



按钮放在四周，将标签放在中间。为 4 个按钮都注册监听器后，不管对哪一个按钮进行操作都会执行监听方法。在监听方法中首先判断事件源，然后在中间的标签中显示是对哪一个按钮进行操作。



15.4 空布局

空布局就是没有使用布局管理器，在空布局的情况下将根据控件的自身信息来为控件指定位置。在使用空布局时，就会使开发人员的工作量变得很多。开发人员需要为每一个控制指定位置和大小，因为这些都是开发人员指定的，所以当容器的大小发生变化时，控件的大小是不会发生变化的。虽然使用空布局加大了开发人员的工作量，但是这样使控件的摆放更加灵活，从而使界面的外观更加多样。

15.4.1 空布局介绍

空布局是不需要使用类来创建的，只需要在程序指定布局管理器为 null。将控件添加到空布局容器中时，仍然是使用 add 方法。因为这里使用的是空布局管理器，所以在添加控件之前，要对控件进行设置操作。设置操作是通过 setBounds 方法来完成的， setBounds 方法的基本语法格式如下所示。

```
public void setBounds(int x,int y,int width,int height);
```

其中，x 和 y 表示的是控件最左上侧的坐标，从而固定了该控件的位置。width 和 height 表示的是空间的宽度和高度，从而指定了控件的大小。

使用空布局的优点是使用简单，对控件的摆放位置灵活。使用空布局的缺点就是需要对每一个空间单独指定大小和位置，这在很多控件的情况下，工作量是可想而知的。

15.4.2 使用空布局

通过上一节中对空布局的学习，读者已经对空布局有了基本了解。在使用空布局时要注意的是必须为每一个控件设置位置和大小，这是读者特别需要注意的。

【范例 15-6】示例代码 15-6 是一个使用空布局的程序。

示例代码 15-6

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.event.*; //导入事件包
03 import java.awt.*;
04 //继承 JFrame 类
05 public class BuJu6 extends JFrame implements ActionListener
06 {
07     JButton jb=new JButton("动"); //创建按钮
08     JLabel jl=new JLabel("标签"); //创建标签
09     JPanel jp=new JPanel(); //创建一个面板
10     int i=10;
11     int j=10;
12     //定义构造器
13     public BuJu6()
14     {
15         this.setTitle("使用空布局管理器"); //设置窗口名称
```

```

15         jp.setLayout(null);           //设置面板的布局为空布局
16         jb.setBounds(50,50,70,20);    //设置按钮的位置和大小
17         jb.addActionListener(this);   //为按钮注册监听器
18         jp.add(jb);                //将按钮添加到面板中
19         jl.setBounds(150,70,100,20); //设置标签的位置和大小
20         jp.add(jl);                //将标签添加到面板中
21         this.add(jp);              //将面板添加到窗口中
22         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //设置窗口的大小和位置
23         this.setVisible(true);      //设置窗口是可见的
24     }
25 }
26 public void actionPerformed(ActionEvent e) //实现监听接口方法
27 {
28     i+=10;
29     j+=10;
30     jl.setBounds((150+i),(70+j),100,20);
31 }
32 public static void main(String args[])
33 {
34     BuJu6 s=new BuJu6();
35 }
36 }

```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 15-11 所示。

【代码解析】 在运行结果中按下按钮后，标签的位置是会发生变化的。该程序很简单，只是为了讲解空布局，有兴趣的读者可以改进该程序，例如将标签换为图片，让该图片围绕着按钮转。在该程序的第 16 行定义面板使用空布局，从而在将按钮和标签添加到面板之前需要先设置按钮和标签的位置和大小。在程序的第 17 行对按钮的位置和大小进行了设置，在程序的第 20 行对标签的位置和大小进行了设置。同时在程序中为按钮注册了监听器，当按下按钮后就会执行监听方法，从而改变标签的位置。



图 15-11 使用空布局

15.5 卡片布局

卡片布局是一种在 Swing 布局管理器中不很常用的布局管理器，但是使用卡片布局在特定情况下能够起到非常好的效果。在卡片布局的容器中可以添加任意多个控件，但是只能在容器中显示一个控件。添加在卡片布局容器中的控件的大小都和容器的大小相同，所有的控件也具有相同的大小。

15.5.1 卡片布局介绍

卡片布局是通过 CardLayout 类创建的。CardLayout 类具有两个构造器，一个是无参构造器，另一个是需要指定控件和容器边界水平间距和垂直间距的构造器。这些已经在前面多次介绍，这里不再做过多介绍。

在卡片布局容器中一次只能显示一个控件，要想显示其他控件，就需要调用 CardLayout 类中的方法来执行。在 CardLayout 类的方法中，first 方法和 last 方法分别是显



示第一次添加的控件和显示最后一次添加的控件。`next` 方法和 `previous` 方法分别是显示下一个添加控件和上一个添加控件。这 4 个方法都是具有一个参数的，该参数是指定对哪一个容器中的控件进行操作。



注意：在卡片布局容器中一次只能显示一个控件。

`CardLayout` 类中还有一个 `show` 方法，使用该方法可以显示指定的控件，该方法具有两个参数，第一个参数是指定对哪一个容器中的控件进行操作，第二个参数是指定要显示控件的名称。控件的名称是提前为控件设置的。

为控件起名称是在添加控件的时候设置的，向卡片布局容器中添加控件同样也是通过 `add` 方法来完成的。但是这里的 `add` 方法需要两个参数，第一个参数指定要添加的控件，第二个参数就是为该控件起的名称。

15.5.2 使用卡片布局

在使用卡片布局时，通常是一个非顶级容器来使用的，这是因为如果让顶级容器使用卡片布局，则整个窗体就显示一个控件，也无法进行操作显示其他控件，这样设计就没有意义。通常是将顶级容器设置为空布局管理器，在其中添加面板和控件。在面板中使用卡片布局，让控件对面板中的控件进行操作。

【范例 15-7】示例代码 15-7 是一个使用卡片布局的程序。

示例代码 15-7

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.event.*; //导入事件包
03 import java.awt.*;
04 //继承 JFrame 类
05 public class BuJu7 extends JFrame implements ActionListener
06 {
07     JButton jbf=new JButton("第一个"); //创建按钮
08     JButton jbl=new JButton("最后一个"); //创建按钮
09     JButton jbp=new JButton("上一个"); //创建按钮
10     JButton jbn=new JButton("下一个"); //创建按钮
11     JButton jbz=new JButton("中间一个"); //创建按钮
12     JPanel jp=new JPanel(); //创建一个面板
13     CardLayout cl=new CardLayout(); //定义构造器
14     public BuJu7()
15     {
16         this.setTitle("使用卡片布局管理器"); //设置窗口名称
17         this.setLayout(null); //设置面板的布局为空布局
18         jbf.setBounds(120, 40, 100, 20); //设置按钮的位置和大小
19         jbl.setBounds(120, 70, 100, 20); //设置按钮的位置和大小
20         jbp.setBounds(120, 100, 100, 20); //设置按钮的位置和大小
21         jbn.setBounds(120, 130, 100, 20); //设置按钮的位置和大小
22         jbz.setBounds(120, 160, 100, 20); //设置按钮的位置和大小
23         this.add(jbf); //将按钮添加到窗体中
24         this.add(jbl);
25         this.add(jbp);
26         this.add(jbn);
```

```

28         this.add(jbz);
29         jbf.addActionListener(this);           //为按钮注册监听器
30         jbl.addActionListener(this);
31         jbp.addActionListener(this);
32         jbn.addActionListener(this);
33         jbz.addActionListener(this);
34         jp.setBounds(10,40,100,100);        //设置面板的位置和大小
35         jp.setLayout(cl);                  //设置面板的布局为卡片布局
36         for(int i=0;i<50;i++)
37         {
38             JButton jb=new JButton("按钮"+i); //创建多个按钮
39             jb.addActionListener(this);        //将新创建的按钮添加到面板中
40         }
41         this.add(jp);                      //将面板添加到窗口中
42         this.setBounds(200,200,300,220);    //设置窗口的大小和位置
43         this.setVisible(true);            //设置窗口是可见的
44     }
45     public void actionPerformed(ActionEvent e) //实现监听接口方法
46     {
47         if(e.getSource()==jbf)
48         {
49             cl.first(jp);                //显示第一个控件
50         }
51         else if(e.getSource()==jbl)
52         {
53             cl.last(jp);               //显示最后一个控件
54         }
55         else if(e.getSource()==jbp)
56         {
57             cl.previous(jp);           //显示上一个控件
58         }
59         else if(e.getSource()==jbn)
60         {
61             cl.next(jp);              //显示下一个控件
62         }
63         else if(e.getSource()==jbz)
64         {
65             cl.show(jp,"25");        //显示指定控件
66         }
67     }
68     public static void main(String args[])
69     {
70         BuJu7 s=new BuJu7();
71     }
72 }

```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 15-12 所示。

【代码解析】 在图 15-12 的运行结果中，通过右边小按钮的操作，就能改变左边面板中显示的控件，具体操作效果，读者可以自己进行操作实验。需要注意的是在该程序中使用卡片布局的是窗体中的面板，而不是窗体。在左边的面板中设置布局管理器为卡片布局管理器，并循环创建并添加到该面板中，默认显示的是添加的第一个控件。在窗体的右边定义了一组按钮，并为这些按钮都注册了监听器。在监听方法中将根



图 15-12 卡片布局



据操作的事件源来执行对应的方法，通过这些方法来进行面板中控件的显示。



15.6 综合练习

1. 编写一个让用户自由选择使用哪一种布局管理器进行控件显示的程序。

【提示】可以先只使用流布局和网格布局进行实验。在程序中首先要判断选择的是哪一种布局管理器，然后根据选择进行控件显示。

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.event.*; //导入事件包
03 import java.awt.*;
04 //继承 JFrame 类
05 public class LianXil extends JFrame implements ActionListener
06 {
07     int i;
08     JButton jbl=new JButton("流布局");
09     JButton jbw=new JButton("网格布局");
10     JPanel jp=new JPanel(); //创建一个面板
11     //定义构造器
12     public LianXil()
13     {
14         this.setTitle("使用不同布局管理器"); //设置窗口名称
15         jp.add(jbl); //将按钮添加到面板中
16         jp.add(jbw); //将按钮添加到面板中
17         jbl.addActionListener(this); //为控件注册监听器
18         jbw.addActionListener(this); //为控件注册监听器
19         this.add(jp); //将面板添加到窗口中
20         this.setBounds(300,250,300,200); //设置窗口的大小和位置
21         this.setVisible(true); //设置窗口是可见的
22     }
23     public void actionPerformed(ActionEvent e) //实现监听接口方法
24     {
25         if(e.getSource()==jbl) //如果用户选择流布局
26         {
27             ++i;
28             JButton jbi=new JButton("按钮"+i); //创建新按钮
29             jp.add(jbi); //将按钮添加到面板中
30             jp.setLayout(new FlowLayout()); //设置面板为流布局
31             this.setTitle("使用流布局"); //改变窗体名称
32             jp.revalidate(); //刷新
33         }
34         else if(e.getSource()==jbw) //如果用户选择网格布局
35         {
36             ++i;
37             JButton jbi=new JButton("按钮"+i); //创建新按钮
38             jp.add(jbi); //将按钮添加到面板中
39             jp.setLayout(new GridLayout(3,4)); //设置面板为网格布局
40             this.setTitle("使用网格布局"); //改变窗体名称
41             jp.revalidate(); //刷新
42         }
43     }
44     public static void main(String args[])
45 }
```

```

45     {
46         LianXi1 s=new LianXi1();
47     }
48 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 15-13 所示。

2. 使用卡片布局编写一个摇奖程序。

【提示】 摆奖程序肯定就是一个能够随机产生不同奖的程序。

```

01 import javax.swing.*;
02 import java.awt.event.*;
03 import java.awt.*;
04 //继承JFrame类
05 public class LianXi2 extends JFrame implements ActionListener
06 {
07     JButton jbr=new JButton("摇奖");           //创建按钮
08     JPanel jp=new JPanel();                   //创建一个面板
09     CardLayout cl=new CardLayout();           //定义构造器
10     public LianXi2()
11     {
12         this.setTitle("使用卡片布局管理器");    //设置窗口名称
13         this.setLayout(null);                   //设置面板的布局为空布局
14         jbr.setBounds(120,40,100,20);          //设置按钮的位置和大小
15         this.add(jbr);                       //为按钮注册监听器
16         jp.setBounds(10,40,100,100);          //设置面板的位置和大小
17         jp.setLayout(cl);                   //设置面板的布局为卡片布局
18         for(int i=1;i<6;i++)
19         {
20             JButton jb=new JButton(i+"等奖"); //创建表示奖级别的多个按钮
21             jb.setBounds(10,40,100,100);        //将新创建的按钮添加到面板中
22             jp.add(jb,""+i);
23         }
24         this.add(jp);                         //将面板添加到窗口中
25         this.setBounds(200,200,300,220);       //设置窗口的大小和位置
26         this.setVisible(true);                //设置窗口是可见的
27     }
28     public void actionPerformed(ActionEvent e) //实现监听接口方法
29     {
30         int i=(int)Math.floor(Math.random()*6); //随机产生一个从1到5之间的整数
31         cl.show(jp,""+i);                    //显示随机控件
32     }
33     public static void main(String args[])
34     {
35         LianXi2 s=new LianXi2();
36     }
37 }

```

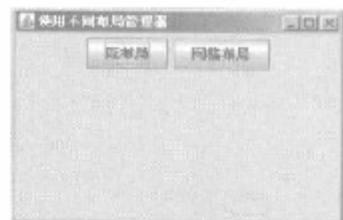


图 15-13 使用不同布局

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 15-14 所示。

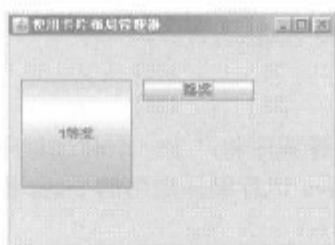


图 15-14 摆奖程序



15.7 小结

通过本章的学习，读者已经对 Swing 开发中经常要使用的布局管理器有了一些的了解。Swing 中不但可以使用本章中讲解的流布局、网格布局、边框布局、空布局和卡片布局，还可以使用箱式布局和弹簧布局。对这些布局方式，读者可以通过电子工业出版社出版的《Java 优化编程（第 2 版）》一书进行更详细的学习。



15.8 习题

一、填空题

1. 流布局是通过_____类来创建，FlowLayout 类具有三种构造器。
2. 网格布局是通过_____类来创建的。GridLayout 类具有三个构造器，使用无参构造器将创建具有默认行和默认列的网格布局。
3. 边框布局是通过_____类创建的。BorderLayout 类具有两个构造器，一个是无参构造器，另一个是_____的构造器，通常使用无参构造器来创建边框布局管理器。
4. 卡片布局是通过_____类创建的。CardLayout 类具有两个构造器，一个是无参构造器，另一个是需要指定控件和容器边界_____和_____的构造器。

二、选择题

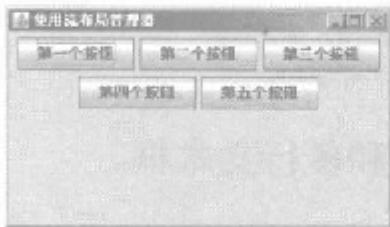
1. 判断下面程序的运行结果（ ）。

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.event.*; //导入事件包
03 import java.awt.*;
04 //继承 JFrame 类
05 public class BuJui extends JFrame
06 {
07     JButton jb1=new JButton("第一个按钮"); //创建第一个按钮
08     JButton jb2=new JButton("第二个按钮"); //创建第二个按钮
09     JButton jb3=new JButton("第三个按钮"); //创建第三个按钮
10     JButton jb4=new JButton("第四个按钮"); //创建第四个按钮
11     JButton jb5=new JButton("第五个按钮"); //创建第五个按钮
12     JPanel jp=new JPanel(); //创建一个面板
13     //定义构造器
14     public BuJui()
15     {
```

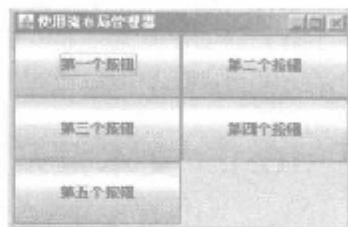
```

16     this.setTitle("使用流布局器");
17     jp.setLayout(new FlowLayout());
18     jp.add(jb1);
19     jp.add(jb2);
20     jp.add(jb3);
21     jp.add(jb4);
22     jp.add(jb5);
23     this.add(jp);
24     this.setBounds(300,250,300,200);
25     this.setVisible(true);
26 }
27 public static void main(String args[])
28 {
29     BuJui s=new BuJui();
30 }
31 }

```



A.



B.

2. 在卡片布局中，使用下面哪一个方法可以直接选择对应的控件（ ）。
- A. first 方法 B. next 方法
 C. previous 方法 D. show 方法
3. 窗口的默认布局管理器是（ ）。
- A. 流布局 B. 网格布局
 C. 边框布局 D. 空布局

三、简答题

1. 简述边框布局的使用，以及各个方位的表示。
2. 简述各个布局的不同，以及各自的使用。

四、编程题

1. 编写一个计算器界面，能够进行基本的加法和减法运算。
2. 编写一个显示四季不同景色的程序，通过单击四周的不同按钮，在中间显示不同的图片。

第 16 章 Swing 常用控件

如果超市中只卖一种商品，那就会显得非常单一。在上一章学习布局管理器时，读者可能就有这样的感觉，一直都是使用按钮控件进行举例说明，这就好像超市中只卖一种商品。本章中就来为超市中提供更多的商品，也就是介绍 Swing 中更多的控件。控件是使界面内容丰富的一个必不可少的一部分，在 Swing 中的控件除了按钮之外，还包括文本框、复选框、单选按钮和菜单等很多内容。通过本章的学习，读者应该实现如下几个目标。

- 了解如何创建文本框和文本框的实际应用。
- 了解如何创建复选框和复选框的实际应用。
- 了解如何创建单选按钮和单选按钮的实际应用。



16.1 文本框及密码框和多行文本框

文本框和按钮一样，都是非常常用的控件，文本框提供了一个输入信息的控件。密码框和多行文本框是和文本框很相似的，密码框和文本框的外观十分相似，只是输入的内容显示为特殊符号，从而起到保护密码的作用。多行文本框从名称上就可以看出是一个具有多行文本的文本框，在多行文本框中输入内容时是可以进行换行操作的。

16.1.1 创建文本框

文本框是通过 JTextField 类来创建的，在创建的文本框中当文本超出文本框规定长度时，将自动滚动文本显示。文本框是通过 JTextField 类的构造器创建的，包括 5 种构造器，如表 16-1 所示。

表 16-1 JTextField 类构造器

构造器	说明
public JTextField()	创建普通的文本框
public JTextField(String text)	创建具有默认值的文本框
public JTextField(int columns)	创建具有指定长度的文本框
public JTextField(String text,int columns)	创建具有默认值和指定长度的文本框
public JTextField(Document doc,String text,int columns)	创建具有默认值、长度与文档模型的文本框

 **提示：**文本框也是会触发事件的，它和按钮一样，都是触发 ActionEvent 事件。按钮是被单击时触发事件，而文本框是当用户按下回车键时触发事件。

【范例 16-1】示例代码 16-1 是一个创建文本框的程序。

示例代码 16-1

```

01 import javax.swing.*;           //导入 Swing 包
02 import java.awt.*;             //导入 AWT 包
03 import java.awt.event.*;        //导入事件包
04 public class KongJian1 extends JFrame implements ActionListener
05 {
06     JTextField jtf=new JTextField(10); //创建文本框
07     JLabel jl=new JLabel();          //创建标签
08     JPanel jp=new JPanel();         //创建面板
09     public KongJian1()
10     {
11         this.setTitle("创建文本框");    //设置窗体名称
12         jtf.addActionListener(this);    //为文本框注册监听器
13         jp.add(jtf);                 //将按钮添加到面板中
14         jp.add(jl);                  //将标签添加到面板中
15         this.add(jp);                //将面板添加到窗体中
16         this.setBounds(300,250,300,200); //设置窗体的位置和大小
17         this.setVisible(true);        //设置窗体可见性
18     }
19     public void actionPerformed(ActionEvent e)
20     {
21         String s=jtf.getText();       //获取文本框的内容
22         jl.setText("你输入的内容为"+s); //设置标签的显示内容
23     }
24     public static void main(String args[])
25     {
26         KongJian1 kj=new KongJian1();
27     }
28 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-1 所示。

在图 16-1 所示的运行结果中，输入内容，例如输入“你好”，进行回车操作，运行结果如图 16-2 所示。



图 16-1 创建文本框

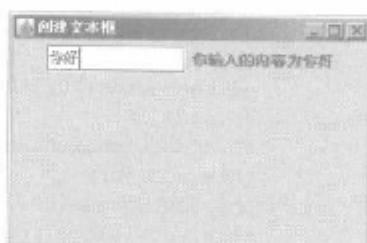


图 16-2 发生事件

【代码解析】在本程序中创建了文本框，同时指定该文本框的长度为 10。为文本框注册了监听器，当在文本框中进行回车操作时，将执行监听方法。在监听方法中将首先获取文本框中的信息，并将该信息作为在标签内容的一部分显示在窗体中。

