

4.5 习题

1. 填空题

- (1) GUI 的中文含义是_____。
- (2) Java 语言提供的图形用户界面的功能包括_____和_____两部分。
- (3) AWT 的缺点是_____。
- (4) awt 包包含在_____包，swing 包包含在_____包。
- (5) 常用的布局管理器有_____, _____, _____, _____。
- (6) Panel 和 Applet 容器的默认布局管理器是_____。
- (7) Frame 容器的默认布局管理器是_____。
- (8) 自定义布局管理器使用的方法 setBounds() 的中参数含义是：前两个参数表示_____, 后两个参数的表示_____。
- (9) JFrame 和 JPanel 都是属于_____包下的类。
- (10) 标签组件使用是_____类，按钮组件使用的是_____类，复选框组件使用的是_____类，单选框组件使用的是_____类，下拉列表组件使用的是_____类，文本框组件使用的是_____类，文本域组件使用的是_____类。
- (11) 菜单由三部分组成，分别是_____, _____, _____。它们分别对应的类为_____, _____, _____。
- (12) 包含事件处理内容的程序一般都要引入_____包。

2. 问答题

- (1) AWT 和 Swing 的关系是什么？
- (2) BorderLayout 布局管理器将窗口划分成几个区域？如何控制组件的摆放？
- (3) GridLayout 布局管理器参数的含义是什么？
- (4) 简单说明 JFrame 和 JPanel 的区别。
- (5) 菜单条的摆放和工具栏的摆放有什么不同？
- (6) 提示信息对话框共有几种类型？区别是什么？
- (7) 总结一下本章介绍的事件，分别对这些事件的含义进行说明。