16.1.2 创建密码框

密码框是文本框的改进的控件，是一种专门用于输入密码的文本框。在文本框中输入信息后，将不直接显示输入的信息，而是使用特定的特殊字符来进行显示。密码框是通过



JPasswordField 类来创建的，因为密码框和文本框的关系，所以 JPasswordField 类的构造器是和 JTextField 类的构造器相同的。



注意：在文本框中输入信息后，将不直接显示输入的信息，而是使用特定的特殊字符来进行显示。

【范例 16-2】示例代码 16-2 是一个创建密码框的程序。

示例代码 16-2

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.*;
03 import java.awt.event.*;
04 public class KongJian2 extends JFrame implements ActionListener
05 {
06     JTextField jtf=new JTextField(10); //创建文本框
07     JPasswordField jpf=new JPasswordField(10); //创建密码框
08     JLabel j11=new JLabel("用户名"); //创建标签
09     JLabel j12=new JLabel("密码"); //创建标签
10     JButton jb=new JButton("提交"); //创建按钮
11     JLabel jl=new JLabel(); //创建标签
12     JPanel jp=new JPanel(); //创建面板
13     public KongJian2()
14     {
15         this.setTitle("创建文本框"); //设置窗体名称
16         jp.setLayout(null); //设置面板为空布局
17         j11.setBounds(30,20,80,30); //设置用户名标签位置和大小
18         jp.add(j11); //将用户名标签添加到面板中
19         j12.setBounds(30,70,80,30); //设置密码标签位置和大小
20         jp.add(j12); //将密码标签添加到面板中
21         jtf.setBounds(80,20,180,30); //设置文本框的位置和大小
22         jp.add(jtf); //将文本框添加到面板中
23         jpf.setBounds(80,70,180,30); //设置密码框的位置和大小
24         jp.add(jpf); //将密码框添加到面板中
25         jb.setBounds(50,130,80,30); //设置按钮的位置和大小
26         jp.add(jb); //将按钮添加到面板中
27         jl.setBounds(10,180,300,30); //设置标签的位置和大小
28         jp.add(jl); //将标签添加到面板中
29         jb.addActionListener(this); //为按钮注册监听器
30         this.add(jp); //将面板添加到窗体中
31         this.setBounds(300,250,350,250); //设置窗体的位置和大小
32         this.setVisible(true); //设置窗体可见性
33     }
34     public void actionPerformed(ActionEvent e) //监听方法
35     {
36         String s1=jtf.getText(); //获取文本框内容
37         String s2=jpf.getText(); //获取密码框内容
38         jl.setText("你的用户名为"+s1+"密码为"+s2); //设置标签内容
39     }
40     public static void main(String args[])
41     {
42         KongJian2 kj=new KongJian2();
43     }
44 }
```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-3 所示。

在图 16-3 所示的运行结果中输入内容，例如输入用户名为“pYq”，密码为“1234”，单击“提交”按钮，运行结果如图 16-4 所示。



图 16-3 创建密码框



图 16-4 发生事件后

【代码解析】 在该程序中，同时定义了一个用户名文本框和一个表示密码的密码框，从而初步实现一个用户名登录的界面。同时在界面中还定义了一个提交按钮，并且为该按钮注册了监听器。当单击提交按钮后，就会执行监听方法。在监听方法中，首先获取文本框和密码框中的内容，然后作为标签显示内容的一部分。在本程序中主要看密码框和文本框的区别，在图 16-4 中可以看到密码框中是不直接显示内容的。从程序中同样也可以看出，密码框和文本框的使用和操作都是相同的。

16.1.3 创建多行文本框

当用户希望进行多行输入时，文本框就不能满足其要求，这时候就需要创建多行文本框。多行文本框也是文本框的一种特殊形式，多行文本框是通过 JTextArea 类实现的。JTextArea 类中提供了 6 种构造器来创建多行文本框，构造器如表 16-2 所示。

表 16-2 JTextArea 类构造器

构造器	说明
public JTextArea()	创建一个没有内容的 JTextArea，默认的行数和列数为 0
public JTextArea(String text)	创建一个具有默认内容的 JTextArea，行数和列数为 0
public JTextArea(int rows,int columns)	创建一个具有指定的行和列的 JTextArea，参数 rows 与 columns 分别表示指定的行与列
public JTextArea(String text,int rows,int columns)	创建一个具有默认内容及行数和列数的 JTextArea，参数 text 为指定的文本内容，参数 rows 与 columns 分别表示指定的行数与列数
public JTextArea(Document doc)	创建一个具有指定文档模型的 JTextArea，行数和列数为 0
public JTextArea(Document doc,String text,int rows,int columns)	创建一个具有指定文档模型、内容、行与列的 JTextArea

在这些构造器中通常使用第三种构造器来创建多行文本框，使用构造器将创建一个指定行数和列数的多行文本框。

【范例 16-3】 示例代码 16-3 是一个创建多行文本框的程序。

示例代码 16-3

```
import javax.swing.*; // 导入 swing 包
import java.awt.*; // 导入 AWT 包
```



```

02 import java.awt.event.*;
03 public class KongJian3 extends JFrame           //导入事件包
04 {
05     JTextArea jta=new JTextArea(6,10);          //创建多行文本框
06     JLabel jl=new JLabel("多行文本框");          //创建标签
07     JPanel jp=new JPanel();                     //创建面板
08     public KongJian3()
09     {
10         this.setTitle("创建文本框");             //设置窗体名称
11         jp.add(jta);                           //将多行文本框添加到面板中
12         jp.add(jl);                           //将标签添加到面板中
13         this.add(jp);                          //将面板添加到窗体中
14         this.setBounds(300,250,300,200);        //设置窗体的位置和大小
15         this.setVisible(true);                 //设置窗体可见性
16     }
17     public static void main(String args[])
18     {
19         KongJian3 kj=new KongJian3();
20     }
21 }

```



图 16-5 创建多行文本框

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-5 所示。

【代码解析】在该程序的第 5 行，创建了一个简单的具有 6 行 10 列的文本框，将该文本框添加到面板中后出现图 16-5 的运行结果。多行文本框是不存在事件的，它通常和其他控件配合实现功能。



16.2 复选框和单选按钮

复选框和单选按钮具有很多相似的地方，它们在实际开发中也经常要使用到。例如需要用户来选择兴趣爱好时，考虑一般人都很少只有一个爱好，这里就可以创建复选框来实现其功能，让用户进行多项选择操作。但是例如性别等信息，是不可能存在多个选择的，它只能在有限的几个选项中选择其中一个，这里就可以使用单选按钮。

16.2.1 创建单选按钮

单选按钮是一种只能在一组选项中选择其中一个选项的控件。单选按钮是通过使用 JRadioButton 类来创建的，在 JRadioButton 类中具有 7 种构造器形式，构造器如表 16-3 所示。

表 16-3 JRadioButton类构造器

构造器	说明
public JRadioButton()	创建一个没有文本与图标并且未被选定的单选按钮
public JRadioButton(Icon icon)	创建一个具有指定图标默认没有选中的单选按钮
public JRadioButton(Icon icon,boolean selected)	创建一个具有指定图标的单选按钮。若 selected 为 true，则该单选按钮处于选中状态，反之则为未选中状态
public JRadioButton(String text)	创建一个具有指定文本默认没有选中的单选按钮，参数 text 为指定的文本

续表

构造器	说 明
public JRadioButton(String text,boolean selected)	创建一个具有指定文本的单选按钮，参数 text 为指定的文本。若 selected 为 true，则该单选按钮处于选中状态，反之则为未选中状态
public JRadioButton(String text,Icon icon)	创建一个具有指定文本和图标默认没有选中的单选按钮，参数 text 为指定的文本，参数 icon 为指定的图标
public JRadioButton(String text,Icon icon,boolean selected)	创建一个具有指定文本和图标的单选按钮，参数 text 为指定的文本，参数 icon 为指定的图标

在通常情况下，创建单选按钮都是通过 public JRadioButton(String text) 构造器来定义的。只使用 JRadioButton 类只是创建一个单选按钮，但是要创建一组单选按钮，还需要使用 ButtonGroup 类来创建。ButtonGroup 类只定义了一个无参构造器来创建一个单选按钮组。



注意：只使用 JRadioButton 类只是创建一个单选按钮，但是要创建一组单选按钮，还需要使用 ButtonGroup 类来创建。

在 ButtonGroup 类中具有几个非常有用的方法。使用 add 方法可以将指定的单选按钮添加到单选按钮组中。使用 remove 方法可以将指定的单选按钮从单选按钮组中删除。使用 getButtonCount 方法可以获取组中单选按钮的个数。单选按钮都是配合组来使用的，单独的单选按钮是没有意义的。

【范例 16-4】示例代码 16-4 是一个创建单选按钮的程序。

示例代码 16-4

```

import javax.swing.*;                                //导入 Swing 包
01 import java.awt.*;                                //导入 AWT 包
02 import java.awt.event.*;                            //导入事件包
03 public class KongJian4 extends JFrame
04 {
05     JRadioButton jrb1=new JRadioButton("男");          //创建一个表示男的单选按钮
06     JRadioButton jrb2=new JRadioButton("女");          //创建一个表示女的单选按钮
07     ButtonGroup bg=new ButtonGroup();                //创建一个单选按钮组
08     JPanel jp=new JPanel();                          //创建面板
09     public KongJian4()
10     {
11         this.setTitle("创建单选按钮");                //设置窗体名称
12         bg.add(jrb1);                                //将单选按钮放到组中
13         bg.add(jrb2);                                //将单选按钮添加到面板中
14         jp.add(jrb1);                                //将面板添加到窗体中
15         jp.add(jrb2);
16         this.add(jp);                                //设置窗体的位置和大小
17         this.setBounds(300,250,300,200);
18         this.setVisible(true);                         //设置窗体可见性
19     }
20     public static void main(String args[])
21     {
22         KongJian4 kj=new KongJian4();
23     }
24 }
```

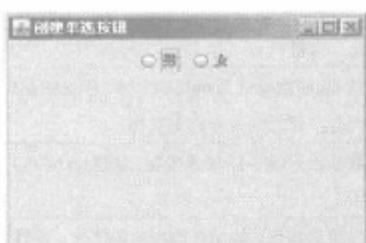


图 16-6 创建单选按钮

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-6 所示。

【代码解析】在该程序的第 5 行和第 6 行创建了两个分别表示“男”和“女”的单选按钮，接下来就是创建了一个单选按钮组。在程序的第 12 行和第 13 行，首先要将创建的单选按钮放到单选按钮组中，这一步是很关键的，读者可以自己实验如果没有这两条语句的状况。

在单选按钮中同样也是会发生 ActionEvent 事件的，当进行选择操作时就会触发该事件。除了 ActionEvent 事件外，单选按钮还有可能发生 ItemEvent 事件。ItemEvent 事件的发生条件是当选项的状态发生改变时才会触发。在单选按钮中两次选择同一个选项是不会触发 ItemEvent 事件的，只有从未选中改变为选中状态，或者从选中状态改变为未选中状态，才会触发 ItemEvent 事件。



注意：ItemEvent 事件的发生条件是当选项的状态发生改变时才会触发。在单选按钮中两次选择同一个选项是不会触发 ItemEvent 事件的。

ItemEvent 事件是通过 ItemListener 监听接口来实现的，在 ItemListener 监听接口中具有一个 itemStateChanged 方法，当发生 ItemEvent 事件时就会执行该方法。

【范例 16-5】示例代码 16-5 是一个单选按钮触发 ItemEvent 事件的程序。

示例代码 16-5

```
01 import javax.swing.*; //导入 Swing 包
02 import java.awt.*; //导入 AWT 包
03 import java.awt.event.*; //导入事件包
04 public class KongJian5 extends JFrame implements ItemListener
05 {
06     JRadioButton jrb1=new JRadioButton("男",true); //创建一个表示男的单选按钮
07     JRadioButton jrb2=new JRadioButton("女"); //创建一个表示女的单选按钮
08     ButtonGroup bg=new ButtonGroup(); //创建一个单选按钮组
09     JPanel jp=new JPanel(); //创建面板
10     public KongJian5()
11     {
12         this.setTitle("创建单选按钮"); //设置窗体名称
13         bg.add(jrb1); //将单选按钮放到组中
14         bg.add(jrb2);
15         jrb1.addItemListener(this); //为单选按钮注册监听器
16         jrb2.addItemListener(this);
17         jp.add(jrb1); //将单选按钮添加到面板中
18         jp.add(jrb2);
19         this.add(jp); //将面板添加到窗体中
20         this.setBounds(300,250,300,200); //设置窗体的位置和大小
21         this.setVisible(true); //设置窗体可见性
22     }
23     public void itemStateChanged(ItemEvent e) //监听方法
24     {
25         System.out.println("选项发生改变");
26     }
}
```

```

27     public static void main(String args[])
28     {
29         KongJian5 kj=new KongJian5();
30     }
31 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-7 所示。

在图 16-7 的运行结果中选择“女”，则后台的运行结果如图 16-8 所示。



图 16-7 单选按钮触发事件



图 16-8 发生事件

【代码解析】从后台的运行结果中可以看出，当进行一次选择后，在后台显示两条语句。这是因为两个选项的状态都发生了改变。在示例代码 16-5 中的第 15 行和第 16 行为单选按钮注册了事件监听器，当单选按钮的状态发生改变后将执行 itemStateChanged 方法，在后台显示选项发生改变。

16.2.2 创建复选框

通过 JRadioButton 与 ButtonGroup 的配合使用，可以很方便地实现单项选择。若需要使用多项选择，则应该使用复选框——JCheckBox 类。与 JRadioButton 的不同是，JCheckBox 不需要编组使用，各个选项之间没有逻辑约束关系。



提示：通过 JRadioButton 与 ButtonGroup 的配合使用，可以很方便地实现单项选择。

该类提供了 8 个构造器，表 16-4 中列出了其中 7 个比较常用的。

表 16-4 JCheckBox 类的常用构造器

构造器	说明
public JCheckBox()	创建一个没有文本与图标并且未被选定的复选框
public JCheckBox(Icon icon)	创建一个具有指定图标默认未被选中的复选框，参数 icon 为指定的图标
public JCheckBox(Icon icon,boolean selected)	创建一个具有指定图标的复选框，参数 selected 表示复选框的选中状态。若 selected 为 true 则处于选中状态，否则处于未选中状态
public JCheckBox(String text)	创建一个具有指定文本默认未被选中的复选框，参数 text 为指定的文本
public JCheckBox(String text,boolean selected)	创建一个具有指定文本的复选框，参数 selected 表示复选框的选中状态。若 selected 为 true 则处于选中状态，否则处于未选中状态
public JCheckBox(String text,Icon icon)	创建一个具有指定文本和图标默认未被选中的复选框，参数 text 为指定的文本，参数 icon 为指定的图标



续表

构造器	说 明
public JCheckBox (String text,Icon icon,boolean selected)	创建一个具有指定文本和图标的复选框，参数 text 为指定的文本，参数 icon 为指定的图标，参数 selected 表示复选框的选中状态。若 selected 为 true 则处于选中状态，否则处于未选中状态

【范例 16-6】示例代码 16-6 是一个创建复选框的程序。

示例代码 16-6

```

import javax.swing.*;                                //导入 Swing 包
import java.awt.*;                                 //导入 AWT 包
import java.awt.event.*;                            //导入事件包
public class KongJian6 extends JFrame implements ItemListener
01 {
02     JCheckBox jcb1=new JCheckBox("游泳");           //创建一个表示爱好的复选框
03     JCheckBox jcb2=new JCheckBox("上网");           //创建一个表示爱好的复选框
04     JCheckBox jcb3=new JCheckBox("看书");           //创建一个表示爱好的复选框
05     JPanel jp=new JPanel();                      //创建面板
06     public KongJian6() {
07     }
08     this.setTitle("创建单选按钮");                  //设置窗体名称
09     jcb1.addItemListener(this);                     //为复选框注册监听器
10     jcb2.addItemListener(this);
11     jcb3.addItemListener(this);
12     jp.add(jcb1);                                //将复选框添加到面板中
13     jp.add(jcb2);
14     jp.add(jcb3);
15     this.add(jp);                                //将面板添加到窗体中
16     this.setBounds(300,250,300,200);              //设置窗体的位置和大小
17     this.setVisible(true);                         //设置窗体可见性
18 }
19 public void itemStateChanged(ItemEvent e) {        //监听方法
20     System.out.println("选项发生改变");
21 }
22 public static void main(String args[]) {
23     KongJian6 kj=new KongJian6();
24 }
25 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-9 所示。

在界面中进行操作，后台的运行结果如图 16-10 所示。



图 16-9 创建复选框



图 16-10 后台结果



【代码解析】从后台的运行结果中可以看出，当进行一次选择后，在后台显示一条语句。当选中一个选项时就会在后台显示一条语句，如果将该复选框从选中状态变为未选中状态，同样显示该语句。这是因为复选框的状态发生变化时都会触发事件，从而执行监听方法。

16.3 选项卡

选项卡也是开发 GUI 界面常用的控件之一，通过使用选项卡可以在同一个窗体中提供很多不同的界面，可以通过选项卡提供的标签在界面间方便地进行切换。本节将为读者详细介绍如何使用 Swing 中的选项卡，主要包括 JTabbedPane 类、ChangeEvent 事件，以及具体案例等内容。

16.3.1 选项卡介绍

选项卡可以产生多个标签框架，每一个标签框架窗口自成一个系统，即包含多个页面，每个页面与一个标签对应。当选择某一个标签时，标签框架窗口会自动显示出此标签框架的内容，并触发一个 ChangeEvent 事件，这个事件由 ChangeListener 监听器监听并处理。



注释：每次只能选择标签组的一个标签。

选项卡具有以下构造方法及常用方法。

- public JTabbedPane()方法：该方法创建一个 TabbedPane 对象，该对象具有默认的 JTabbedPane.TOP 选项卡布局。
 - public JTabbedPane(int tabPlacement)方法：该方法使用指定的参数 tabPlacement 创建一个 TabbedPane 对象，tabPlacement 参数有常值，JTabbedPane.TOP、JTabbedPane.BOTTOM、JTabbedPane.LEFT 或 JTabbedPane.RIGHT。
 - public JTabbedPane(int tabPlacement, int tabLayoutPolicy)方法：该方法以指定的参数 tabPlacement、tabLayoutPolicy 创建一个 TabbedPane 对象，该对象具有指定的 tabPlacement 选项卡布局和 tabLayoutPolicy 选项卡布局策略。其中参数 tabLayoutPolicy 有常值，JTabbedPane.WRAP_TAB_LAYOUT、JTabbedPane.SCROLL_TAB_LAYOUT。
- 选项卡中的常用方法定义如下所示。

```
void addTab(String str, Component comp)
```

其中，str 是标签的标题，comp 是应加入标签的组件。通常情况下，加入的是 JPanel 或其子类。

16.3.2 创建选项卡

学习了 JTabbedPane 类的构造器和相关方法后就可以来创建选项卡。选项卡的创建和前面控件的创建都是很类似的。

【范例 16-7】示例代码 16-7 是一个创建选项卡的程序。

示例代码 16-7

```
01 import javax.swing.*;
02 import javax.swing.event.*;
```



```
03 import java.awt.*;
04 public class KongJian7 extends JFrame implements ChangeListener
05 {
06     //创建三个选项卡窗格
07     private JTabbedPane jtp1=new JTabbedPane(JTabbedPane.LEFT);
08     private JTabbedPane jtp2=new JTabbedPane(JTabbedPane.TOP);
09     private JTabbedPane jtp3=new JTabbedPane(JTabbedPane.RIGHT);
10     //创建标签
11     JLabel jl=new JLabel("您选中了换行方式的选项卡 0。",JLabel.CENTER);
12     public KongJian7()
13     {
14         //分别将选项卡窗格 jtp2 与 jtp3 添加进 jtp1
15         jtp1.addTab("换行方式",jtp2);
16         jtp1.addTab("滚动方式",jtp3);
17         //为选项卡窗格 jtp2 与 jtp3 设置选项卡标签超过一行以后的处理策略
18         jtp2.setLayoutPolicy(JTabbedPane.WRAP_TAB_LAYOUT);
19         jtp3.setLayoutPolicy(JTabbedPane.SCROLL_TAB_LAYOUT);
20         //为选项卡面板 jtp2 与 jtp3 各添加 20 个选项卡
21         for(int i=0;i<5;i++)
22         {
23             jtp2.addTab("Tab"+i,new JLabel("这里是选项卡"+i,JLabel.CENTER));
24             jtp3.addTab("Tab"+i,new JLabel("这里是选项卡"+i,JLabel.CENTER));
25         }
26         //将选项卡窗格 jtp1 添加进窗体
27         this.add(jtp1);
28         //将标签添加进窗体
29         this.add(jl,BorderLayout.SOUTH);
30         //为三个选项卡窗格注册 ChangeEvent 事件监听器
31         jtp1.addChangeListener(this);
32         jtp2.addChangeListener(this);
33         jtp3.addChangeListener(this);
34         //设置窗体的标题、大小位置及可见性
35         this.setTitle("选项卡示例");
36         this.setBounds(100,100,500,200);
37         this.setVisible(true);
38         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39     }
40     //实现 ChangeListener 监听接口中的事件处理方法
41     public void stateChanged(ChangeEvent e)
42     {
43         //获取外层选项卡窗格当前选中的选项卡索引
44         int indexOuter=jtp1.getSelectedIndex();
45         //获取内层选项卡窗格当前选中的选项卡索引
46         int indexInner=
47             ((JTabbedPane)jtp1.getSelectedComponent()).getSelectedIndex();
48         //设置标签的内容
49         jl.setText("您选中了 "+jtp1.getTitleAt(indexOuter)+" 选项卡 "+index
50         inner+"。");
51     }
52     public static void main(String[] args)
53     {
54         //创建 Sample23_1 窗体对象
55         KongJian7 kj=new KongJian7();
56     }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-11 所示。

【代码解析】在该程序中为每一个选项卡窗格注册了 ChangeEvent 事件监听器。当选择一个选项卡后，将触发事件，从而执行监听方法。事件处理方法中根据选中选项卡的变化动态修改标签的文本内容。

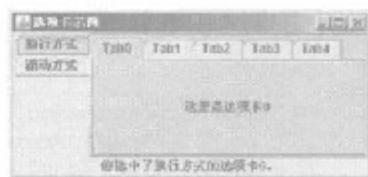


图 16-11 创建选项卡

16.4 分隔窗格

分隔窗格 (JSplitPane) 也是 Swing 中常用的控件之一，其能够将单个空间分隔成两个部分，并在两个部分中显示不同的内容，本节将为读者详细介绍 JSplitPane 类的相关知识与使用。

16.4.1 分隔窗格介绍

JSplitPane 控件允许在单个空间中放置两个控件，开发人员可以自由决定按水平方向或垂直方向划分空间，还可以在程序运行期间使用鼠标自由调整空间的分隔比例。通过 JSplitPane 控件的嵌套使用，可以将空间分隔成更多的部分。JSplitPane 类提供了 5 个构造器，如表 16-5 所示。JSplitPane 类中表示分隔方向的常量如表 16-6 所示。

表 16-5 JSplitPane类的构造器

构造器	说明
public JSplitPane()	创建水平分隔的分隔窗格，并自动在左右两个空间中各添加一个按钮控件来填充
public JSplitPane(int newOrientation)	创建按指定方向分隔的分隔窗格，参数 newOrientation 表示分隔的方向
public JSplitPane(int newOrientation, boolean newContinuousLayout)	创建按指定方向分隔的分隔窗格，参数 newOrientation 表示分隔的方向，参数 newContinuousLayout 决定当分隔条改变位置时控件是否连续重绘
public JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)	创建按指定方向分隔的分隔窗格，参数 newOrientation 表示分隔的方向。参数 newLeftComponent 与 newRightComponent 分别为左（上）边与右（下）边的指定控件
public JSplitPane(int newOrientation, boolean newContinuousLayout, Component newLeftComponent, Component newRightComponent)	创建按指定方向分隔的分隔窗格，参数 newOrientation 表示分隔的方向，参数 newContinuousLayout 决定当分隔条改变位置时控件是否连续重绘。参数 newLeftComponent 与 newRightComponent 分别为左（上）边与右（下）边的指定控件

表 16-6 JSplitPane 类中表示分隔方向的常量

常量名称	说明
VERTICAL_SPLIT	表示按垂直方向将空间分为上下两个部分，每部分可以放置一个控件
HORIZONTAL_SPLIT	表示按水平方向将空间分为左右两个部分，每部分可以放置一个控件

16.4.2 创建分隔窗格

通过上一节的介绍，对 JSplitPane 控件有了大体的了解，其能够将窗格分隔成两个部分，且只能是两个部分，但是可以通过嵌套使用的方式将窗格分隔成任意多份。本节将给出一个嵌套使用 JSplitPane 控件的例子，进一步加深读者对该控件的理解。



【范例 16-8】示例代码 16-8 是一个创建分隔窗格的程序。

示例代码 16-8

```
00 import javax.swing.*;
01 //定义该类继承自 JFrame
02 public class KongJian8 extends JFrame
03 {
04     //创建两个不同分隔方向的 JSplitPane 对象
05     private JSplitPane jspOuter=new JSplitPane(JSplitPane.HORIZONTAL_
SPLIT,true);
06     private JSplitPane jspInner=new JSplitPane(JSplitPane.VERTICAL_
SPLIT,true);
07     public KongJian8()
08     {
09         //将标签设置到 jspOuter 的左边窗格中
10         jspOuter.setLeftComponent(new JLabel("这里是左边窗格",JLabel.
CENTER));
11         //将 jspinner 设置到 jspOuter 的右边窗格中
12         jspOuter.setRightComponent(jspInner);
13         //设置 jspinner 控件中的两个子控件
14         jspInner.setTopComponent(new JLabel("这里是右上窗格",JLabel.
CENTER));
15         jspInner.setBottomComponent(new JLabel("这里是右下窗格",JLabel.
CENTER));
16         //设置 jspinner 中分隔条的初始位置
17         jspinner.setDividerLocation(150);
18         //设置两个分隔条的宽度（高度）
19         jspOuter.setDividerSize(4);
20         jspInner.setDividerSize(4);
21         //将 jspOuter 添加进窗体
22         this.add(jspOuter);
23         //设置窗体的标题、大小位置以及可见性
24         this.setTitle("分隔窗格示例");
25         this.setResizable(false);
26         this.setBounds(00,100,425,250);
27         this.setVisible(true);
28         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29     }
30     public static void main(String[] args)
31     {
32         //创建 Sample23_2 窗体对象
33         KongJian8 kj=new KongJian8();
34     }
35 }
```

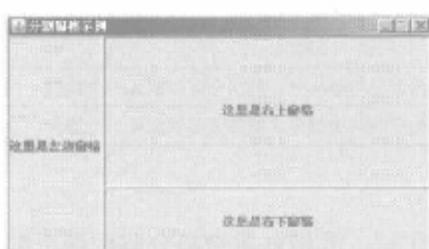


图 16-12 创建分隔窗格

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-12 所示。

【代码解析】在该程序中使用了嵌套的分隔窗格，首先将整个窗体分隔为左右窗格，然后将右边的窗格分隔为上下窗格。使用这种嵌套的方法可以将窗体分为很多种形式，同样也能将窗体分为任意多份。这样就能使界面的形式更加丰富。



16.5 滑块和进度条

很多 GUI 应用程序中都通过滑块来让用户进行指定范围值的输入，这样很方便，而且用户也不再可能输入错误的数值，使界面变得很友好。通过进度条系统能够向用户即时反馈一些信息，避免用户不知道系统处于何种状态而焦急的等待。本节将向读者介绍如何使用 Swing 中提供的滑块与进度条，通过本节的学习，读者也可以方便地开发出使用滑块与进度条的应用。

16.5.1 创建滑块

JSlider 类是 Swing 包中提供的用于实现滑块的控件，通过 JSlider 控件可以让用户在限定的范围内方便地选择需要的值。JSlider 类提供的滑块可以是水平方向的，也可以是垂直方向的，并且可以根据需要设置成为不同的外观风格。



提示：滑块组件是由可以拖动的滑块和一个范围组件组成的；用户可以通过拖动滑块在一个区间范围里进行选择。

```
JSlider ageSlider = new JSlider();
ageSlider = new JSlider(SwingConstants.VERTICAL, 0, 120, 20);
```

当滑块被拖动时，滑块的值将发生变化，触发 ChangeEvent 事件。滑块组件的事件监听器要实现 ChangeListener 接口，并实现此接口中的 stateChanged 方法，示例代码如下所示。

```
AgeListener myAgeListener = new AgeListener();
ageSlider.addChangeListener(myAgeListener);
private class AgeListener implements ChangeListener
{
    public void stateChanged(ChangeEvent event)
    {
        ...
    }
}
```

滑块值发生变化时，使用 getValue 方法来获得滑块值，示例代码如下所示。

```
myTextField.setText("") + sourceSlider.getValue());
```

【范例 16-9】示例代码 16-9 是一个创建滑块的程序。

示例代码 16-9

```
01 import java.awt.*;
02 import javax.swing.*;
03 import javax.swing.event.*;
04 public class KongJian9 extends JFrame
05 {
06     int WIDTH = 400;
07     int HEIGHT = 280;
08     JLabel choosedLabel = new JLabel("Choosed age:");
09     JLabel choiceLabel = new JLabel("Please choose age:");
10     private JTextField myTextField;
11     JPanel agePanel;
12     public KongJian9() {
13         setTitle("JSliderUseDemo");
14         setSize(WIDTH, HEIGHT);
15         Container contentPane = getContentPane();
```



```
16     AgeListener myAgeListener = new AgeListener();
17     // 建立容纳滑块的面板
18     agePanel = new JPanel();
19     // 新建默认样式的滑块
20     JSlider ageSlider = new JSlider();
21     ageSlider.addChangeListener(myAgeListener);
22     agePanel.add(choiceLabel);
23     agePanel.add(ageSlider);
24     // 新建竖向滑块，并指定最大值和初始值
25     ageSlider = new JSlider(SwingConstants.VERTICAL, 0, 100, 20);
26     ageSlider.addChangeListener(myAgeListener);
27     agePanel.add(ageSlider);
28     // 建立容纳文本域的面板
29     JPanel textPanel = new JPanel();
30     // 新建文本域
31     myTextField = new JTextField("", 15);
32     textPanel.add(choosedLabel, BorderLayout.NORTH);
33     textPanel.add(myTextField, BorderLayout.CENTER);
34     contentPane.add(agePanel, BorderLayout.NORTH);
35     contentPane.add(textPanel, BorderLayout.CENTER);
36 }
37 // 两个滑块共用的事件监听器
38 private class AgeListener implements ChangeListener {
39     public void stateChanged(ChangeEvent event) {
40         JSlider sourceSlider = (JSlider) event.getSource();
41         myTextField.setText("+" + sourceSlider.getValue());
42     }
43 }
44 public static void main(String[] args) {
45     KongJian9 frame = new KongJian9();
46     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47     frame.setVisible(true);
48 }
49 }
```

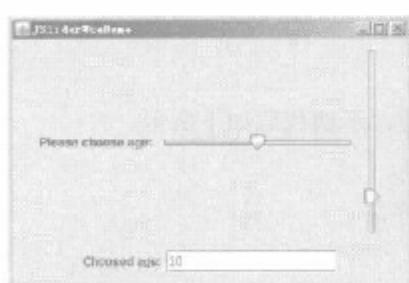


图 16-13 创建滑块

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-13 所示。

【代码解析】在该程序中创建了两个滑块，一个水平放置的滑块，一个垂直放置的滑块。在运行结果中对滑块进行操作，都会在下面的文本框中显示滑块上面的刻度。这是因为在程序中为滑块注册了监听器。当滑块的刻度发生变化后将触发事件，从而执行监听方法，也就在文本框中显示当前操作的滑块的刻度。

16.5.2 创建进度条

JProgressBar 是 Swing 中提供的用来实现进度条的控件，使用其可以非常方便地完成进度条的开发。在应用中恰当使用进度条可以即时通告用户系统的一些信息，避免用户因不知道系统运行情况而焦急地等待，从而使界面更加友好。



注意：Swing 中提供的进度条不但可以像常见的进度条一样显示工作的进度，而且可以通过设置为模糊模式以动画形式来表示系统正在运行。

JProgressBar 类提供了 5 个构造器，其中有 4 个是比较常用的，表 16-7 列出了这 4 个常用的构造器。

表 16-7 JProgressBar类的常用构造器

构造器	功能
public JProgressBar()	创建一个显示边框但不显示信息字符串的水平进度条，初始值和最小值都为 0，最大值为 100
public JProgressBar(int orient)	创建一个显示边框但不显示信息字符串并且具有指定方向的进度条，初始值和最小值都为 0，最大值为 100，参数 orient 指定了进度条的方向
public JProgressBar(int min,int max)	创建一个显示边框但不显示信息字符串并且具有指定最小值和最大值的水平进度条，进度条的初始值设置为指定的最小值，参数 min 与参数 max 分别为指定的最小值与最大值
public JProgressBar(int orient,int min,int max)	创建一个显示边框但不显示信息字符串并且具有指定最小值和最大值及指定方向的进度条，参数 orient 指定了进度条的方向，参数 min 与参数 max 分别为指定的最小值与最大值

表 16-7 中的参数 orient 表示进度条的方向（水平或垂直），其值一般使用 JProgressBar 中的静态常量来表示，如表 16-8 所示。

表 16-8 表示进度条方向的常量值

常量名	描述	常量名	描述
VERTICAL	该值表示垂直方向	HORIZONTAL	该值表示水平方向

同时 JProgressBar 类中还提供了大量的方法，开发人员可以通过这些方法对进度条进行各种操作，表 16-9 中列出了该类中的一些常用方法。

表 16-9 JProgressBar类中的常用方法

方法	功能
public int getOrientation()	该方法将返回进度条的方向，返回值为表 16-8 所列的值之一
public void setOrientation(int newOrientation)	设置进度条的方向，参数 newOrientation 为指定的方向值，为表 16-8 所列的值之一
public void setStringPainted(boolean b)	设置进度条是否应该呈现信息字符串，若参数 b 为 true，则将呈现信息字符串，否则不呈现信息字符串
public boolean isStringPainted()	返回进度条呈现信息字符串的情况，若返回值为 true 则呈现信息字符串，否则不呈现信息字符串
public void setString(String s)	设置进度条要显示的信息字符串，如果没有设置则默认值为进度条当前进度的百分比形式，例如“20%”，参数 s 为指定的字符串
public double getPercentComplete()	返回进度条完成的百分比，其返回值应该是在 0.0~1.0 之间的 double 值
public int getValue()	该方法将返回进度条的当前值
public void setValue(int n)	设置进度条的当前值，参数 n 为指定的当前值
public int getMinimum()	返回进度条的最小值
public void setMinimum(int minimum)	设置进度条的最小值，参数 minimum 为指定的最小值
public int getMaximum()	该方法将返回进度条的最大值



续表

方 法	功 能
public void setMaximum(int maximum)	设置进度条的最大值，参数 maximum 为指定的最大值
public void setIndeterminate(boolean newValue)	该方法将设置进度条是否处于模糊模式，当参数 newValue 为 true 时，则表示设置为模糊模式，否则表示设置为普通模式。
public boolean isIndeterminate()	该方法将确定进度条是否处于模糊模式下，若返回值为 true 则是，否则则不是。

一旦进度条的当前值被修改，就会触发一个 ChangeEvent 事件，因此也可以给进度条注册 ChangeEvent 事件监听器。

【范例 16-10】示例代码 16-10 是一个创建进度条的程序。

示例代码 16-10

```

01 import javax.swing.*;
02 import javax.swing.event.*;
03 //定义该类继承自 JFrame
04 public class KongJian10 extends JFrame
05 {
06     //创建 JPanel 容器
07     private JPanel jp=new JPanel();
08     //创建进度条与滑块
09     private JProgressBar jpb=new JProgressBar(0,1000);
10     //创建标签数组
11     private JLabel[] jlArray={new JLabel("进度指示条")};
12     public KongJian10()
13     {
14         //为容器 JPanel 设置布局管理器
15         jp.setLayout(null);
16         //设置标签的大小位置，并将标签添加进 JPanel 容器
17         for(int i=0;i<jlArray.length;i++)
18         {
19             jlArray[i].setBounds(20,20+i*100,80,30);
20             jp.add(jlArray[i]);
21         }
22         //设置进度条的大小位置
23         jpb.setBounds(20,150,450,26);
24         //将进度条添加进容器 JPanel
25         jp.add(jpb);
26         //设置进度条的初始值
27         jpb.setValue(500);
28         //设置进度条将显示信息字符串
29         jpb.setStringPainted(true);
30         //将 JPanel 添加进窗体
31         this.add(jp);
32         //设置窗体的标题、大小位置及可见性
33         this.setTitle("进度条示例");
34         this.setBounds(100,100,500,250);
35         this.setVisible(true);
36         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
37     }
38     public static void main(String[] args)
39     {
40         //创建 Sample23_3 窗体对象
41         KongJian10 kj=new KongJian10();

```

```

42      }
43  }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-14 所示。

【代码解析】在该程序中简单的创建了一个进度条，在该程序中并没有为进度条注册事件监听器，进度条通常是和其他控件配合使用的。有兴趣的读者可以写一个让滑块和进度条配合使用的程序。

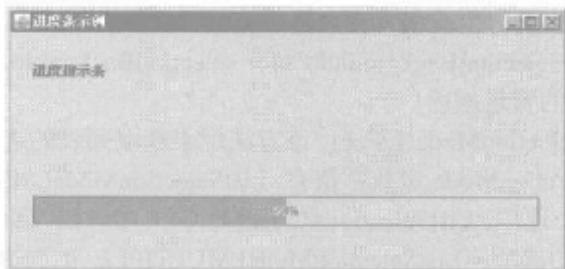


图 16-14 创建进度条

16.6 列表框

很多 GUI 应用程序中都需要让用户从一些选项中选择一项或多项，如果选项不多采用单选按钮或复选框是很方便的，但如果选项比较多就是采用列表框比较合适了。列表是图形用户界面程序中常用到的组件，列表允许用户从列表项中选择一个或多个选项，默认情况下，列表支持单选；选择状态由分隔符来区别，并且可以通过 JList 中提供的 setSelectionMode 方法使得列表支持多选。

16.6.1 列表框介绍

JList 能够为用户提供一组可供选择的选项，这些选项可以以一列或多列的形式显示。默认的选择模式下，可以通过鼠标单击来选择单个选项，也可以在按住特定控制键的同时，单击鼠标来进行多项选择。



注意：JList 类没有提供滚动功能，但是可以通过将其放置在 JScrollPane 中来实现滚动操作。

JList 的常用构造方法如下所示。

- public JList()方法：该方法可以构造一个使用空模型的 JList 对象。
 - public JList(Object[] listData)方法：该方法以显示指定数组中的元素构造一个 JList。
- JList 常用方法如下：
- public void addListSelectionListener(ListSelectionListener listener)方法：该方法为列表加入事件处理方法。
 - public void clearSelection()方法：该方法用于删除列表中所有的选项。
 - public int[] getSelectedIndices()方法：该方法用于获取所有被选取选项的索引位置。



- public Object[] getSelectedValues()方法：该方法用于获取所有被选取选项的值。
- public void setListData(Object[] listData)方法：该方法用于设置列表中的选项。
- public void setSelectedIndices(int[] indices)方法：该方法用于设置列表中被选中的多个选项。
- public void setSelectedValue(Object anObject, boolean shouldScroll)方法：该方法用于设置列表中被选中的单个选项。
- public void setSelectionMode(int selectionMode)方法：该方法用于设置列表的选择模式，如单选、多选。
- public void setSelectionBackground(Color selectionBackground)方法：该方法用于设置被选中选项的背景颜色。
- public int getSelectionMode()方法：该方法用于获取列表的选择模式。

其中选择模式 selectionMode 参数常值有：ListSelectionModel.SINGLE_SELECTION 表示一次只可以选择一个选项；ListSelectionModel.SINGLE_INTERVAL_SELECTION 表示一次可以选择一个连续的选项；ListSelectionModel.MULTIPLE_INTERVAL_SELECTION 表示没有选择限制。

当列表中选中的选项发生变化时会触发 ListSelectionEvent 事件，其是 Swing 中新增的一种事件，全称类名为 javax.swing.event.ListSelectionEvent。因此若希望在选项发生变化时能执行一定的代码，则需要为列表注册 ListSelectionEvent 事件监听器。



提示：当列表中选中的选项发生变化时会触发 ListSelectionEvent 事件，其是 Swing 中新增的一种事件，全称类名为 javax.swing.event.ListSelectionEvent。

ListSelectionEvent 事件的监听器需要实现 javax.swing.event.ListSelectionListener 监听接口，在该监听接口中声明了处理 ListSelectionEvent 事件的方法 valueChanged，该方法原型如下：

```
public void valueChanged(ListSelectionEvent e)
```

16.6.2 创建列表框

通过对 JList 类构造器的学习，读者就可以使用 JList 类构造器来创建列表框。本节将使用 JList 类来创建一个列表框，同时为该列表框注册事件监听器。

【范例 16-11】示例代码 16-11 是创建列表框的程序。

示例代码 16-11

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import javax.swing.event.*;
05 //定义该类继承自 JFrame
06 public class XongJian11 extends JFrame implements ListSelectionListener
07 {
08     //创建列表
09     private JList jl=new JList();
10     //创建 JPanel 对象
11     private JPanel jpy=new JPanel();
```

```

12 //创建滚动窗口
13 private JScrollPane jspz=new JScrollPane(jl);
14 private JScrollPane jpsy=new JScrollPane(jpy);
15 //创建分隔窗格
16 JSplitPane jspw=new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,jspz,jpsy);
17 //创建字符串数组
18 private String[] str=new String[5];
19 //创建复选框数组
20 private JCheckBox[] jcb=new JCheckBox[5];
21 public KongJian11()
22 {
23     //设置分隔窗格中分隔条的初始位置与宽度
24     jspw.setDividerLocation(100);
25     jspw.setDividerSize(4);
26     //设置列表的选择模式使其允许多选
27     jl.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_
SELECTION);
28     //设置列表的数据模型
29     jl.setListData(str);
30     //为列表注册 ListSelectionEvent 监听器
31     jl.addListSelectionListener(this);
32     //设置 JPanel 的布局管理器
33     jpy.setLayout(new GridLayout(4,10));
34     //对复选框数组进行初始化
35     for(int i=0;i<str.length;i++)
36     {
37         str[i]=“选择项”+i;
38         jcb[i]=new JCheckBox(str[i]);
39         jcb[i].setBounds(120+(i%6)*45,10+(i/6)*35,40,30);
40         jcb[i].setEnabled(false);
41         jpy.add(jcb[i]);
42     }
43     //将容器 JSplitPane 添加进窗体
44     this.add(jspw);
45     //设置窗体的标题、大小位置及可见性
46     this.setTitle(“列表示例”);
47     this.setResizable(false);
48     this.setBounds(100,100,360,250);
49     this.setVisible(true);
50     this setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
51 }
52 //实现 ListSelectionListener 接口中的事件处理方法
53 public void valueChanged(ListSelectionEvent e)
54 {
55     //当列表触发事件后，设置复选框的选项
56     for(int i=e.getFirstIndex();i<=e.getLastIndex();i++)
57     {
58         jcb[i].setSelected(((JList)e.getSource()).getSelectedIndex(i));
59     }
60 }
61 public static void main(String[] args)
62 {
63     KongJian11 k1=new KongJian11();
64 }
65 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-15 所示。

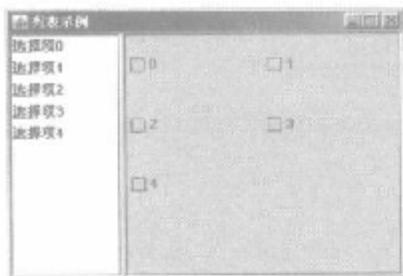


图 16-15 创建列表框

【代码解析】在示例代码 16-11 中使用了一个 JList 列表框与一组 JCheckBox 复选框，当 JList 中选中的选项发生变化时使用对应的 JCheckBox 显示选中选项的情况。列表可以支持多种选择，当选项多的时候，列表框是设置选项的最佳控件。

16.6.3 下拉列表框

下拉列表及组合框，下拉列表与列表不同的是下拉列表只支持单个选项，只允许用户选择一个选项。

优点是能节省空间，使界面更紧凑；并且只有用户单击下拉列表时，列表选项才会显示。



注意：在默认的情况下，下拉列表是不可以被用户编辑的，但是可以使用 JComboBox 提供的方法 setEditable 方法使其可以被编辑。

JComboBox 控件实际上组合了一个文本框与一个下拉列表，在默认情况下 JComboBox 控件提供的文本框是不可编辑的。在文本框旁边有一个包含向下箭头的小按钮，在按下这个按钮之后，会出现显示选项的弹出式列表，用户可以从其中选择需要的选项。

当当前选项发生变化时，下拉列表框会触发 ItemEvent 事件；当用户明确提交一个选中值时，会触发 ActionEvent 事件。当前选项发生变化可能是由于下拉选择选中了另外一个选项，或者在可编辑的情况下输入新值后回车。用户明确提交一个选中值可能是由于下拉选择选中了另外一个选项，或者在可编辑的状态下按下回车键。

【范例 16-12】示例代码 16-12 是一个创建下拉列表框的程序。

示例代码 16-12

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import javax.swing.event.*;
05 import java.util.*;
06 //定义该类继承自 JFrame
07 public class KongJian12 extends JFrame implements ActionListener
08 {
09     //创建 JPanel 对象
10     private JPanel jp=new JPanel();
11     //创建标签数组
12     private JLabel[] jlArray={new JLabel("请选择日期格式"),new JLabel("当前日
期为")};
13     //创建表示下拉列表框数据模型的字符串数组
14     private String[] str={"mm-dd-yyyy","yyyy mm-dd"};
15     //创建下拉列表框
16     private JComboBox jcb=new JComboBox(str);
17     //创建显示结果的文本框
18     private JTextField jtf=new JTextField();
19     //创建描述日期的字符串数组
20     String[] temp=getDate();
21     public KongJian12()
22     {
23         //设置 JPanel 的布局管理器
24         jp.setLayout(null);
```

```

25      //设置标签大小位置，并将标签添加到 JPanel 中
26      for(int i=0;i<jlArray.length;i++)
27      {
28          jlArray[i].setBounds(20,20+i*40,120,30);
29          jp.add(jlArray[i]);
30      }
31      //设置下拉列表框大小位置并将其添加到 JPanel 中
32      jcb.setBounds(120,30,150,26);
33      jp.add(jcb);
34      //为下拉列表框注册动作事件监听器
35      jcb.addActionListener(this);
36      //设置文本框显示的初始内容
37      jtf.setText(temp[1]+"-"+temp[2]+"-"+temp[0]);
38      //设置文本框大小位置并将其添加到 JPanel 中
39      jtf.setBounds(120,60,100,30);
40      jp.add(jtf);
41      //设置文本框为不可编辑状态
42      jtf.setEditable(false);
43      //将容器 JPanel 添加进窗体
44      this.add(jp);
45      //设置窗体的标题、大小位置及可见性
46      this.setTitle("下拉列表框示例");
47      this.setBounds(100,100,500,100);
48      this.setVisible(true);
49      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
50  }
51  //该方法将返回表示年月日的字符串数组
52  public String[] getDate()
53  {
54      String[] date=new String[3];
55      Date d=new Date();
56      //获取表示年份的字符串
57      date[0]=""+(1900+d.getYear());
58      //获取表示月份的字符串
59      date[1]=((d.getMonth()+1)>9)?""+(d.getMonth()+1) :"0"+(d.getMonth()
+1);
60      //获取表示日期的字符串
61      date[2]=(d.getDate()>9)?""+d.getDate():"0"+d.getDate();
62      return date;
63  }
64  //实现 ActionListener 接口中的方法
65  public void actionPerformed(ActionEvent e)
66  {
67      if(jcb.getSelectedIndex()==0)
68      {
69          //当选择下拉列表框中第一项时执行的代码
70          jtf.setText(temp[1]+"-"+temp[2]+"-"+temp[0]);
71      }
72      else if(jcb.getSelectedIndex()==1)
73      {
74          //当选择下拉列表框中第二项时执行的代码
75          jtf.setText(temp[0]+"-"+temp[1]+"-"+temp[2]);
76      }
77  }
78  public static void main(String[] args)
79  {
80      KongJian12 kj=new KongJian12();
81  }
82 }

```

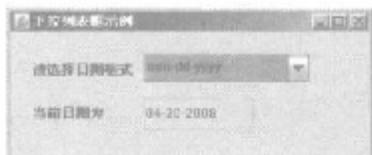


图 16-16 创建下拉列表框

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-16 所示。

【代码解析】在本程序的第 52 行到第 63 行是对日期显示的形式进行返回的方法，使用该方法才能得到运行结果中的日期显示形式。在本程序的 16 行创建了一个下拉列表框，该列表框的选项是由一个数组定义的，在结果中也可以看出下拉列表中具有这两个选项。在该程序中还为下拉列表注册了事件监听器，当进行选项选择时就会在文本框中以选择的形式来显示当前时间。



16.7 菜单

随着 GUI 开发的普及，菜单在开发中也变得越来越重要，几乎每个应用程序都会提供相应的菜单。因此，Swing 为菜单的开发提供了良好的支持，通过 Swing 中提供的菜单系列控件，开发人员可以非常方便地开发出各种各样的菜单，本节将对 Swing 中菜单的开发进行详细的介绍。

16.7.1 菜单介绍

菜单（JMenu）是标题栏下面的一行文字部分。菜单是应用程序中最常用的组件。菜单的组织方式为一个菜单条 JMenuBar 包含多个菜单项（ JMenuItem）。 JMenuItem 有两个子类，分别为 JRadioButtonMenuItem 及 JCheckBoxMenuItem 用于表示单选菜单项和复选菜单项。当用户选择某个菜单项后，就会触发一个 ActionEvent 时间，由 ActionListener 监听器处理。

菜单项有两种状态：启用状态和禁用状态，菜单项的状态可以使用 setEnabled 方法设置。创建完整的菜单一般需要以下几步。

- 创建菜单栏；
- 创建菜单及子菜单；
- 创建菜单项；
- 将菜单项加入到了菜单或菜单中，将子菜单加入到菜单中，将菜单加入到菜单栏中。

提示：可以为菜单项设置快捷键，使用方法 setMnemonic 可以设置菜单项的快捷键；也可以通过 setAccelerator 方法设置快捷键。

代码如下：

```
//设置菜单的快捷键  
fileMenu.setMnemonic('F');  
//设置“Exit”菜单项的快捷键为“T”  
JMenuItem exitItem = new JMenuItem("Exit", 'T');  
//设置“Exit”菜单项的加速器为“Ctrl+T”  
exitItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_T, InputEvent.CTRL_MASK));
```

16.7.2 创建菜单

上一节对菜单进行了简单的介绍，在本节中就来先创建一个简单的菜单，然后再对菜单进行更详细的讲解。

【范例 16-13】示例代码 16-13 是一个创建菜单的程序。

示例代码 16-13

```

01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import javax.swing.event.*;
05 //定义该类继承自 JFrame
06 public class KongJian13 extends JFrame
07 {
08     public KongJian13()
09     {
10         this.setTitle("学生管理系统");
11         JMenuBar menubar1=new JMenuBar();
12         this.setJMenuBar(menubar1);
13         JMenu menu1=new JMenu("文件");
14         JMenu menu2=new JMenu("编辑");
15         JMenu menu3=new JMenu("视图");
16         menubar1.add(menu1);
17         menubar1.add(menu2);
18         menubar1.add(menu3);
19         JMenuItem item1=new JMenuItem("打开");
20         JMenuItem item2=new JMenuItem("保存");
21         JMenuItem item3=new JMenuItem("打印");
22         JMenuItem item4=new JMenuItem("退出");
23         menu1.add(item1);
24         menu1.add(item2);
25         menu1.addSeparator();
26         menu1.add(item3);
27         menu1.addSeparator();
28         menu1.add(item4);
29         this.show();
30         this.setBounds(100,100,500,100);
31         this.setVisible(true);
32     }
33     public static void main(String[] args)
34     {
35         KongJian13 kj=new KongJian13();
36     }
37 }
38

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 16-17 所示。

【代码解析】在本程序中，首先创建了一个菜单栏容器，然后将该菜单栏添加到窗体中。在第 13 行到第 15 行创建了三个菜单，然后将这三个菜单添加到菜单栏中。接下来又创建了 4 个菜单项，并将这 4 个菜单项添加到第一个“打开”菜单下。该程序很简单，程序中也没有涉及事件。读者学习这个程序主要是了解菜单的基本结构。

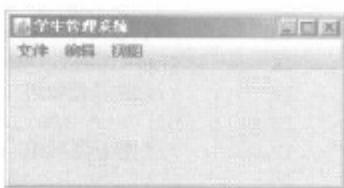


图 16-17 创建菜单



16.7.3 创建弹出式菜单

弹出式菜单有时也称为右键菜单，其一般在用户按下鼠标右键时在鼠标位置弹出，能够给用户的操作提供更大的方便。从某种程度上来说，右键菜单设计的好坏直接影响应用程序的易用性，本节将对 Swing 中弹出式菜单的开发进行详细的介绍。



注意：JPopupMenu 类实现弹出菜单。JPopupMenu 类不是继承 JMenu 类的而是从 JComponent 类继承过来。弹出式菜单的创建和菜单的创建基本相同，也需要新建一个弹出式菜单后再加入菜单项。

通过调用 JPopupMenu 类提供的 show 方法可以将弹出式菜单显示在指定控件的指定位置，下面的代码片段说明了如何显示弹出式菜单。

```
//测试鼠标事件是否应该触发弹出式菜单  
if(jpm.isPopupTrigger(e))  
{//显示弹出式菜单  
    jpm.show(this,e.getX(),e.getY());  
}
```

从上述代码片段中可以看出，由于弹出式菜单一般是由鼠标触发的，因此，在调用 show 方法显示弹出式菜单之前，一般都要调用 isPopupTrigger 方法对情况进行判断。



16.8 综合练习

1. 使用本节所学的控件编写一个用户注册程序。

【提示】可以先搭建一个最简单的界面程序，在向里面添加其他控件。例如下面给出的让用户选择性别和爱好的程序。

```
01 import java.awt.*;  
02 import java.awt.event.*;  
03 import javax.swing.*;  
04 //定义该类继承自 JFrame  
05 public class LianXil extends JFrame implements ActionListener  
06 {  
07     //创建 JPanel 对象  
08     private JPanel jp=new JPanel();  
09     //创建复选框数组  
10     private JCheckBox[] jcbArray=  
11         {new JCheckBox("上网"),new JCheckBox("运动")};  
12     //创建单选按钮数组  
13     private JRadioButton[] jrbArray=  
14         {new JRadioButton("男"),new JRadioButton("女",true)};  
15     //创建按钮数组  
16     private JButton[] jbarray={new JButton("提交"),new JButton("清空")};  
17     //创建标签数组  
18     private JLabel[] jlArray=  
19         {new JLabel("性别"),new JLabel("爱好"),new JLabel("输入内容为："')};  
20     //创建文本框  
21     private JTextField jtf=new JTextField();  
22     //创建按钮组
```



```

23     private ButtonGroup bg=new ButtonGroup();
24     public LianXil()
25     {
26         //设置 JPanel 布局管理器
27         jp.setLayout(null);
28         //对各个控件进行设置
29         for(int i=0;i<2;i++)
30         {
31             //设置单选按钮与复选框的大小位置
32             jrbArray[i].setBounds(40+i*100,40,80,30);
33             jcbArray[i].setBounds(40+i*120,100,120,30);
34             //将单选按钮与复选框添加到 JPanel 中
35             jp.add(jrbArray[i]);
36             jp.add(jcbArray[i]);
37             //为单选按钮与复选框注册动作事件监听器
38             jrbArray[i].addActionListener(this);
39             jcbArray[i].addActionListener(this);
40             //将单选按钮添加到按钮组中
41             bg.add(jrbArray[i]);
42             if(i>1) continue;
43             //设置标签与普通按钮的大小位置
44             jlArray[i].setBounds(20,20+i*50,80,30);
45             jbArray[i].setBounds(400+i*120,200,80,26);
46             //将标签与普通按钮添加到 JPanel 中
47             jp.add(jlArray[i]);
48             jp.add(jbArray[i]);
49             //为普通按钮注册动作事件监听器
50             jbArray[i].addActionListener(this);
51         }
52         //设置调查结果标签的大小位置，并将其添加到 JPanel 中
53         jlArray[2].setBounds(20,150,120,30);
54         jp.add(jlArray[2]);
55         //设置调查结果文本框的大小位置，并将其添加到 JPanel 中
56         jtf.setBounds(120,150,500,26);
57         jp.add(jtf);
58         //设置显示调查结果的文本框为不可编辑状态
59         jtf.setEditable(false);
60         //将 JPanel 添加进窗体
61         this.add(jp);
62         //设置窗体的标题、大小位置及可见性等
63         this.setTitle("注册页面");
64         this.setBounds(100,100,700,280);
65         this.setVisible(true);
66         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
67     }
68     public void actionPerformed(ActionEvent e)
69     {
70         if(e.getSource()==jbArray[1])
71             {//清空按钮执行的动作
72                 bg.clearSelection();
73                 for(int i=0;i<jcbArray.length;i++)
74                     jcbArray[i].setSelected(false);
75                 jtf.setText("");
76             }
77             else
78             {//其他按钮执行的动作
79                 //创建两个临时字符串
80                 StringBuffer templ=new StringBuffer("你是一个");
81                 templ.append(jcbArray[e.getSource().getIndex()].getText());
82                 jtf.setText(templ.toString());
83             }
84     }
85 }

```



```
StringBuffer temp2=new StringBuffer();
for(int i=0;i<2;i++)
{
    //获取年龄段的选中值
    if(jrbArray[i].isSelected())
    {
        temp1.append(jrbArray[i].getText());
    }
    //获取爱好的选中值
    if(jcbArray[i].isSelected())
    {
        temp2.append(jcbArray[i].getText()+"，");
    }
}
//打印结果
if(temp2.length()==0)
{//如果没有选取爱好
    jtf.setText("兴趣爱好选项不能为空!!!");
}
else
{//选取了爱好
    temp1.append("生，你比较喜欢");
    temp1.append(temp2.substring(0,temp2.length()-1));
    jtf.setText(Temp1.append(".").toString());
}
}
58
59     public static void main(String[] args)
60     {
61         LianXii x=new LianXii();
62     }
63 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序, 然后使用 Java 运行编译产生的 class 程序, 运行结果如图 16-18 所示。

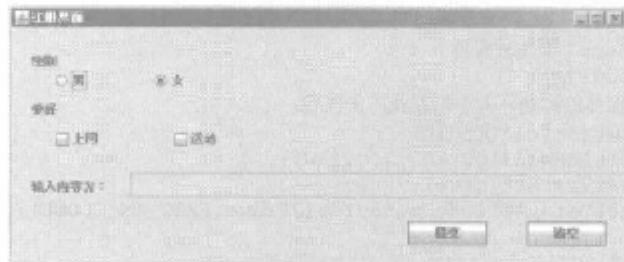


图 16-18 注册页面程序



16.9 小结

本章对 Swing 中的常用控件进行了讲解, 包括文本框、复选框、单选按钮、选项卡、分隔窗格、滑块、滚动条、列表框和菜单。Swing 中的控件除了这些外, 还有很多控件, 但是这些控件都是不常用的, 如果想对这些控件进行了解。可以参考电子工业出版社出版的《Java 优化编程 (第 2 版)》一书来进行更详细的学习。



16.10 习题

一、填空题

1. 文本框是通过_____类来创建的，在创建的文本框中当文本超出文本框规定长度时，将自动滚动文本显示。
2. 单选按钮是通过使用_____类来创建的。
3. _____类是 Swing 包中提供的用于实现滑块的控件，通过 JSlider 控件可以让用户在限定的范围内方便地选择需要的值。
4. _____是 Swing 中提供的用来实现进度条的控件，使用其可以非常方便地完成进度条的开发。
5. JComboBox 控件实际上组合了一个_____与一个_____，在默认情况下 JComboBox 控件提供的文本框是不可编辑的。
6. 文本框也是会触发事件的，它和按钮一样，都是触发_____事件。按钮是被单击时触发事件，而文本框是当_____触发事件。
7. 使用_____方法可以将指定的单选按钮添加到单选按钮组中。使用_____方法可以将指定的单选按钮从单选按钮组中删除。
8. ItemEvent 事件的发生条件是当_____时才会触发。
9. JSplitPane 控件允许在单个空间中放置两个控件，开发人员可以自由决定按_____方向或_____方向划分空间，还可以在程序运行期间使用鼠标自由调整空间的分割比例。

二、选择题

1. 选择获取组中单选按钮个数的方法（ ）。
A. add 方法 B. remove 方法 C. getButtonCount 方法
2. 选择设置用户注册性别的控件（ ）。
A. 单选按钮 B. 复选框
C. 下拉列表 D. 选项卡
3. 选择定义用户爱好的控件（ ）。
A. 单选按钮 B. 复选框
C. 下拉列表 D. 选项卡
4. 当滑块被拖动时，滑块的值将发生变化，从而触发的事件是（ ）。
A. ChangeEvent B. ActionEvent
C. ActionListener D. JPopupMenu
5. 设置进度条当前值的方法是（ ）。
A. getOrientation() B. isStringPainted()
C. setValue(int n) D. isIndeterminate()
6. 定义弹出式菜单的类为（ ）。
A. JpopupMenu B. JMenuItem
C. JradioMenuItem D. JCheckBoxMenuItem



三、简答题

1. 简述文本框和密码框的异同。
2. 简述创建菜单、菜单栏、菜单项的不同。

四、编程题

1. 编写一个用户注册的界面程序，在该界面中要具有用户名、密码、性别、用户爱好等内容。
2. 编写一个简易 Word 界面程序。

第 17 章 JDBC 数据库编程

在超市中购买东西时，在很多的商品中是不容易找到自己想要的商品的，这时候通常就需要找超市管理人员来帮忙解决。这就好像 Java 中的数据库编程 JDBC。在 Java 程序中，如果希望对很多的数据进行操作时，通常需要使用数据库编程。本章将学习如何进行数据库编程。通过本章的学习，读者应该实现如下几个目标。

- 对数据库有基本了解。
- 熟练掌握 JDBC 的编程步骤。
- 掌握如何在 Java 中进行数据库操作。



17.1 数据库基本介绍

数据库在应用程序中占有相当重要的地位，几乎所有的系统都必须要使用数据库。数据库发展到现在已经相当成熟了，已由原来的 Sybase 数据库，发展到现在的 SQL(Structured Query Language)、Oracal 等高级数据库。

17.1.1 数据库介绍

首先介绍一下什么是数据库。数据库的基本结构分三个层次，反映了观察数据库的三种不同角度。物理数据层是数据库的最内层，是物理存储设备上实际存储的数据的集合。这些数据是原始数据，同时也是加工的对象，由内部模式描述的指令操作处理的位串、字符和字组成。

概念数据层是数据库的中间一层，是数据库的整体逻辑表示，指出了每个数据的逻辑定义及数据间的逻辑联系，是保存记录的集合。它所涉及的是数据库所有对象的逻辑关系，不是它们的物理情况，而是数据库管理员概念下的数据库。

逻辑数据层是用户所看到和使用的数据库，表示了一个或一些特定用户使用的数据集合，即逻辑记录的集合。

17.1.2 数据库应用架构

数据库应用架构包括两种不同形式的数据库应用程序架构模型，主要包括 C/S 两层结构的与三层（或多层）结构的两种。

两层结构数据库应用架构模型的特点是所有的用户输入、验证及数据访问的功能都位于客户端中，一般来说客户端只适用于某一种特定的数据库。客户端与数据库服务器二者之间一般使用专用的协议进行连接，也有的情况是使用通用的数据库连接，如 JDBC、ODBC 等。

但是使用两层结构数据库也是存在很大缺点的。客户端与数据库服务器之间直接耦合，依赖度很高，无论哪边发生变化，都会直接影响到另一边。任何一种数据库服务器能



够支持的连接数都是很有限的，如果客户端很多，而又让每个客户端独自占用一个数据库连接不利于提高数据库的利用效率，也有可能造成其他用户不能正常使用数据库。

提示：在现在的开发中，已经很少使用两层结构的数据库应用模型了，而都是使用更加优越的三层结构数据库应用模型。三层结构的数据库应用模型的特点主要是，客户端与数据库之间不直接耦合，而是通过中间层应用服务器进行耦合，当客户端或数据库需要发生变化时可以通过中间层隔离变化，减小影响。

一般情况下在三层结构中，客户端软件都由通用的浏览器来担任，这样在对应用进行部署时就省去了为每台机器安装专用客户端的麻烦。同时，当开发了新的应用后，客户端机器也不需要做任何改变，打开浏览器浏览的自然就是新的功能了。根据需要，中间层的应用服务器可以同时连接几个同构或异构的数据库服务器，而这些在客户端的使用者是感觉不到也不用关心的。每个客户端不必独占一个数据库连接，可以大大提高数据库连接与数据库的利用效率。

17.1.3 数据库模型

数据库又可以从基于不同的模型来分类，可以分为层次型数据库、网状型数据库、关系型数据库、面向对象型数据库。层次型数据库是一组通过链接而互相联系在一起的记录。树结构图是层次型数据库的模式。层次模型的特点是记录之间的联系是通过指针实现，表示的是对象的联系。其缺点是无法反映多对象的联系，并且由于层次顺序的严格和复杂，导致数据的查询和更新操作复杂，因此应用程序的编写也比较复杂。

网状数据库是基于网络模型建立的数据库。网络模型，是使用网格结构表示实体类型、实体间联系的数据模型。网状模型的特点是记录之间的联系通过指针实现，多对多的联系容易实现。缺点是编写应用程序比较复杂，程序员必须熟悉数据库的逻辑结构。

关系数据库是基于关系模型建立的数据库。关系模型由一系列表格组成，用表格来表达数据集，用外键（关系）来表达数据集之间的联系。现在应用最常见的就是关系数据库，在下一节也主要来介绍一下关系数据库。



提示：关系数据库是使用最广泛的数据库。

对象型数据库是建立在面向对象模型基础之上。面向对象模型中最基本的概念是对象和类。对象是现实世界中实体的模型化，共享同一属性集和方法集的所有对象构成一个类。类可以有嵌套结构。系统中的所有类组成一个有根、有向无环图，称为类层次。



17.2 JDBC 数据库编程介绍

JDBC 就是 Java DataBase Connectivity，即 Java 数据库连接。JDBC 主要完成的几个任务分别是，与数据库建立一个连接；向数据库发送 SQL 语句；处理数据库返回的结果。实用 Java 程序语言和 JDBC 工具包开发程序，是独立于平台和厂商的。JDBC 就是将 Java 程序语言编写出来的程序，与数据库相连接。接下来，将详细讲述如何利用 JDBC 为程序连接数据库。



17.2.1 JDBC 和 ODBC 的关系

在 JDBC 数据库编程中经常要使用 ODBC。所以，在讲述 JDBC 的驱动程序分类之前，首先介绍什么是 ODBC。ODBC 是指 Open DataBase Connectivity，即开放数据库互连，它建立了一组规范，并且提供了一组对数据库访问的标准 API（应用程序编程接口），这些 API 利用 SQL 来完成其大部分任务。ODBC 也提供了对 SQL 的支持。

JDBC 驱动程序由实施了这些接口的类组成，JDBC 的总体结构有 4 个组件，分别为应用程序、驱动程序管理器、驱动程序和数据源。将 JDBC 转换成 ODBC 驱动器，依靠 ODBC 驱动器和数据库通信。在这种方式下，ODBC 驱动程序和桥代码必须出现在用户的每台机器中，这种类型的驱动程序最适合于企业网（这种网络上客户机的安装不是主要问题），或者是用 Java 编写的三层结构的应用程序服务器代码。

本地 API 一部分用 Java 来编写的驱动程序。这种类型的驱动程序把客户机 API 上的 JDBC 调用转换为 Oracle、Sybase、Informix、DB2 或其他 DBMS 的调用。像其他驱动程序一样，这种类型的驱动程序，要求将某些二进制代码加载到每台客户机上。

JDBC 网络纯 Java 驱动程序将 JDBC 转换为与 DBMS 无关的网络协议，这种协议又被某个服务器转换为一种 DBMS 协议。这种网络服务器中间件能够将它的纯 Java 客户机连接到多种不同的数据库上，所用的具体协议取决于提供者。通常，这是最为灵活的 JDBC 驱动程序。所有这种解决方案的提供者，都提供适合于 Intranet 用的产品。为了使这些产品支持 Internet，它们必须处理 Web 所提出的安全性、通过防火墙的访问等额外要求，几家提供者正将 JDBC 驱动程序，加到他们现有的数据库中间件产品中。

本地协议纯 Java 驱动程序类型的驱动程序将 JDBC 调用直接转换为 DBMS 所使用的网络协议，这将允许从客户机机器上直接调用 DBMS 服务器，是 Intranet 访问的一个很实用的解决方法。由于许多这样的协议都是专用的，因此数据库提供者自己将是主要来源。

17.2.2 为什么使用 JDBC 数据库编程

目前市面上有很多种数据库，例如 Oracle、Sybase、MS SQL Server 和 MS Access 等数据库。有些读者就会认为既然有这么多数据库，这里要学习数据库编程，是不是就要学习对应每一种数据库的编程方法呢。在 JDBC 之前是这样的，但是有了 JDBC 后，就变得非常容易。

JDBC 在数据库编程中将起到非常重要的作用。首先程序员可以使用 Java 开发基于数据库的应用程序，在遵守 Java 语言规则的同时，可以使用标准的 SQL 语句访问任何数据库。如果数据库厂商提供较低层的驱动程序，程序员可以在自己的软件中，使用比较优化的驱动程序。

很多数据库系统带有 JDBC 驱动程序，Java 程序就通过 JDBC 驱动程序与数据库相连，执行查询、提取数据等操作。Sun 公司还开发了 JDBC-ODBC bridge，用此技术，Java 程序就可以访问带有 ODBC 驱动程序的数据库。目前，大多数数据库系统都带有 ODBC 驱动程序，所以，Java 程序能访问诸如 Oracle、Sybase、MS SQL Server 和 MS Access 等数据库。



17.3 SQL 数据库操作技术

在 Java 中进行数据库操作，除了需要 JDBC 编程外，还需要一个数据库的技术，那就是 SQL 技术。SQL 技术是专门的，直接的对数据库操作的技术。

17.3.1 什么是 SQL

SQL 是 Structured Query Language 的缩写，Structured Query Language 翻译过来叫做结构化查询语言。SQL 是一种专门用来与数据库通信的语言。与其他语言（如 Java、Visual Basic 这样的程序设计语言）不一样，SQL 由很少的词构成，这是有意而为的。SQL 能够很好地完成一项任务——提供一种从数据库中读写数据的简单有效的方法。

SQL 是存在很多优点的，从整体的角度来说 SQL 有如下的优点。首先 SQL 不是某个特定数据库供应商专有的语言，几乎所有重要的 DBMS 都支持 SQL，所以，此语言几乎能与所有数据库打交道。然后就是 SQL 简单易学。它的语句全都是由具有很强描述性的英语单词组成，而且这些单词的数目不多，这个就是它简单易学的主要原因。最后 SQL 看上去尽管很简单，但实际上是一种强有力的语言，灵活使用其语言元素，可以进行非常复杂和高级的数据库操作。



注意：SQL 语句全都是由具有很强描述性的英语单词组成，而且这些单词的数目不多。

17.3.2 如何进行 SQL 操作

使用 SQL 能够完成数据库的创建、添加、删除、修改、查询等操作。本节就来简单地学习一下如何进行 SQL 操作。

查询操作是数据库操作中最常见的操作。在 SQL 中，使用 Select 语句可在需要的表单中检索数据，在进行检索之前，必须知道需要的数据存储在哪里，Select 语句可由多个查询子句组成。查询操作的基本结构如下所示。

```
Select [all|distinct] [into new_table_name]
From (表名|视图名)
[where 搜索条件]
Group by 把查到的按什么标准分组
(having 搜索条件)
(order by 按什么顺序排序) [升序|降序]
```

在查询操作的基本结构中，all 是指明查询结果中可以显示值相同的列，同时 all 是系统默认值。distinct 是指明查询结果中如有值相同的列，只显示其中的一列，对 distinct 来说，NULL 被认为相同的值。into 子句用于把查询结果存放到一个新建表中。



注意：select....into 命令不能与 compute 子句一起使用。新表是由 select 子句指定的列构成。

From 子句指定需要进行数据查询的表。where 子句指定数据检索的条件，以限制返回

的数据。Group by 子句指定查询结果的分组条件。having 子句指定分组搜索条件，它通常与 Group by 子句一起使用。order by 子句指定查询结果的排序方式。

除了查询操作外，还有其他操作。数据插入语句如下：

数据插入：insert into <表名> (列名) value (对应列的值)

数据修改语句如下：

数据修改：update <表名> set <列名>=<表达式> (where<条件>)

数据删除语句如下：

数据删除：delete (from) <表名|视图名> (where 子句)

17.4 创建数据库

在前面的学习中，读者已经对数据库编程的基本知识有了一些基本介绍。在学习在 Java 中进行数据库操作之前，首先来学习一下如何创建数据库。这里以 Access 数据库和 SQL Server 数据库为例。

17.4.1 创建 Access 数据库

学习如何使用 JDBC 进行数据库开发之前，首先需要建立一个数据库。本节使用的是非常简单易用的 Access 创建的数据库。例如创建一个记录学生信息的数据库，如表 17-1 所示。

表 17-1 学生信息表

表 名	字 段 名	字段类型	字段含义	备 注
student	sid	整数数值	学生编号	sid 字段为主键
	sname	字符串文本	学生姓名	
	sage	整数数值	学生年龄	
	stel	字符串文本	联系电话	

同时，为了方便后面示例程序的运行，在表中插入了一些示例数据，如图 17-1 所示。

学号	姓名	年龄	电话
2007001	张三	20	1234567
2007002	李四	22	7654321
2007003	王五	21	4563271

图 17-1 示例数据

本节只是介绍了本章所使用 Access 数据库的表结构与数据信息，因为 Access 数据库非常容易建立，这里就不再详细地讲解如何创建 Access 数据库。

17.4.2 创建 SQL Server 数据库

有些读者可能并不喜欢使用 Access 数据库的，所以下面为这部分读者讲解如何创建标准的数据库，这里以 SQL Server 数据库为例。首先讲解如何安装 SQL Server 数据库，然后学习如何创建 SQL Server 数据库。

首先介绍如何安装 SQL Server 2005，具体步骤如下所示。



- ① 把 SQL Server 2005 安装光盘放入光驱，弹出自动选择菜单。在此菜单中可以选择自己需要安装的选项，这里通常选择安装中的第一项，首先安装组件等内容。
- ② 因为需要安装 SQL，所以选择“安装服务器组件、工具、联机丛书和示例”，使用它可以安装 SQL Server 2005 软件。其他的一些帮助文档、说明文档之类的安装，这些可以在需要时进行安装，选择后弹出如图 17-2 所示的界面。
- ③ 选择“我接受许可条款和条件”选项。
- ④ 单击“安装”按钮，系统便开始安装软件了，此时安装软件可能从最外围基础软件开始。
- ⑤ 一段时间后，会出现如图 17-3 所示的界面。

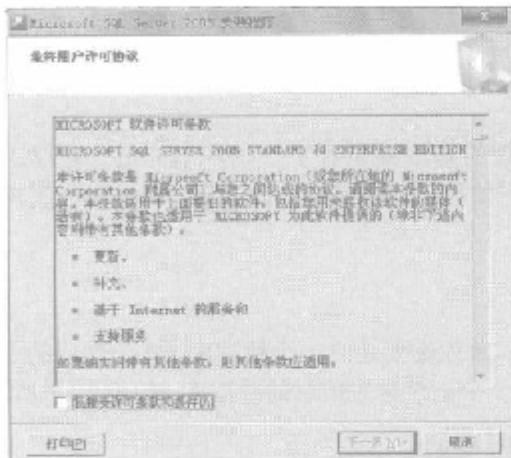


图 17-2 最终协议许可协议界面



图 17-3 安装必要组件完成界面

- ⑥ 安装完这些组件后，系统将会进入到 SQL Server 程序的安装。前面所安装的都是一些安装 SQL Server 数据库所必需的组件。
- ⑦ 系统扫描到计算机配置完成后，便会正式安装 SQL Server 软件，弹出安装界面。
- ⑧ 单击“下一步”按钮，弹出界面如图 17-4 所示。



图 17-4 系统检查配置界面



⑨ 在图 17-4 中，安装程序将把前面扫描过的计算机硬件和软件信息，统一显示在显示框中，让用户能够很清晰自己系统中哪些是不符合安装要求的，直接单击“下一步”按钮，就会进行安装。

⑩ 系统计算硬盘的剩余空间后，看是否能安装 SQL Server 软件，直接单击“下一步”按钮，弹出输入注册码的界面。

⑪ 输入姓名、公司名称及注册码的信息，单击“下一步”按钮，弹出界面如图 17-5 所示。

⑫ 在图 17-5 中，安装程序将让用户选择安装哪些额外组件，默认是不选择的，直接单击“下一步”按钮，会出现界面如图 17-6 所示。

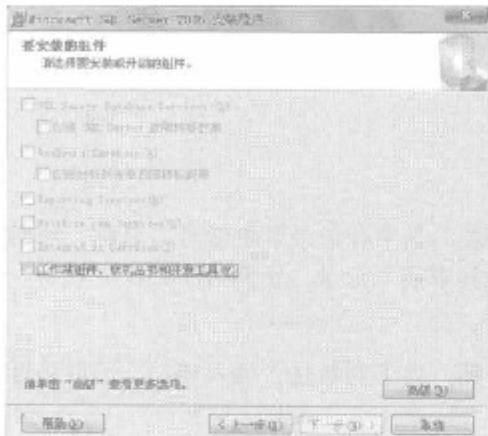


图 17-5 选择安装组件界面

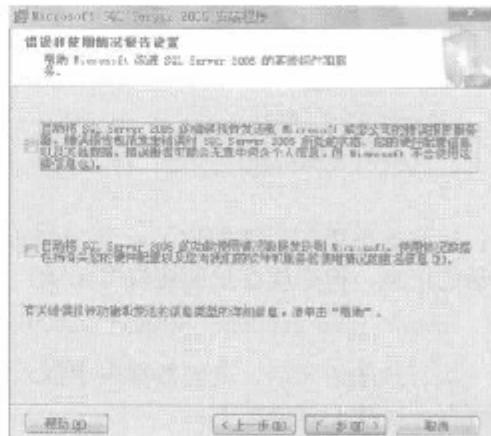


图 17-6 错误和使用情况报告设置界面

⑬ 在图 17-6 中，系统将会把一些错误和使用情况设置信息显示给用户，让用户知道自己目前安装软件的情况，单击“下一步”按钮，进入准备安装界面。

⑭ 当选择了要安装的客户组件后，在图 17-6 中的文本区中会显示出，需要安装的额外组件，单击“安装”按钮，进行安装。

⑮ 此时，要经过比较长时间的等待后，直到所有的组件全部安装完成后，弹出如图 17-7 所示的界面。

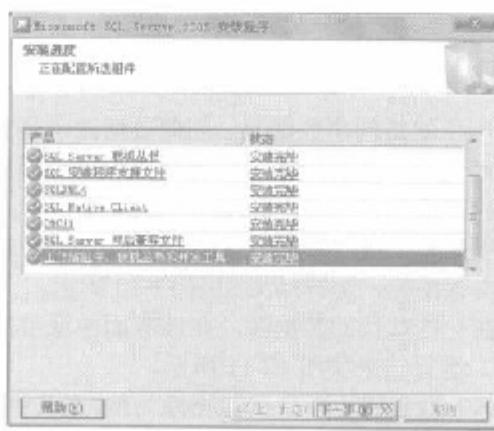


图 17-7 安装过程中的界面



⑯ 单击“下一步”按钮，会进入安装完成界面。

这样，SQL Server 2005 数据库安装完毕，接下来就是学习如何在 SQL Server 中创建数据库。在 SQL Server 数据库中，数据库是通过 SQL 语言来创建的。创建一个标准数据库的顺序是先创建一个数据库，在数据库中创建数据表，然后向数据表中插入数据。这样就创建了一个可以在 Java 程序中使用的数据库。

首先创建一个自己的数据库，它是通过 SQL 语句中的 create 语句创建的。示例如下：

```
create database myDatabase;
```

然后在该数据库中创建一个数据表，这里还是创建一个学生表，来记录学生信息。示例如下：

```
create table student
(
    sid char(10),
    sname char(20),
    sage char(5),
    stel char(10)
);
```

执行该条语句后就创建了一个 student 数据表，其中有记录学生学号、名称、年龄和电话的信息。但是现在里面还没有记录，需要向其中插入记录。示例如下：

```
insert into student values('200801', 'Tom', '23', '123456789');
```

执行该条语句后，就向数据表中插入一条语句，为了判断是否插入成功，可以进行查询数据库操作。示例如下：

```
select * from student;
```

执行该条语句后就可以看到插入的话语。



17.5 JDBC 编程步骤

前面几节介绍的内容，都是进行 JDBC 数据库编程的基础。在本节就来对如何进行 JDBC 编程进行详细的讲解。JDBC 编程有严格的步骤，本节将一步一步地学习如何进行 JDBC 数据库编程。

17.5.1 创建数据源

进行 JDBC 编程的第一步就是创建数据源。创建数据源也是有严格步骤的，读者可以按照如下步骤进行学习，经过两三次的操作，读者就应该能够很熟练地创建数据源。

- ① 找到控制面板中的管理工具一项，如图 17-8 所示。
- ② 打开管理工具，里面有一项“数据源（ODBC）”，如图 17-9 所示。
- ③ 选择数据源，双击该图标，运行结果如图 17-10 所示。
- ④ 在图 17-10 中存在一些默认的数据源，在该界面中就可以添加自己的数据源。在界面中单击“添加”按钮，运行结果如图 17-11 所示。
- ⑤ 这里以在 SQL Server 数据库中创建的数据库为例，在其中选择 SQL Server 数据库，然后单击“完成”按钮，运行结果如图 17-12 所示。



图 17-8 管理工具



图 17-9 数据源

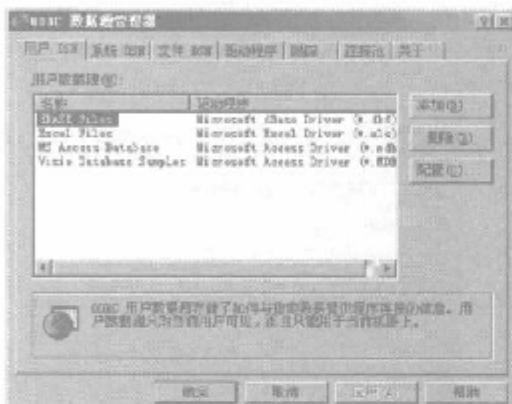


图 17-10 数据源管理器



图 17-11 添加

⑥ 在该界面中，可以为新创建的数据源进行命名，读者可以随便为该数据源起一个名字。其中第二项为对该数据源的描述，这里是可以不填的。第三项是选择连接的是哪一个SQL Server 数据库，这里选择本地数据库。填写完成后，单击“下一步”按钮，运行结果如图 17-13 所示。

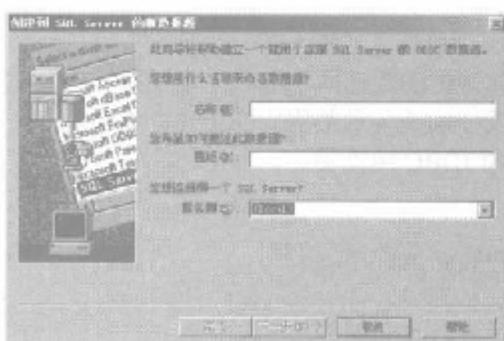


图 17-12 数据源命名

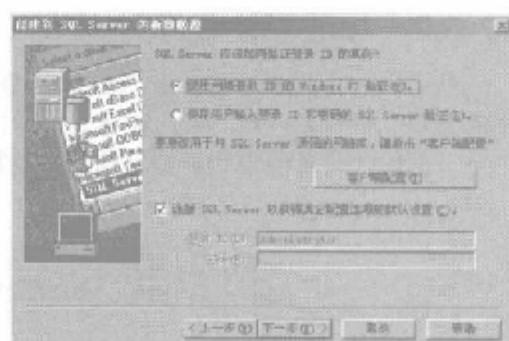


图 17-13 验证

- ⑦ 在出现的界面中使用默认值，单击“下一步”按钮，运行结果如图 17-14 所示。
 ⑧ 在图 17-14 的界面中选择默认的数据库，同样也是单击“下一步”按钮，运行结果如图 17-15 所示。

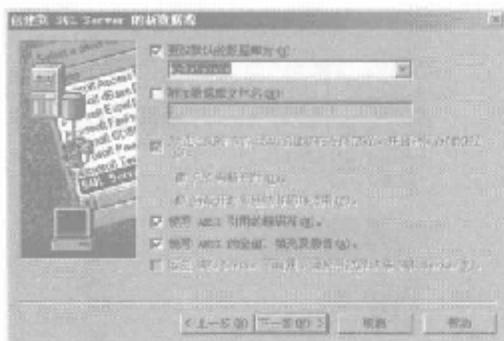


图 17-14 选择数据库

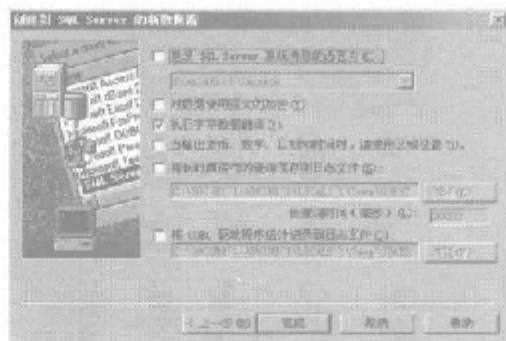


图 17-15 选择数据库后

⑨ 在该界面中同样采用默认的选择，单击“完成”按钮，运行结果如图 17-16 所示。

⑩ 创建数据源的步骤到如图 17-15 所示界面已经创建完成，创建完成后可以对是否创建完成进行测试的，只需要单击“测试数据源”按钮，运行结果如图 17-17 所示。



图 17-16 完成

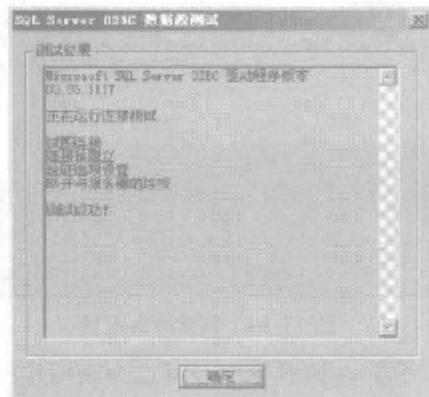


图 17-17 测试数据源结果

执行到这一步后就表示，数据源已经创建完成，并且数据源创建成功。

17.5.2 加载驱动程序

从前面的介绍中可以看出，在使用 JDBC 连接特定的数据库之前首先要加载相应数据库的 JDBC 驱动类，本节将向读者介绍如何在开发中加载各种数据库的 JDBC 驱动类。

下面的代码片段说明了如何加载特定数据库的 JDBC 驱动类。

```
1     try
2     {
3         //通过 Class 类的 forName 方法加载指定的 JDBC 驱动类
4         Class.forName("驱动类全称类名");
5     }
6     catch(java.lang.ClassNotFoundException e)
7     {
8         e.printStackTrace();
9     }
```

使用 Class 类的 forName 方法加载指定的 JDBC 驱动类时要给出驱动类的全称类名。由于 Class 类的 forName 方法可能抛出捕获异常 java.lang.ClassNotFoundException，因此在

调用此方法时必须进行异常处理。



注意：JavaSE 安装以后仅自带了 JDBC-ODBC 连接桥的驱动程序，在本书中也将使用 JDBC-ODBC 连接桥的形式来连接数据库。其他类型的数据库在加载驱动类之前要首先从 Internet 上下载或从数据库的安装目录中寻找驱动程序对应的 jar 包，并将 jar 包文件所在的路径添加到 classpath 环境变量中。

17.5.3 建立数据库连接

加载驱动程序后，就可以进行建立数据库连接。建立数据连接是通过调用 `java.sql.DriverManager` 类的 `getConnection` 方法来建立的，下面对该方法进行介绍。

```
public static Connection getConnection(String url, String user, String password)
throws SQLException
```

参数 `url` 为指定数据库的连接字符串，参数 `user` 为要连接数据库的用户名，参数 `password` 为用户名对应的密码。如果没有用户名与密码，可以用两个空字符串来代替。此方法有可能抛出捕获异常 `java.sql.SQLException`，因此在调用此方法时必须进行异常处理。

指定数据库的连接字符串由三部分组成，各部分之间用“`:`”分隔，如下所示。

`jdbc:<子协议>:<子名称>`

子协议指的是数据库的类型，例如可以是 `odbc`、`mysql` 或 `oracle` 等。子名称指的是数据源的名称或数据库的网络标识字符串。

例如，下面的代码片段以连接前面创建的 ODBC 数据源 `student` 为例，说明了如何获取与关闭数据库连接。

```
1 //声明连接引用
2 Connection con=null;
3 //创建数据库连接字符串
4 String url="jdbc:odbc:student";
5 try
6 {
7     //创建数据库连接
8     con=DriverManager.getConnection(url,"","");
9     //连接以后操作数据库的代码
10 }
11 catch(java.sql.SQLException e)
12 {
13     e.printStackTrace();
14 }
```

上述代码中第 8 行通过 `DriverManager` 类的 `getConnection` 方法获取了数据库连接，在该程序 `url` 中给出的子协议为 `odbc`，数据源为前面所建立的 `student` 数据源。

17.5.4 进行数据库操作

上一节中讲解了如何建立数据库连接，当成功地创建了数据库连接后，就可以使用连接对象提供的 `createStatement` 方法来进行数据库操作，这里的数据库操作是通过 SQL 语句来完成的，下面给出了该方法的原型。



```
public Statement createStatement() throws SQLException
```

创建了语句对象后就可以调用语句对象的 `executeUpdate` 方法来执行对数据库进行更新的语句了，下面给出了该方法的原型。

```
public int executeUpdate(String sql) throws SQLException
```

参数 `sql` 为要执行的 SQL 语句对应的文本字符串，如 “`insert into student values('200801','Tom','23','123456789');`”。该方法返回值表示成功地操作了多少条数据库记录。同样该方法也是可以发生异常的，所以也需要进行异常处理。

例如，下面的代码片段说明了如何创建语句对象与执行更新数据库的 SQL 语句。

```
1   try
2   {
3       //获取 statement 对象
4       Statement stat=con.createStatement(); //con 为一个指向 Connection 对象的
引用
5       //插入一条记录
6       stat.executeUpdate("insert into student values('200801','Tom',
7           '23','123456789');");
8       //关闭语句对象
9       stat.close();
10      }
11  catch(java.sql.SQLException e)
12  {
13      e.printStackTrace();
14  }
```

在该程序中首先是使用 `createStatement` 方法创建了一个 `Statement` 对象，然后使用创建后的 `Statement` 对象调用 `executeUpdate` 方法来对数据库进行操作。`executeUpdate` 方法的参数为一个 SQL 参数，将按照该 SQL 进行数据库操作。

17.5.5 获取数据库中信息

上一节中学习了如何进行数据库操作，并进行了如何向数据库中插入记录。有时候经常需要来判断是否插入成功，这时候就需要来获取数据库中的信息。获取数据库中的信息是通过调用 `Statement` 对象的 `executeQuery` 方法来执行查询数据库的 SQL 语句。该方法原型如下：

```
public ResultSet executeQuery(String sql) throws SQLException
```

参数 `sql` 为要执行的 SQL 查询语句，例如 “`select * from student`”，返回值为 `ResultSet` 类型的对象引用。`ResultSet` 类型的对象中封装了查询的结果，可以调用其 `next` 方法移动指向结果集中记录的游标，下面给出了该方法的原型。

```
public boolean next() throws SQLException
```

该方法是获取数据库信息的最重要的方法。同时，`ResultSet` 中还提供了很多获取当前记录指定字段值的 `getXXX` 方法，如表 17-2 所列。

在所有的 `getXXX` 方法中都有两个重载的方法，其中一个接收 `int` 类型的参数，参数值为要获取值的字段对应的列索引，例如 “`getString(1)`” 将返回第一列的字符串内容。另一个接收 `String` 类型的参数，参数值为要获取值的字段对应的列名称，例如 “`getString("sid")`” 将获取 `sid` 列的字符串内容，并且所有的 `getXXX` 方法都有可能抛出 `java.sql.SQLException`

捕获异常，因此在调用时要进行异常处理。

表 17-2 getXxx方法及其返回值类型

方法名	返回类型	方法名	返回类型
getBoolean	boolean	getByte	byte
getBytes	byte[]	getDate	java.sql.Date
getDouble	double	getFloat	float
getInt	int	getLong	long
getObject	Object	getShort	short
getString	String	getTime	java.sql.Time
getTimestamp	java.sql.Timestamp		



注意：所有的 getXxx 方法都有可能抛出 java.sql.SQLException 异常，因此在调用时要进行异常处理。

17.5.6 JDBC 数据库编程实例

在详细地学习 JDBC 编程后，下面将以一个综合进行 JDBC 数据库编程的实例来进一步讲解 JDBC 数据库编程。

【范例 17-1】示例代码 17-1 是一个 JDBC 数据库编程的程序。

示例代码 17-1

```

01 import java.sql.*;
02 public class ShuJuXu
03 {
04     public static void main(String[] args)
05     {
06         //声明Connection引用
07         Connection con=null;
08         try
09         {
10             //加载 JDBC-ODBC 桥驱动类
11             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
12             //创建数据库连接
13             con=DriverManager.getConnection("jdbc:odbc:student","","");
14             //创建 Statement 对象
15             Statement stat=con.createStatement();
16             //在数据库中插入一条记录
17             int count=stat.executeUpdate("insert into student values
('200801','Tom','23','123456789')");
18             System.out.println("成功插入了"+count+"条记录!!! ");
19             //对 student 表进行全表检索
20             ResultSet rs=stat.executeQuery("select * from student");
21             //打印表头信息
22             System.out.println("学号\t姓名\t\t年龄\t\t电话");
23             //循环打印结果集中的每一条记录
24             while(rs.next())
25             {
26                 //获取当前记录中第一列内容
27                 String sid=rs.getString(1);
28                 //获取当前记录中第二列内容

```



```
29         String sname=rs.getString(2);
30         //获取当前记录中第二列内容
31         String sage=rs.getString(3);
32         //获取当前记录中第四列内容
33         String stel=re.getString(4);
34         //打印本条记录的内容
35         System.out.println("id:"+id+" "+sname+" "+sage+" "+stel);
36     }
37     //关闭结果集合
38     rs.close();
39     //关闭语句
40     stat.close();
41 }
42 catch(Exception e)
43 {
44     e.printStackTrace();
45 }
46 finally
47 {
48     try
49     {
50         //关闭数据库连接
51         con.close();
52     }
53     catch(Exception e)
54     {
55         e.printStackTrace();
56     }
57 }
58 }
59 }
```

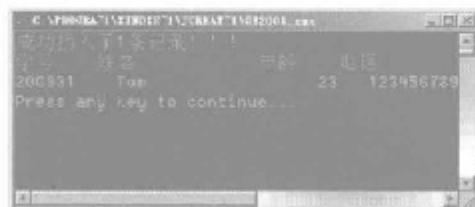


图 17-18 数据库操作

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 17-18 所示。

【代码解析】在执行该程序之前，要先按照前面的讲解创建一个数据库，并在该数据库中创建一个 student 数据表，还需要创建数据源来指定该数据库。在程序的第 11 行使用 JDBC-ODBC 的形式加载驱动程序。在第 13 行建立数据库连接，在第 17 行进行数据库操作，在第 20 行获取数据库中信息，并使用 next 方法循环得到每一条记录。



17.6 事务处理

对数据库进行并发操作时，为了避免由于并发操作带来的问题，一般要将同一个任务中对数据库的增、删、改、查操作编写到一个事务中，同一个事务中的所有操作要么全部执行成功，要么都不执行。因此 JDBC 也提供了对事务开发的支持，本节将向读者介绍 JDBC 中有关事务开发的知识。

17.6.1 事务介绍

事物是 SQL 中的单个逻辑工作单元，一个事务内的所有语句被作为整体执行，遇到错



误时，可以回滚事务，取消事务所做的所有改变，从而可以保证数据库的一致性和可恢复性。

一个事务逻辑工作单元必须具有以下4种属性，包括原子性、一致性、隔离性和永久性。原子性是指一个事务必须作为一个原子单位，它所做的数据修改操作要不全部执行，要不全部取消。一致性是指当事务完成后，数据必须保证处于一致性的状态。隔离是指一个事务所做的修改必须能够跟其他事务所作的修改分离开来，以免在并发处理时，发生数据错误。永久性是指事务完成后，它对数据库所做的修改应该被永久保持。

进行事务操作主要使用 Connection 对象中的三个方法。setAutoCommit 方法是指将此连接的自动提交模式设置为给定状态。如果参数 autoCommit 的值为 true，则每执行一句 SQL 命令将自动被作为单个事务提交；否则，其将所有 SQL 语句聚集到一个事务中，直到调用 commit 方法或 rollback 方法为止，其默认值为 true。



注意：如果参数 autoCommit 的值为 true，则每执行一句 SQL 命令将自动被作为单个事务提交；否则，其将所有 SQL 语句聚集到一个事务中，直到调用 commit 方法或 rollback 方法为止，其默认值为 true。

commit 方法是指提交当前的事务，并自动开始下一个事务。执行此方法后，会释放此 Connection 对象当前持有的所有数据库锁，同时要注意此方法只应该在自动提交模式被禁用的情况下使用。

rollback 方法是指回滚当前的事务，并自动开始下一个事务。执行此方法后，会释放此 Connection 对象当前持有的所有数据库锁，同时要注意此方法只应该在自动提交模式被禁用的情况下使用。

17.6.2 进行事务操作

对事务有了一个基本了解后，在本节中就通过程序的形式来了解如何进行事务操作。

【范例 17-2】示例代码 17-2 是一个进行事务操作的程序。

示例代码 17-2

```

01 import java.sql.*;
02 public class ShuJuKu2
03 {
04     public static void main(String[] args)
05     {
06         //声明 Connection 引用
07         Connection con=null;
08         //声明 Oracle 数据库的连接字符串、用户名及密码
09         String url="jdbc:odbc:student";
10         String user="";
11         String password="";
12         try
13         {
14             //加载 Oracle 驱动类
15             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
16             //创建数据库连接
17             con=DriverManager.getConnection(url,user,password);
18             //禁用自动提交模式，开始一个事务
19             con.setAutoCommit(false);
20             //创建 Statement 对象

```



```
21         Statement stat=con.createStatement();
22         System.out.println("查询原数据库内记录");
23         myVoid(stat);
24         //向表中插入一行记录
25         stat.executeUpdate("insert into student values('200801','Tom',
26                           '23','123456789')");
27         //再次向表中插入一行记录
28         stat.executeUpdate("insert into student values('200802',
29                           'Lucy','23','555555555')");
30         System.out.println("插入记录后数据库内记录");
31         myVoid(stat);
32         //将事务提交
33         con.commit();
34         //恢复自动提交模式
35         con.setAutoCommit(true);
36         //关闭语句
37         stat.close();
38     }
39     catch(Exception e)
40     {//如果出现错误将发起回滚操作
41         e.printStackTrace();
42         try
43         {
44             con.rollback();
45             System.out.println("出现异常，事务回滚");
46         }
47         catch(Exception ae)
48         {
49             e.printStackTrace();
50         }
51     }
52     finally
53     {
54         try
55         {
56             //关闭数据库连接
57             con.close();
58         }
59         catch(Exception e)
60         {
61             e.printStackTrace();
62         }
63     }
64     //自定义的检索music表全表的方法
65     public static void myVoid(Statement stat) throws SQLException
66     {
67         ResultSet rs=stat.executeQuery("select * from student");
68         //打印表头信息
69         System.out.println("学号\t姓名\t年龄\t电话");
70         //循环打印结果集中的每一条记录
71         while(rs.next())
72         {
73             //获取当前记录中第一列内容
74             String sid=rs.getString(1);
75             //获取当前记录中第二列内容
76             String sname=rs.getString(2);
77             //获取当前记录中第三列内容
```

```

78         String sage=rs.getString(3);
79         //获取当前记录中第四列内容
80         String stel=rs.getString(4);
81         //打印本条记录的内容
82         System.out.println(sid+" "+ename+" "+sage+" "+stel);
83     }
84     //关闭结果集
85     rs.close();
86 }
87 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序, 然后使用 Java 运行编译产生的 class 程序, 运行结果如图 17-19 所示。

再次运行程序, 运行结果如图 17-20 所示。

【代码解析】在本程序中使用了提交事务和回滚事务。在本程序中先获取数据库中本身的记录, 然后插入两条语句后, 再来获取记录, 程序的第 31 行使用了事务提交。从而实现插入效果。但是再次运行程序, 因为数据库中已经存在要插入的记录, 所以再次插入会发生异常, 从而在 catch 语句中进行回滚事务。

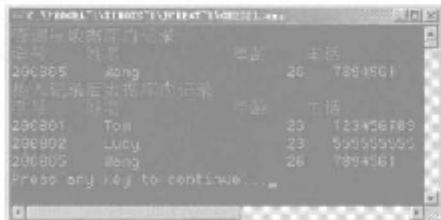


图 17-19 提交事务

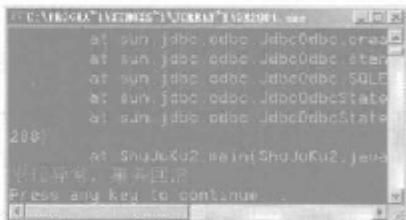


图 17-20 回滚事务

17.7 综合练习

简要说明 JDBC 编程步骤。

【提示】由于 JDBC 的编程步骤比较多, 读者是很难全部记下来的, 经常会由于没有做其中一步而造成程序发生错误, 所以这里再来简单的说一下 JDBC 的编程步骤。

① 创建数据库

因为很多数据库操作都是指定某一个数据库来进行操作的, 所以在进行数据库编程前要首先建立一个数据库。如何创建库是会因为数据库的不同而不同的。读者可以根据需求来相应地学习特定的一种数据库。

② 创建数据源

创建数据源是数据库操作中最容易丢的一步, 也是最难发现的一步。很多读者发现程序发生错误后, 都会首先去找程序中有什么错误, 从而很难发现没有创建数据源的错误。创建数据源中将为数据源起一个名字, 读者在进行该步操作时一定要起一个容易记住的名字, 因为在程序中还要用到该名称。

③ 编写程序

JDBC 数据库编程剩下的步骤都是在编写程序中的。在编写程序中也是不能丢掉必要步骤的。首先要加载驱动, 根据数据库的不同而加载不同的驱动。然后就是建立数据库连



接，接下来就是使用 SQL 语句进行数据库操作。最后就是获取信息。



17.8 小结

本章首先对数据库、JDBC 和 SQL 进行了简单的介绍，这些都是 Java 中进行数据库编程的基础。该书中对这些知识的讲解是很简单的，如果读者想了解更多这方面的知识可以参考电子工业出版社出版的《SQL SERVER 2005 T-SQL 数据库设计》一书来进行学习。



17.9 习题

一、填空题

1. 数据库应用架构包括两种不同形式的数据库应用程序架构模型，主要包括_____和_____。
2. 数据库又可以从基于不同的模型来分类，可以分为_____、_____、_____、_____。
3. JDBC 驱动程序由实施了这些接口的类组成，JDBC 的总体结构有 4 个组件：_____、_____、_____和_____。
4. 使用 SQL 能够完成数据库的_____、_____、_____、_____等操作。
5. 数据库的基本结构分三个层次，分别是_____、_____、_____。
6. 一个事务逻辑工作单元必须具有以下 4 种属性，包括_____、_____、_____和_____。

二、选择题

1. 回滚当前业务的方法是（ ）；
A. setAutoCommit 方法 B. commit 方法 C. rollback 方法
2. 加载驱动程序的方法是（ ）。
A. forName 方法 B. getConnection 方法
C. createStatement 方法 D. executeQuery 方法

三、简答题

1. 简述 SQL 语言的优点。
2. 简述 JDBC 数据库联系的编程步骤。
3. 简述连接数据库的不同。

四、编程题

1. 编写一个查询学生系统中，大一学生的数量。
2. 编写一个界面程序，在该界面程序中能够对指定的学生修改年龄。

第 18 章 Java 中输入/输出流

在学习输入/输出流之前，先来了解一下实际中的管道的问题。管道通常是可以双向流通的，Java 中的流也是这样的，有输出流和输入流。管道是可以输送石油，也是可以输送水的，Java 中的流也是可以输送不同的东西的，包括字节和字符等内容。Java 中的输入/输出流又称为 I/O 流。通过本章的学习，读者应该能够实现如下几个目标。

- 了解什么是 I/O 流。
- 掌握流的分类。
- 熟练掌握流如何进行文件操作。



18.1 I/O 流简介

学习如何使用 I/O 流进行输入/输出操作之前，首先应当了解 I/O 流的基本原理与分类，这样才能恰当地运用各种 I/O 流进行不同的输入/输出操作。

18.1.1 什么是 I/O 流

数据流是形象的概念，可以理解为是一种“数据的管道”。管道中流动的东西可以是基于字节，也可以是基于字符的等。就好像管道里面可以流动水，也可以流动石油一样，当程序需要读取数据的时候，就会开启一个通向数据源的管道，这个数据源可以是存放在硬盘中的文件，也可以是内存中的数据，或者是网络上的数据。

Java 中的数据流分为两种，一种是字节流，另一种是字符流。这两种流主要由 4 个抽象类来表示，分别为 InputStream，OutputStream，Reader，Writer，输入输出各两种。其中 InputStream 和 OutputStream 表示字节流，Reader 和 Writer 表示字符流，其他各种各样的流均是继承这 4 个抽象类而来的。

18.1.2 节点流与处理流

根据流功能层次的不同可以将其分为节点流（Node Streams）与处理流（Processing Streams）两类，下面列出了这两种流的异同。

节点流一般用于直接从指定的位置进行读/写操作，例如磁盘文件、内存区域、网络连接等，其中一般只提供了一些基本的读写操作方法，功能比较单一。

处理流往往是用于对其他输入/输出流进行封装，对内容进行过滤处理，其中一般提供了一些功能比较强大的读写方法。

实际应用中，通常是将节点流与处理流二者结合起来使用。节点流直接与指定的源或目标相连，例如某个文件、某个网络连接等。而处理流则对节点流或其他处理流进一步进行封装，提供更丰富的输入/输出操作能力，例如缓冲、按字符串行读写等。



提示：节点流直接与指定的源或目标相连，而处理流则对节点流或其他处理流进一步进行封装，提供更丰富的输入/输出操作能力。

18.1.3 字节流与字符流

根据流处理数据类型的不同也可以将其分为字节流与字符流两类，下面列出了这两种流的不同之处。

字节流以字节为基本单位来处理数据的输入/输出，一般都用于对二进制数据的读写，例如声音、图像等数据。

字符流以字符为基本单位来处理数据的输入和输出，一般都用于对文本类型数据的读写，例如文本文件、网络中发送的文本信息等。

表 18-1 列出了 Java I/O 中字节流与字符流的 4 个抽象基类，Java I/O 中的其他字节流与字符流都派生自这 4 个抽象基类。

表 18-1 字节流与字符流的抽象基类

I/O 流	字节流	字符流
输入流	InputStream	Reader
输出流	OutputStream	Writer

18.1.4 抽象基类

字节流和字符流的抽象基类包括 InputStream 类、OutputStream 类、Reader 类和 Writer 类，在后面将要讲到的流类都是继承这几个抽象基类的。

1. InputStream 类

InputStream 类是一个输入流，同样也是一个字节流。InputStream 类是表示字节输入流的所有类的超类。其中定义了一些基本的读取字节数据流的方法，由其子类继承并扩展。声明的方法列表如表 18-2 所示。

表 18-2 InputStream 中的方法

方法名	作用
public int available() throws IOException	获取可以从此输入流读取（或跳过）的字节数
public void close() throws IOException	关闭输入流，同时释放系统资源
public void mark(int readlimit)	在输入流中标记位置，参数 readlimit 为位置参数
public boolean markSupported()	用于测试该输入流是否支持 mark 和 reset 方法
public abstract int read() throws IOException	从输入流中读取下一个数据字节
public int read(byte[] b) throws IOException	从输入流中读取字节数据，并存入字节数组 b 中
public int read(byte[] b,int off,int len) throws IOException	从输入流中读取 len 个数据字节，并存入字节数组 b 中
public void reset() throws IOException	将此流重新定位到最后一次调用 mark 方法时所处的位置
public long skip(long n) throws IOException	跳过并且放弃输入流中的 n 个字节数据

使用 InputStream 类是可能发生 IOException 异常的，所以在使用时是要进行异常处理的。

2. OutputStream类

OutputStream类是一个输出流，同样也是一个字节流。OutputStream类是表示输出字节流的所有类的超类。输出流接收输出字节并将这些字节发送到某个接收器。其中声明的方法列表如表18-3所示。

表18-3 OutputStream中的方法

方法名	作用
public void close() throws IOException	关闭输出流，同时释放系统资源
public void flush() throws IOException	刷新输出流，同时输出所有缓冲的输出字节
public void write(byte[] b) throws IOException	将字节数组b中的b.length个字节数据输入到输出流
public void write(byte[] b,int off,int len) throws IOException	将字节数组b中的len个字节数据（从偏移量off开始）输入到输出流
public abstract void write(int b) throws IOException	将指定的字节输入到输出流

同样使用InputStream类是可能发生IOException异常的，所以在使用时是要进行异常处理的。

3. Reader类

Reader类是一个输入流，同样也是一个字符流。Reader类是所有输入字符流的超类，也就是说所有的输入字符流都派生自Reader类，其中提供了很多关于字符流输入操作的方法，表18-4列出了其中常用的一些。

表18-4 Reader类中的常用方法

方法名	功能
public abstract int read() throws IOException	读取单个字符，返回值的低16比特存放读取字符的编码（0~65535），高16比特忽略，如果因为已经到达流末尾而没有可读的字符，则返回值-1，往往都会将此方法的返回值强制类型转换成char类型
public int read(char[] cbuf) throws IOException	从输入流中读取一定数量的字符，并将其存储在缓冲区字符数组cbuf中，以整数形式返回实际读取的字符数。若流中实际可读的字符数小于数组cbuf的长度，则返回值会小于数组cbuf的长度，否则返回值等于数组cbuf的长度
public abstract int read(char[] cbuf,int off,int len) throws IOException	将输入流中最多len个字符读入字符数组cbuf。将读取的第一个字符存储在元素cbuf[off]中，下一个存储在cbuf[off+1]中，依次类推。方法的返回值为实际读取的字符数。若流中实际可读的字符数小于len，则返回值会小于len，否则返回值等于len
public long skip(long n) throws IOException	跳过此输入流中的指定数量的字符，参数n为指定的数量，返回值为实际跳过的字符数
public boolean ready() throws IOException	判断此字符输入流是否准备好被读，若是则返回true，否则返回false
public void close() throws IOException	关闭此输入流并释放与该流关联的所有系统资源
public void mark(int readAheadLimit) throws IOException	在此输入流中标记当前位置，参数readAheadLimit指出从此标记位置开始此流可以记忆的最大字符数，未来调用reset方法可以回到标记处重新读取字符。没有使用mark方法设置标记的流是不能回到某处再次读取的
public void reset() throws IOException	将此流重新定位到最后一次对此流调用mark方法时的位置
public boolean markSupported()	测试此输入流是否支持mark和reset方法，支持则返回true，否则返回false



同样使用 Reader 类是可能发生 IOException 异常的，所以在使用时是要进行异常处理的。

4. Writer 类

Writer 类是一个输出流，同样也是一个字符流。Writer 类是所有输出字符流的超类，也就是说所有的输出字符流都派生自 Writer 类，其中提供了很多关于字符流输出操作的方法，表 18-5 列出了其中常用的一些。

表 18-5 Writer类中的常用方法

方法名	功能
public void write(int c) throws IOException	将指定的字符写入此输出流，参数 c 表示要写入的字符。要注意的是，c 的低 16 个比特被作为一个字符写入流，而高 16 个比特被忽略
public void write(char[] cbuf) throws IOException	将指定字符数组 cbuf 的内容写入输入流
public abstract void write(char[] cbuf, int off, int len) throws IOException	将指定字符数组 cbuf 中从偏移量 off 开始的 len 个字符写入此输出流
public void write(String str) throws IOException	向流中写入指定字符串 str 的各个字符
public void write(String str, int off, int len) throws IOException	将指定字符串 str 中从偏移量 off 开始的 len 个字符写入此输出流
public void flush() throws IOException	刷新此输出流并强制写出所有缓冲中的输出字符
public void close() throws IOException	关闭此输出流并释放与此流有关的所有系统资源

同样使用 Reader 类是可能发生 IOException 异常的，所以在使用时是要进行异常处理的。字节流与字符流提供的方法很相似的。它们的不同主要体现在操作的基本单位不同，一个是以字节为基本单位，另一个是以字符为基本单位。



注意：字节流和字符流的不同主要体现在操作的基本单位不同，一个是以字节为基本单位，另一个是以字符为基本单位。



18.2 使用流进行文件操作

使用流来进行文件操作是流应用中最常见的应用之一。使用流能够读取文件内容，同样也能够写入文件内容。进行文件操作的类包括 File、FileInputStream、FileOutputStream、FileReader、FileWriter 等几个类。

18.2.1 使用 File 类进行文件与目录操作

Java 中专门提供了一个表示目录与文件的类——java.io.File，通过其可以获取文件、目录的信息，对文件、目录进行管理。File 类一共提供了 4 个构造器，表 18-6 列出了其中常用的三个。

在 File 类中最常用的是第一个构造函数。使用构造函数 public File(String pathname) 创建一个文件对象。其中，如果 pathname 是实际存在的路径，则该 File 对象表示的是目录；如果 pathname 是存在的文件名，则该 File 对象表示的是文件。

表 18-6 File类的常用构造器

构造器名	功 能
public File(String pathname)	通过指定的路径字符串 pathname 创建一个 File 对象，如果给定字符串是空字符串，那么创建的 File 对象将不代表任何文件或目录
public File(String parent, String child)	根据指定的父路径字符串 parent 以及子路径字符串 child 创建一个 File 对象。若 parent 为 null，则与单字符串参数构造器效果一样，否则 parent 将用于表示目录，而 child 则表示该目录下的子目录或文件
public File(File parent, String child)	根据指定的父 File 对象 parent 及子路径字符串 child 创建一个 File 对象。若 parent 为 null，则与单字符串参数构造器效果一样，否则 parent 将用于表示目录，而 child 则表示该目录下的子目录或文件

【范例 18-1】示例代码 18-1 是一个创建文件对象的程序。

示例代码 18-1

```

31 import java.io.*;
32 public class File1
33 {
34     public static void main(String[] args)
35     {
36         File f1 = new File("C:\\\\");           //创建 File 对象
37         File f2 = new File("C:\\\\a.txt");
38         if (f1.isDirectory())                  //判断 f1 对象是否为目录
39         {
40             System.out.println("f1 对象表示的是目录");
41         } else if (f1.isFile()) {              //判断 f1 对象是否为文件
42             System.out.println("f1 对象表示的是文件");
43         }
44         if (f2.isDirectory()) {              //判断 f2 对象是否为目录
45             System.out.println("f2 对象表示的是目录");
46         } else if (f2.isFile()) {            //判断 f2 对象是否为文件
47             System.out.println("f2 对象表示的是文件");
48         }
49     }
50 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-1 所示。

【代码解析】程序运行之前，必须先在 C 盘的根目录下创建一个 test.txt 文件。程序中创建了两个 File 对象，分别表示目录和文件；“isDirectory”与“isFile”为 File 类定义的两个方法，分别用于判断 File 对象表示的是目录还是文件。这里要注意如果在 C 盘没有 test.txt 这个文件，则不会有“f2 对象表示的是文件”的结果，程序会抛出异常。

创建了 File 对象后便可以通过其提供的方法来进行各种操作了，表 18-7 列出了 File 类中提供的一些常用方法。

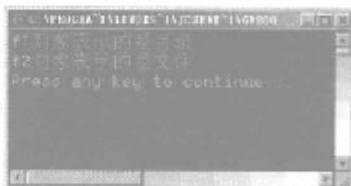


图 18-1 创建文件对象



表 18-7 File 类中的常用方法

方 法 名	功 能
public String getName()	返回此 File 对象表示的文件或目录的名称
public String getParent()	返回此 File 对象表示的文件或目录的父目录路径的名字字符串。如果没有父目录，则返回 null
public File getParentFile()	返回一个 File 对象，该对象将表示当前 File 的父目录。如果没有父目录，则返回 null
public String getPath()	返回此 File 表示文件或目录的路径字符串
public boolean isAbsolute()	测试此 File 是否采用的是绝对路径，若是则返回 true，否则返回 false
public String getAbsolutePath()	返回此 File 对象对应的文件或目录的绝对路径字符串
public boolean canRead()	测试 File 对象对应的文件是否是可读的，若是则返回 true，否则返回 false
public boolean canWrite()	测试 File 对象对应的文件是否是可写的，若是则返回 true，否则返回 false
public boolean canExecute()	测试 File 对象对应的文件是否是可执行的，若是则返回 true，否则返回 false
public boolean exists()	测试 File 对象对应的文件或目录是否存在，若是则返回 true，否则返回 false
public boolean isDirectory()	测试 File 对象表示的是否为目录，若是则返回 true，否则返回 false
public boolean isFile()	测试 File 对象表示的是否为文件，若是则返回 true，否则返回 false
public boolean isHidden()	测试 File 对象表示的是否为隐藏文件或目录，若是则返回 true，否则返回 false
public long lastModified()	返回 File 对象表示的文件或目录的最后修改时间，时间采用距离 1970 年 1 月 1 日 0 时的毫秒数来表示
public long length()	返回 File 对象表示的文件或目录的大小，以字节为单位
public boolean createNewFile() throws IOException	若 File 对象表示的文件不存在，可以调用此方法创建一个空文件。若创建成功则返回 true，否则返回 false
public boolean delete()	删除 File 对象表示的文件或目录，如果表示的是目录，则该目录必须为空才能删除。若成功删除则返回 true，否则返回 false
public String[] list()	若 File 对象表示的是一个目录，则调用此方法可以返回此目录中文件与子目录的名称。返回的名称都组织在一个字符串数组中
public File[] listFiles()	若 File 对象表示的是一个目录，则调用此方法可以返回此目录中文件与子目录对应的 File 对象，返回的 File 对象都组织在一个 File 数组中
public boolean mkdir()	创建此 File 指定的目录，若成功创建则返回 true，否则返回 false
public boolean mkdirs()	创建此 File 指定的目录，其中将包括所有必需但不存在的父目录。若成功创建则返回 true，否则返回 false

接下来看这些方法的使用。

[范例 18-2] 示例代码 18-2 是一个使用 exists 方法判断文件或目录是否存在的程序。

示例代码 18-2

```

import java.io.*;
public class Liu2
{
01     public static void main(String[] args)
02     {
03         File f1 = new File("C:\\b.txt");           // 创建一个 File 类对象
04         if (!f1.exists())                         // 判断文件或目录是否存在
05         {
06             try {
07                 System.out.println("文件不存在，创建该文件");

```



```

08         f1.createNewFile();           //如果不存在则创建文件
09         System.out.println("创建成功！");
10     } catch (IOException e) {
11         e.printStackTrace();
12     }
13 }
14 if (f1.isDirectory()) {           //判断对象是否为目录
15     System.out.println("f2 对象表示的是目录，目录名为：" + f1.getName());
16 } else if (f1.isFile()) {          //判断对象是否为文件
17     System.out.println("f2 对象表示的是文件，文件名为" + f1.getName());
18     System.out.println("路径为：" + f1.getPath());
19 }
20 }
21 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-2 所示。

再次运行程序，运行结果就会发生变化，运行结果如图 18-3 所示。

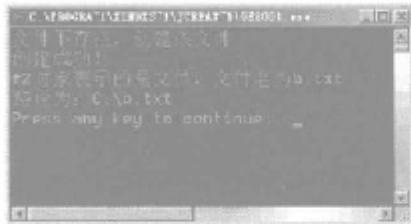


图 18-2 第一次运行结果

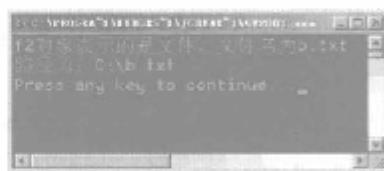


图 18-3 第二次运行结果

【代码解析】在本程序中的第 3 行首先创建了一个 File 对象，然后在第 4 行判断该 File 对象表示的目录或文件是否存在。如果没有该目录或文件，则在第 8 行使用 createNewFile 方法创建该目录或者文件，createNewFile 方法是可以发生异常的，所以需要进行异常处理。同样在本程序中使用了 isDirectory 方法和 isFile 方法来判断创建的 File 对象是目录还是文件。

18.2.2 FileInputStream 类与 FileOutputStream 类

FileInputStream 类是 InputStream 的子类。FileInputStream 类主要用于从文件系统中的某个文件中获取输入字节。OutputStream 类是 OutputStream 的子类，OutputStream 主要用于将数据以字节流写入目标文件的输出流。



提示：FileInputStream 类从文件系统中的某个文件中获取输入字节。主要用于读取诸如图像数据之类的原始字节流。要读取字符流，可以考虑使用 FileReader。

表 18-8 为 FileInputStream 中的方法声明及其使用描述。

表 18-8 FileInputStream 类的方法

方法返回类型	方法声明	方法描述
int	available()	返回可以从该文件输入流中读取的字节数
void	close()	关闭文件输入流，同时释放系统资源



续表

方法返回类型	方法声明	方法描述
protected void	finalize()	确认文件输入流不在被引用的状态，并可以调用 close 方法
java.nio.channels.FileChannel	getChannel()	返回与该文件输入流有关的 FileChannel 对象
FileDescriptor	getFD()	返回连接到文件系统（正被该 FileInputStream 使用）中实际文件的 FileDescriptor 对象
int	read()	从该输入流中读取一个数据字节，并以 int 类型返回
int	read(byte[] b)	读取输入流中 b.length 个字节的数据，并存入字节数组 b 中
int	read(byte[] b, int off, int len)	读取输入流中从偏移量 off 开始的 len 个字节的数据，并存入字节数组 b 中
long	skip(long n)	跳过输入流中的数据，同时丢弃 n 个字节的数据

【范例 18-3】示例代码 18-3 是一个使用 FileInputStream 类的程序。

示例代码 18-3

```

01 import java.io.*;
02 public class Liu3
03 {
04     public static void main(String[] args) throws Exception
05     {
06         File f=new File("c:\\c.txt"); //创建 File 对象
07         FileInputStream fis=new FileInputStream(f); //创建 FileInputStream 对象
08         char ch;
09         for(int i=0;i<f.length();i++)
10         {
11             ch=(char)fis.read(); //读取内容
12             System.out.print(ch); //显示
13         }
14         fis.close(); //关闭
15     }
16 }
```

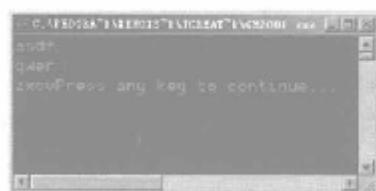


图 18-4 从文件中输出字符

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-4 所示。

【代码解析】在本程序中的第 6 行首先是创建了一个 File 对象，指定要进行操作的是哪一个文件。然后将 File 对象作为参数，创建一个 FileInputStream 对象。在最后使用 for 循环并通过 read 方法来获取文件中的内容，并显示在后台中。

与 FileInputStream 类相对， FileOutputStream 类用于将数据写入 File 或 FileDescriptor 的输出流。主要用于写入诸如图像数据之类的原始字节的流。要写入字符流，可以考虑使用 FileWriter。表 18-9 列出了 FileOutputStream 中的方法声明及其使用描述。

表 18-9 FileOutputStream类的方法

方法返回值	方法声明	方法描述
void	close()	关闭该文件输出流，同时释放系统资源



续表

方法返回值	方法声明	方法描述
protected void	finalize()	断开到文件的连接，确认该文件输出流不处在被引用状态，并可以调用 close 方法
java.nio.channels.FileChannel	getChannel()	返回与该文件输出流有关的 FileChannel 对象
FileDescriptor	getFD()	返回与该流有关的文件描述符 FileDescriptor 对象
void	write(byte[] b)	将 b.length 个字节的数据从指定字节数组写入到该文件输出流中
void	write(byte[] b, int off, int len)	将指定字节数组中的部分数据（从偏移量 off 开始的 len 个字节）写入到该文件输出流
void	write(int b)	将指定字节 b 写入到该文件输出流

【范例 18-4】示例代码 18-4 是一个使用 FileOutputStream 类的程序。

示例代码 18-4

```

01 import java.io.*;
02 public class Liu4
03 {
04     public static void main(String[] args) throws IOException
05     {
06         File f=new File("c:\\d.txt");           //创建 File 对象
07         FileOutputStream fos=new FileOutputStream(f); //创建 FileOutputStream 对象
08         for(int i='a';i<='z';i++)
09         {
10             fos.write(i);                      //写入
11         }
12         fos.close();                         //关闭
13     }
14 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-5 所示。

从后台是看不出变化来的，打开 C 盘下 d.txt 文件，运行结果如图 18-6 所示。

【代码解析】在本程序中同样是要先创建一个 File 对象，然后将该对象作为参数，创建一个 FileOutputStream 对象。然后使用字符循环，将这些字符都写入指定的文件中。从文件的显示结果中可以看出显示从 a 到 z 的字符。

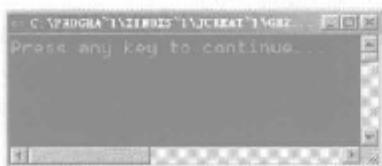


图 18-5 写入字节

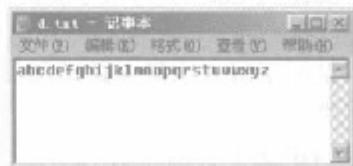


图 18-6 写入信息后的文件



提示： FileInputStream 类和 FileOutputStream 类经常一起配合使用的，最经常的使用就是读取一个文件中的信息，然后将信息复制到另一个文件中。



【范例 18-5】示例代码 18-5 是一个 FileInputStream 类和 FileOutputStream 类结合使用的程序。

示例代码 18-5

```
01 import java.io.*;
02 public class Liu5
03 {
04     public static void main(String[] args) throws IOException
05     {
06         File f1=new File("c:\\1.txt");           //创建两个 File 对象
07         File f2=new File("c:\\2.txt");
08         FileInputStream fis=new FileInputStream(f1); //创建 FileInputStream 对象
09         FileOutputStream fos=new FileOutputStream(f2); //创建 FileOutputStream 对象
10         byte[] b=new byte[(int)f1.length()];
11         fis.read(b);                         //读取 1 文件
12         for(int i=0;i<f1.length();i++)
13         {
14             fos.write(b[i]);                //写入 2 文件
15         }
16         fis.close();
17         fos.close();
18     }
19 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-7 所示。

【代码解析】在本程序中首先创建了两个 File 对象，要对应的在指定的位置创建这两个文件。其中 f1 是一个被读取文件，f2 是一个要写入的文件。该程序看起来功能很复杂，其实只是简单地从一个文件读取信息，然后将信息写入另一个文件。

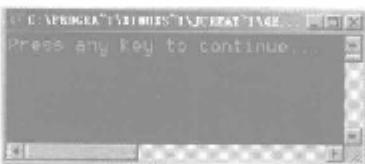


图 18-7 FileInputStream 类和 FileOutputStream 类结合使用

18.2.3 FileReader 类与 FileWriter 类

FileReader 与 FileWriter 和 FileInputStream 与 FileOutputStream 类似，所不同的是它们是针对字符进行操作，而不是字节。这两个类的间接父类是字符流 Reader 和 Writer。其中，FileWriter 是用于写入字符文件的便捷类。FileReader 是用于读取字符文件的便捷类。在 FileReader 类及 FileWriter 类中未自定义方法，继承了其父类及间接父类中的方法。

【范例 18-6】示例代码 18-6 是一个 FileReader 类与 FileWriter 类结合使用的程序。

示例代码 18-6

```
01 import java.io.*;
02 public class Liu6
03 {
04     public static void main(String[] args)
05     {
```

```

06     File f = new File("C://aaa.txt");           //创建一个File类对象
07     String str = "FileReader and FileWriter";
08     char[] buffer = new char[str.length()];      //创建一个char数组,初始化
09     buffer = str.toCharArray();                  //为str字符串对象的长度大小
10
11     try {
12         if (!f.exists())
13         {
14             f.createNewFile();
15         }
16         FileWriter writer = new FileWriter(f);
17         writer.write(buffer);                    //输出整个字符数组到文件
18         writer.close();
19         //从文件中读取数据
20         FileReader reader1 = new FileReader(f); //从文件头读取所有数据
21         char[] result1 = new char[20];
22         int i1 = reader1.read(result1);
23         System.out.println("读取文件中所有字符, 字符数为:" + i1);
24         System.out.println(result1);
25         FileReader reader4 = new FileReader(f);
26         char[] result4 = new char[100];          //从文件中逐个读取字符, 直
27                                         //到文件末尾, 返回-1
28         int i = 0;
29         System.out.println("从文件中逐个读取字符:");
30         while (i != -1)
31         {
32             i = reader4.read();
33             if (i != -1)
34                 System.out.print(String.valueOf(Character.toChars(i))
35             );
36         }
37     }
38     catch (IOException e)
39     {
40         e.printStackTrace();
41     }
42 }
43 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 18-8 所示。

【代码解析】本程序是一个使用 FileReader 类与 FileWriter 类进行文件读写的程序。在本程序中要读写的内容为一个字符所组成的数组。首先是使用 FileWriter 类的 write 方法将该字符数组写入到文件中，这样在该文件中就具有“FileReader and FileWriter”内容。接下来就是创建一个 FileReader 对象来读取该文件。在读取文件时，首先获取文件中字符的数量，然后显示全部字符。从程序的第 29 行到第 36 行是将文件中的字符逐个显示。

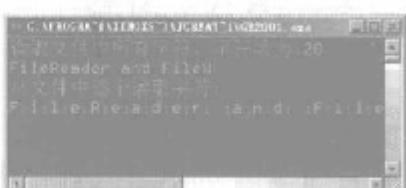


图 18-8 FileReader 类与 FileWriter 类



18.3 综合练习

字节流和字符流主要区别有哪些？

【提示】字节流是最基本的，所有 `InputStream` 和 `OutputStream` 类的子类都是字节流，其主要用于处理二进制数据，并按字节来处理。

但是实际开发中很多的数据是文本，这就提出了字符流的概念，它按虚拟机的 `encode` 来处理，也就是要进行字符集的转化。这两者之间通过 `InputStreamReader` 和 `OutputStreamWriter` 类来关联。实际上，通过 `byte[]` 和 `String` 来关联在实际开发中出现的汉字问题，这都是在字符流和字节流之间转化不统一而造成的。在从字节流转化为字符流时，实际上就是 `byte[]` 转化为 `String`。

```
public String(byte bytes[], String charsetName)
```



注意：有一个关键的参数字符集编码，通常可以省略，就是操作系统的 lang。

字符流转化为字节流，实际上是 `String` 转化为 `byte[]`。

```
byte[] String.getBytes(String charsetName)
```

至于 `Java.io` 中还出现了许多其他的流，主要是为了提高性能和使用方便，如 `BufferedInputStream`、`PipedInputStream` 等。



18.4 小结

在本章中主要对 Java 中的流进行了详细的讲解，同时也讲解了如何使用流进行文件操作。进行文件操作是流的最重要作用之一，这里对其进行详细的讲解。如果读者想了解更多的流的知识，可以参考电子工业出版社出版的《Java 程序设计经典教程：融合上机操作实例》一书进行学习。



18.5 习题

一、填空题

1. Java 中的数据流分为两种，一种是_____，另一种是_____。
2. 根据流功能层次的不同可以将其分为两类：_____与_____。
3. 根据流处理数据类型的不同也可以将其分为两类：_____与_____。
4. 字节流和字符流的抽象基类包括_____类、_____类、_____类和_____类组成。
5. 使用构造函数 `public File(String pathname)` 创建一个文件对象。其中，如果_____是实际存在的路径，则该_____表示的是目录；如果_____是存在的文件名，则该_____表示的是文件。



二、选择题

1. 在 InputStream 类中表示读取下一个数据字节的方法是()。
A. close()方法 B. mark 方法 C. read()方法 D. reset()方法
2. 在 OutputStream 类中表示刷新输出流，同时输出所有缓冲的输出字节的方法是()。
A. close()方法 B. flush() C. write 方法 D. reset()方法
3. 在 FileInputStream 类中表示跳过输入流中的数据，同时丢弃 n 个字节的数据的方法是()。
A. close()方法 B. skip 方法 C. read()方法 D. reset()方法
4. 在 FileInputStream 类中表示将指定字节 b 写入到该文件输出流的方法是()。
A. close()方法 B. flush() C. write 方法 D. reset()方法

三、简答题

1. 什么是流，Java 中的流的分类是按照什么来分的，分为哪几类。
2. 在 Java 中，如何使用流来对文件进行操作。

四、编程题

1. 编写一个读取文件的程序，在该程序中能够依次打印文件中的每一个字符。
2. 编写一个程序，在该程序中能够读取 A 文件，并把 A 文件的内容写入到 B 文件中。

第19章 集合框架

在日常生活中，放衣服是一门学问，是把衬衫和裤子等衣服都放在一个盒子中呢？还是每一件衣服放在一个盒子中呢？读者都知道这两种做法都不好，通常都是将一类衣服放在一起。同样，在Java中也提供了这样的功能，那就是集合框架。在前面已经学习了数组，集合框架也是和数组一样来保存一组数据。集合框架主要包括列表、集合和映射。通过本章的学习，读者应该实现如下几个目标。

- 了解什么是集合框架和集合框架包括哪些形式。
- 掌握什么是列表和列表中包括哪些类和接口。
- 掌握什么是集合和集合中包括哪些类和接口。
- 掌握什么是映射和映射中包括哪些类和接口。



19.1 集合框架总论

集合是某一类对象的通称，这类对象代表以某种方式组合到一起的一组对象；它是将多个元素组合为一个单元的对象，用于存储、检索、操纵和传递数据。而对象的集合，指的是对象引用的集合而不是对象的集合，在Java集合中只存储引用，对象在集合之外。集合框架提供用于管理对象集合的接口和类，它包括接口、实现和算法。

19.1.1 什么是集合框架

集合框架是Java提供的可定义对象，该对象由其他的对象组成，如常见的向量(Vector)类。集合框架是一个统一的可以代表及操作集合，并能对这些集合独立的进行操作的一些结构。集合框架的主要优点在于提高了编码效率、性能和复用性。

Java集合框架提供了有效的数据结构及算法，因此程序员不需要自己编写代码实现这些功能。Java集合框架提供了高性能的数据结构及算法的实现。因为对各个接口的实现是可以互换的，因此程序很容易转换接口。提高了软件的复用性，软件可以提供标准的集合框架的接口对其进行操作。

Java集合框架主要由一组用来操作对象的接口组成。不同接口描述一组不同数据类型。在这些接口中Collection接口是层次结构中的根接口。在Collection接口中具有开发中经常用到的set接口、list接口和map接口。

19.1.2 Collection接口

上一节中已经介绍，Collection接口是集合继承树中最顶层的接口，该接口声明了集合中常用到的一些通用方法，表19-1中给出了这些方法。

这些方法都是集合框架中最基本的方法，由于其他接口都继承Collection接口，所以这些接口中都继承了这些方法。

表 19-1 Collection类的方法名及说明

方法名	说 明
boolean add(Object o)	该方法添加参数指定的元素 o 至集合中
boolean addAll(Collection c)	该方法将参数集合 c 中所有的元素都添加到该集合中
void clear()	该方法将删除该集合中所有的元素
boolean contains(Object e)	该方法用于判断元素 e 是否在该集合中, 如果是则返回 true, 否则返回 false
boolean containsAll(Collection c)	该方法用于判断该集合中是否包含参数 c 集合中的所有元素, 如果是则返回 true, 否则返回 false
boolean equals(Object o)	这个方法将比较该集合与参数指定的对象 o。如果相等, 返回 true, 否则返回 false
int hashCode()	该方法将返回集合的哈希码的整型表现形式的值
boolean isEmpty()	判断该集合是否为空, 如果为空则返回 true, 否则为 false
Iterator iterator()	返回该集合的迭代器, iterator, 可以用户遍历集合中的元素
boolean remove(Object o)	如果集合中存在参数指定的元素 o, 则将该元素从集合中删除
boolean removeAll(Collection c)	将该集合中的某些元素删除, 这些元素是参数 c 中存在的
boolean retainAll(Collection c)	与 removeAll(Collection c)方法相反, 将集合中所有的非参数 c 中的元素删除, 仅保留 c 中存在的元素
int size()	返回该集合的大小, 即其中的元素个数
Object[] toArray()	将该集合中的元素, 以数组的形式返回。数据的类型为 Object
Object[] toArray(Object[] a)	返回包含此 collection 中所有元素的数组; 返回数组的运行时类型与指定数组的运行时类型相同



19.2 列表

List 列表作为集合中的一种, 其主要特点在于其中的元素保持一定的顺序, 并且元素是可以重复的。本节将具体讲解 List 的使用及其实现类(如 ArrayList、LinkedList)的使用。List 接口继承自 Collection 接口, 代表列表的功能(角色), 其中的元素可以按索引的顺序访问, 所以也可以称之为有索引的 Collection。实现该接口的类均属于 Ordered 类型, 具有列表的功能, 其元素顺序均是按添加(索引)的先后进行排列的。

19.2.1 List 列表接口

除了继承了 Collection 声明的方法外, List 接口在 iterator、add、remove、equals 和 hashCode 方法的基础上加了一些其他约定, 超过了 Collection 接口中指定的约定。同时, List 比 Collection 多了 10 个方法, 这些方法可以分为访问方法、迭代器方法、搜索方法和插入、删除方法。

List 接口声明了三种对列表元素进行定位(索引)访问方法。

- `Object get(int index)`: 参数 index 表示将要需要得到元素的索引。该方法将返回此列表中指定 index 位置上的元素。
- `List subList(int fromIndex, int toIndex)`: 参数 fromIndex 为指定的起始索引, 参数 toIndex 为指定的结束索引, 该方法将返回一个新的列表, 这个新的列表将包含原来列表中从指定的起始索引到指定的结束索引并且不包含结束索引的元素。



- `Object[] toArray()`: 该方法将类表转换成一个 `Object` 类型的对象数组，该数组用元素的顺序与列表中元素的顺序相同，并将该数组返回。

提示: `List` 接口声明了特殊的迭代器，称为 `ListIterator`，除了允许 `Iterator` 接口提供的正常操作外，该迭代器还允许元素插入和替换，以及双向访问。还提供了一个方法来获取从列表中指定位置开始的列表迭代器。`List` 接口提供了对 `ListIterator` 的获取的两种方法，分别是 `listIterator` 方法和 `listIterator(int index)` 方法。

`List` 接口声明了两种搜索指定对象的方法。从性能的观点来看，应该小心使用这些方法。在很多实现中，这些方法将执行高开销的线性搜索。`List` 接口对其声明如下：

- `int indexOf (Object o)`: 参数 `o` 为指定查找的元素，该方法将遍历整个列表查找指定元素 `o`，若列表中存在，则返回第一个找到的元素的索引，若列表中不存在，则返回负数。
- `int lastIndexOf (Object o)`: 参数 `o` 为指定查找的元素，该方法将遍历整个列表查找指定元素 `o`，若列表中存在，则返回最后一个找到的元素的索引，若列表中不存在，则返回负数。

`List` 接口声明了两种在列表的任意位置高效插入和删除元素的方法。

- `Object set (int index, Object element)`: 参数 `index` 表示需要替换元素的索引。参数 `o` 表示将要替换为的元素。该方法操作成功后将返回替换掉的元素。
- `Object remove (int index)`: 参数 `index` 为将要移除元素的索引。返回从列表中移除的元素。该方法操作后所有后续元素均向前移动，即列表中间不能有空位。
- `boolean remove (Object o)` 方法：参数 `o` 为指定的需要移除的元素。若列表包含一个或多个与指定 `o` 相同的元素，则移除该元素，并返回 `true`，否则返回 `false`。
- `boolean removeAll (Collection c)` 方法：参数 `c` 为包含指定需要移除元素的 `Collection`，该方法将列表中有的并且 `c` 中有的元素从 `set` 中移除，若有元素被移除 `set` 则返回 `true`，否则返回 `false`。
- `boolean retainAll (Collection c)` 方法：参数 `c` 为包含指定需要保留元素的 `Collection`，该方法将列表中有的并且 `c` 中没有的元素从 `set` 中移除，若有元素被移除 `set` 则返回 `true`，否则返回 `false`。

`java.util.List` 的几种实现中，有三种最为常用的实现类，这三个类分别是 `Vector` 类、`ArrayList` 类和 `LinkedList` 类。接下来将逐个介绍。

19.2.2 Vector 类

`Vector` 类也称为向量，从 Java 一诞生就有，后来被作为集合框架的一部分，其性能特点与 `ArrayList` 基本上是相同的。不同之处是该类的功能方法是同步的，同一时刻只能有一个线程访问，没有特殊需要，现在一般都使用 `ArrayList`，`ArrayList` 会在下一节中讲解。



注意: `Vector` 类与 `ArrayList` 不同之处是该类的功能方法是同步的，同一时刻只能有一个线程访问。

`Vector` 类具有 4 种构造器，表 14-6 列出了 `Vector` 类的 4 个构造器。

表 19-2 Vector类的构造器

构造器名	功 能
public Vector()	该构造器将构造一个空的 Vector 对象。该对象的初始容量为 10
public Vector(int initialCapacity)	参数 initialCapacity 表示指定的初始容量，该构造器将构造一个具有指定容量的空 Vector 对象
public Vector(int initialCapacity,int capacityIncrement)	参数 initialCapacity 表示指定的初始容量，参数 capacityIncrement 表示当 Vector 满了时候，其容量的增量。该构造器将构造一个具有指定初始容量与指定增量的空 Vector 对象
public Vector(Collection c)	参数 c 为包含指定元素的 Collection。该构造器将构造一个以 c 中的元素为初始内容的 Vector 对象

Vector 提供了用于增加元素的方法，方法如下所述。

- public void addElement (Object obj) 方法：该方法是将指定的组件添加到该向量的末尾，并将其大小增加 1。如果向量的大小比容量大，则增大其容量。
- public void addElement(int index, Object obj) 方法：该方法在该向量的指定位置 index 插入指定的元素 obj，并将当前位于该位置的元素及所有后续元素右移（即将元素索引加 1）。如果索引超出范围（即 $index < 0 \text{ 或 } index > size()$ ），则程序抛出 ArrayIndexOutOfBoundsException 异常。
- public void insertElementAt (Object obj,int index) 方法：该方法将指定对象作为此向量中的组件插入到指定的 index 处。

【范例 19-1】示例代码 19-1 是一个使用 Vector 类增加元素的程序。

示例代码 19-1

```

01 import java.util.*;
02 public class JiHe1
03 {
04     public static void main(String[] args)
05     {
06         //创建一个Vector 对象,容量初始化为5
07         Vector<String> v1 = new Vector<String>(5); //这时参数与返回值都必须
08                                         //为 String 类型
09         v1.addElement(new String("one"));           //添加一个字符串对象
10         v1.addElement("three");
11         v1.addElement("four");
12         v1.insertElementAt("zero", 0);              //在位置 0 插入一
13                                         //个字符串对象
14         v1.insertElementAt("two", 2);
15         v1.insertElementAt("five", 5);
16         System.out.println("v1:" + v1);            //输出 Vector 对
17                                         //象中的元素值
18         System.out.println("v1 的容量为: " + v1.capacity()); //输出 Vector 对
19                                         //象的容量
20         //创建一个Vector 对象,容量初始化为5,容量的增量为1
21         Vector<String> v2 = new Vector<String>(5, 1); //这时参数与返回
22                                         //值都必须为
23                                         //String 类型
24         v2.addElement("one");                      //添加一个字符串
25                                         //对象

```



```

21     v2.addElement("three");
22     v2.addElement("four");
23     v2.insertElementAt("zero", 0);           //在位置 0 插入一
24     v2.insertElementAt("two", 2);           //个字符串对象
25     v2.insertElementAt("five", 5);
26     System.out.println("v2:" + v2);
27     System.out.println("v2 的容量为：" + v2.capacity()); //输出 Vector 对
28                           //象的容量
}

```

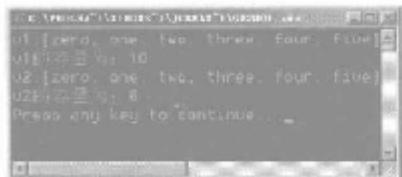


图 19-1 Vector 类增加元素

方法的使用，通过结果可以看出两种不同构造函数下创建的 Vector 对象在容量增长方面的区别。

Vector 类还具有删除、查询和修改其中元素的方法，这些方法的使用是和添加方法的使用相同的，这里就不再进行重复的讲解。下面给出一个进行综合操作的程序。

【范例 19-2】示例代码 19-2 是一个使用 Vector 类的程序。

示例代码 19-2

```

01 import java.util.*;
02 public class JiHe2
03 {
04     public static void main(String[] args)
05     {
06         //创建 Vector 类的对象
07         Vector v=new Vector();
08         //初始化 Vector 对象中的元素值
09         for(int i=0;i<5;i++)
10         {
11             v.add(String.valueOf(i));
12         }
13         //对 Vector 进行操作
14         for(int i=5;i<10;i++)
15         {
16             v.add(String.valueOf(9-i+5));
17         }
18         //打印 Vector 列表
19         System.out.println("这里是 Vector 操作后的结果：");
20         System.out.println(v);
21     }
22 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-2 所示。

【代码解析】示例代码 19-2 中分两次一共向 Vector 对象中添加了 10 个元素，第一次添加的为 0~4 的字符串，

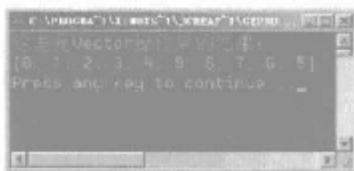


图 19-2 进行 Vector 操作

第2次添加的为9~5的字符串。在平常的开发中Vector对象是使用不多的，这里只是一个简单的介绍。

19.2.3 ArrayList类

本节主要向读者介绍ArrayList类，其是List接口最常用的实现之一，可以向其中添加包括null值在内的所有对象引用型的元素，甚至该类对象引用自己也可以作为其中的元素，这样便可以方便地搭建一个树状结构的集合。

ArrayList有三种构造方法，方法如下所示。

- public ArrayList()方法：该构造器将构造一个空的ArrayList对象。该对象的初始容量为10。
- public ArrayList(int initialCapacity)方法：参数initialCapacity表示指定的初始容量，该构造器将构造一个具有指定容量的空ArrayList对象。
- public ArrayList(Collection c)方法：参数c为包含指定元素的Collection。该构造器将构造一个以c中的元素为初始内容的ArrayList对象。



注意：该类内部实际上是依赖数组实现的，因此对元素进行随机访问的性能很好。但是，如果进行大量的插入、删除操作，此类的性能很差，并不适合。

ArrayList类和Vector类一样，同样也具有很多的方法，这里不可能为每一种方法都给出程序。表19-3列出了所有的方法。

表19-3 ArrayList类中的方法

方法名	方法描述
public boolean add(Object o)	该方法将指定的元素o追加到此列表的末尾
public void add(int index, Object o)	该方法将指定的元素o插入此列表中的指定index索引位置
public boolean addAll(Collection c)	该方法将对象c中的所有元素追加到此列表的末尾
public boolean addAll(int index, Collection c)	该方法将参数c中的所有元素从指定的index索引位置开始插入到此列表中
public void clear()	该方法删除列表对象中的所有元素
public Object clone()	该方法返回此ArrayList对象的复制对象，返回为Object对象
public boolean contains(Object elem)	该方法用于判断此列表中是否包含指定的元素elem，如果包含则返回true
public void ensureCapacity(int minCapacity)	该方法用于增加此ArrayList对象的容量，以确保它至少能够容纳最小容量参数所指定的元素数
public E get(int index)	该方法返回此列表对象中指定索引index位置上的元素
public int indexOf(Object elem)	该方法将返回给定参数elem元素第一次出现的位置
public boolean isEmpty()	该方法将判断此列表中是否没有元素
public int lastIndexOf(Object elem)	该方法返回指定的对象elem元素在列表中最后一次出现的位置索引
public Object remove(int index)	该方法删除此列表中指定位置index上的元素
public boolean remove(Object o)	该方法从此该列表中删除指定元素o
protected void removeRange(int fromIndex, int toIndex)	该方法删除列表中索引在fromIndex和toIndex之间的所有元素。包括fromIndex，不包括toIndex
public Object set(int index, Object o)	该方法用指定的元素替代此列表中指定位置上的元素



续表

方法名	方法描述
public int size()	该方法返回此列表中的元素数
public Object[] toArray()	该方法返回一个此列表中所有元素的数组
public <T> T[] toArray(T[] a)	该方法返回一个此列表中所有元素的数组，返回的数组存在在参数 a 中
public void trimToSize()	该方法将此 ArrayList 实例的容量调整为列表的当前大小

【范例 19-3】示例代码 19-3 是一个使用 ArrayList 类的程序。

示例代码 19-3

```

01 import java.util.*;
02 public class JiHe3
03 {
04     public static void main(String[] args)
05     {
06         //创建列表ArrayList 的对象
07         ArrayList al=new ArrayList();
08         //初始化ArrayList 对象中的元素
09         for(int i=0;i<5;i++)
10         {
11             al.add(String.valueOf(i));
12         }
13         //对ArrayList 进行操作
14         for(int i=6;i<8;i++)
15         {
16             al.set(i-5,String.valueOf(i));
17         }
18         //打印ArrayList 列表中的内容
19         System.out.println("这里是ArrayList 操作后的结果: ");
20         System.out.println(al);
21         //取出指定索引的元素并处理
22         Object o=al.get(3);
23         String s=(String)o;
24         System.out.println("索引为 3 的元素长度为: "+s.length());
25     }
26 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-3 所示。

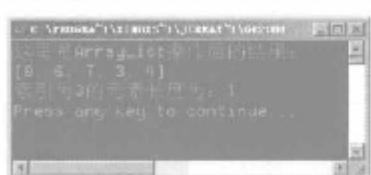


图 19-3 使用 ArrayList 类

【代码解析】集合框架中的类大都在 java.util 包中，因此编程时要在代码开始加上“import java.util.*;”。在没有使用泛型的情况下，无论放进集合（这里的集合指所有 Collection 接口的实现）的是什么类型的元素，取出来的都是 Object 类型的引用，若需要当作具体类型使用需要进行强制类型转换。

ArrayList 类中提供了可以删除其中元素的方法，其方法声明及使用说明如下所示。

- public E remove (int index) 方法：该方法删除列表中参数指定位置 index 上的元素。向左移动所有后续元素，即将其索引减 1。其中参数 index 为要删除元素的索引。
- public boolean remove (Object o) 方法：该方法删除列表指定元素 o。如果列表中包含指定的元素，则返回 true，否则为 false。

- public void clear()方法：该方法删除列表中所有的元素。调用这个方法后，列表为空。

【范例 19-4】示例代码 19-4 是一个在 ArrayList 类中进行删除操作的程序。

示例代码 19-4

```

01 import java.util.*;
02 public class JiHe4
03 {
04     public static void main(String[] args)
05     {
06         //创建一个ArrayList 对象
07         ArrayList<String> col = new ArrayList<String>(); //参数与返回值都必须为
08                                         //String 类型
09         for (int i = 0; i < 5; i++)
10             {                                //循环加入 5 个字符串对象
11                 col.add(String.valueOf(i));
12             }
13         System.out.println("开始时 col 中的元素有: ");
14         for (Iterator<String> iter = col.iterator(); iter.hasNext();)
15             {                                //遍历输出对象 col 中的元素
16                 System.out.println("the element is :" + iter.next());
17             }
18         col.remove(1);                      //删除对象 col 中位置 1
19                                         //的元素
20         System.out.println("在删除其中一个元素后 col 中的元素有: ");
21         for (int i = 0; i < col.size(); i++) //顺序输出对象 col 中
22                                         //的元素
23             {
24                 System.out.println("element at " + i + " is " + col.get(i));
25             }
26     }
27 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-4 所示。

【代码解析】在本示例中，首先声明了一个 ArrayList 类型的对象，对其添加元素后，将其结果输出；接着利用 remove 方法，将其中位于索引为 1 的元素删除掉，再将其中所有元素输出。根据结果可以看到 remove 方法的作用。注意到在这个示例中使用了 size 方法获取该 ArrayList 的长度，这里是指其中包含元素的个数；并利用了 get 方法，获取了相应索引值的元素。因此可以看出所有位于 1 索引值后的元素全部前移，其索引值减一。

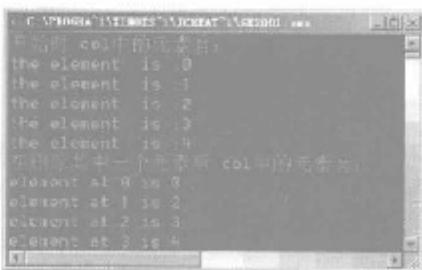


图 19-4 删除操作

19.2.4 LinkedList 类

本节主要介绍 LinkedList 类，其功能与 ArrayList、Vector 相同，都是列表（List）的实现。其内部是依赖双链表来实现的，因此具有很好的插入删除性能，但随机访问元素的性能相对较差，适合用在插入、删除多，元素随机访问少的场合。表 19-4 列出了 LinkedList



类的几个构造器。

表 19-4 LinkedList类的构造器

构造器名	功 能
public LinkedList()	该方法将构造一个空的列表
public LinkedList(Collection c)	构造一个包含指定集合 c 中的所有元素的列表，这些元素按其集合的迭代器返回的顺序排列；其中参数 c 为要将其元素放到此集合中的列表；如果指定的集合为 null，程序抛出 NullPointerException 异常

LinkedList 没有同步方法，如果多个线程同时访问一个 List，则必须自己实现访问同步，一种解决方法是在创建 List 时构造一个同步的 List。

【范例 19-5】示例代码 19-5 是一个使用 LinkedList 类的程序。

示例代码 19-5

```

01 import java.util.*;
02 public class JiHe5
03 {
04     public static void main(String[] args)
05     {
06         //创建列表LinkedList 类的对象
07         LinkedList ll=new LinkedList();
08         //初始化LinkedList 对象
09         for(int i=0;i<5;i++)
10         {
11             ll.add(String.valueOf(i));
12         }
13         //对LinkedList 进行插入操作
14         for(int i=5;i<10;i++)
15         {
16             ll.add(i,String.valueOf(10-i));
17         }
18         //打印LinkedList 列表
19         System.out.println("这里是LinkedList 操作后的结果:");
20         System.out.println(ll);
21     }
22 }
```

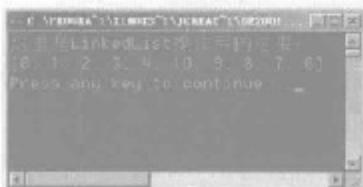


图 19-5 LinkedList 类

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-5 所示。

【代码解析】和前面的程序一样，该程序是一个使用 LinkedList 类进行插入操作的程序。有了前面学习的基础，该程序是很容易理解的。

LinkedList 类中同样具有很多方法。在前面学习 Vector 类和 ArrayList 类时，已经使用添加和删除方法进行了举例，这里就类使用 LinkedList 类中提供了获取或修改某个位置的元素方法来举例。获取和删除方法包括如下几个方法。

- public E element()方法：该方法找到但不删除此列表的头（即第一个元素），这个方法是实现了接口 Queue 中的同名方法。
- public E get (int index) 方法：该方法返回此列表中指定位置 index 处的元素。如



果指定的索引超出范围 ($\text{index} < 0 \text{ 或 } \text{index} >= \text{size}()$)，则程序抛出 `IndexOutOfBoundsException` 异常。

- `public E getFirst()`方法：该方法返回此列表的第一个元素。如果此列表为空，则程序抛出 `NoSuchElementException` 异常。
- `public E getLast()`：该方法返回此列表的最后一个元素。如果此列表为空，则程序抛出 `NoSuchElementException` 异常。
- `public int indexOf (Object o)`方法：该方法返回此列表中元素 `o` 的索引，需要注意的是，如果包含多个 `o`，则返回为第一次出现的索引值。如果列表中不包含此元素，则返回-1。这个方法是覆盖其父类 `AbstractList<E>` 中的 `indexOf` 方法。其中参数 `o` 为要搜索的元素。
- `public int lastIndexOf (Object o)`方法：该方法返回此列表中最后出现的指定元素的索引，如果列表中不包含此元素，则返回-1。参数 `o` 为要搜索的元素。
- `public E peek()`方法：该方法找到但不删除此列表的头，即第一个元素。如果此队列为空，则返回 `null`。
- `public E poll()`方法：该方法找到并删除此列表的头，即第一个元素。如果此队列为空，则返回 `null`。
- `public E set (int index,E element)`方法：该方法将此列表中指定位置的元素替换为指定的元素。其中参数 `index` 为要替换的元素的索引，`element` 为要在指定位置存储的元素。

【范例 19-6】示例代码 19-6 是一个使用 `LinkedList` 类进行设置和删除操作的程序。

示例代码 19-6

```

01 import java.util.*;
02 public class JiHe6
03 {
04     public static void main(String[] args)
05     {
06         //创建一个LinkedList 对象
07         LinkedList<String> liList = new LinkedList<String>();
08         //参数与返回值都必须为 String 类型
09         for (int i = 0; i < 5; i++)
10             {
11                 liList.add(String.valueOf(i));
12             }
13         System.out.println(liList);
14         System.out.println(liList.get(3));
15         liList.set(3, "aaa");
16         System.out.println(liList);
17         System.out.println(liList.peek());
18         System.out.println(liList.poll());
19         System.out.println(liList);
20         System.out.println("第一个元素是" + liList.getFirst());
21         System.out.println("最后一个元素是" + liList.getLast());
22     }
23 }
```



【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-6 所示。



图 19-6 LinkedList 类获取和修改方法

【代码解析】在该程序的第 13 行使用 get 方法来获取位置 3 的元素，然后使用 set 方法指定“aaa”来替代该元素。接下来在第 16 行使用 peek 方法来找到列表的第一个元素，而第 18 行的 poll 方法则是找到列表的第一个元素并删除。在程序的最后，使用 getFirst 方法和 getLast 方法来获取第一个元素和最后一个元素。



19.3 集合

Set 集合是一种不包含重复元素的 Collection，即任意的两个元素 e1 和 e2 比较，结果都不相等。注意，Set 的构造函数有一个约束条件，传入的 Collection 参数不能包含重复的元素。必须小心操作可变对象（Mutable Object）。如果一个 Set 中的可变元素改变了自身状态，Object.equals(Object)=true 会发生一些问题。

19.3.1 Set 接口

Set 接口与 List 接口最大的区别在于 Set 中没有重复的元素。Set 是非常简单的集合，Set 中的对象没有特定顺序。Sorted 接口具有排序的功能，TreeSet 类则是实现了该接口；HashSet 类使用哈希算法存取集合中的元素，存取速度比较快。Set 接口声明如下所示。

```
public interface Set<E> extends Collection<E>
```

提示：Set 接口与 List 接口最大的区别在于 Set 中没有重复的元素。

Set 接口的 API 中所有方法如表 19-5 所示；读者可以通过下面几节的内容学习其具体使用方法。

表 19-5 Set 接口的方法

方法修饰符	方法名	方法描述
boolean	add(E e)	将参数指定的元素 e 添加至集合中，如果该集合中已经存在该元素，则该集合不变化
boolean	addAll(Collection<?extends E> c)	如果该集合中不存在所有的元素，将参数指定 c 集合中的所有元素添加至集合中
void	clear()	该方法删除集合 set 中的所有元素

续表

方法修饰符	方法名	方法描述
boolean	contains(Object o)	该方法判断参数指定的元素 o 是否存在于集合中，如果存在则返回 true，否则为 false
boolean	containsAll(Collection<?> c)	该方法判断参数 c 中所有的元素是否存在于集合中，如果存在则返回 true，否则返回 false
boolean	equals(Object o)	该方法用于比较指定的对象 o 是否与集合相等，如果相等，则返回 true，否则返回 false
int	hashCode()	该方法将返回该集合的哈希码的整数形式的值
boolean	isEmpty()	该方法将判断该集合是否为空，如果为空则返回 true，否则为 false
Iterator<E>	iterator()	该方法返回该集合的迭代器 iterator，可以用于遍历集合中的元素
boolean	remove(Object o)	该方法将删除集合中的参数指定的元素 o，如果集合中不存在元素，则该集合不变化
boolean	removeAll(Collection<?> c)	该方法删除集合中与参数集合 c 中元素相同的所有元素
boolean	retainAll(Collection<?> c)	与方法 retainAll(Collection<?> o) 的作用正好相反，该方法仅保留参数指定 c 的元素。该集合中的其他元素都被删除掉
int	size()	返回该集合的元素个数，即其容量
Object[]	toArray()	将该集合中的元素以数组的形式返回，数组类型为 Object
<T>T[]	toArray(T[] a)	该方法返回一个包含 set 中所有元素的数组；返回数组的运行时类型是指定数组的类型

因为 Set 是继承了 Collection 接口的，所以所有 Collection 接口的方法都被继承了。因为 Set 中不允许出现重复的元素，因此如果试图将复制元素加到集合中时，add 方法将返回 false。Set 集合本身并没有定义任何附加的方法。

19.3.2 SortedSet 接口

SortedSet 继承自 Set 接口，所以该接口不但具有 Set 的所有功能，而且是一个 Sorted 类型的 Set。也就是说，实现该接口的类将按元素的天然顺序自动排序，不管插入的顺序是什么，其总会按照元素的天然顺序进行遍历。表 19-6 列出了该接口中的常用方法。

表 19-6 SortedSet 接口中的常用方法

方法名	功能
Object first()	该方法将返回 SortedSet 中第一个元素，也就是最小的元素
Object last()	该方法将返回 SortedSet 中最后一个元素，也就是最大的元素
SortedSet headSet(Object toElement)	参数 toElement 为一个指定的元素，该方法将返回一个包含所有小于指定元素并且不包含指定元素的 SortedSet
SortedSet tailSet(Object fromElement)	参数 fromElement 为一个指定的元素，该方法将返回一个包含所有大于指定元素并且包含指定元素的 SortedSet
SortedSet subSet(Object fromElement, Object toElement)	参数 fromElement 为指定的起始元素，参数 toElement 为指定的结束元素，该方法将返回一个包含从起始元素到结束元素中所有元素并且不包含结束元素的 SortedSet



说明：从表 19-6 中所列的方法可以看出，实现了该接口的集合当中的元素有固有的顺序。

19.3.3 TreeSet 类

TreeSet 类是实现了接口 SortedSet 的类，已知 SortedSet 提供了集合元素的顺序存储，其中元素保持升序排列。此类保证排序后的 set 按照升序排列元素，根据使用的构造方法不同，可能会按照元素的自然顺序进行排序。

TreeSet 类提供了 4 种构造方法，这些方法声明及使用描述如下所示。

- TreeSet() 方法：该构造器将构造一个空的 TreeSet 对象。
- TreeSet(Collection c) 方法：参数 c 为包含指定元素的 Collection。该构造器将构造一个以 c 中的元素为初始内容的 TreeSet 对象。
- TreeSet(Comparator c) 方法：参数 c 为指定的比较器，该构造器将构造具有指定比较器的空 TreeSet 对象。
- TreeSet(SortedSet s) 方法：参数 s 为包含指定元素的 SortedSet。该构造器将构造一个以 s 中的元素为初始内容的 TreeSet 对象。

【范例 19-7】示例代码 19-7 是一个使用 TreeSet 类的程序。

示例代码 19-7

```

01 import java.util.*;
02 public class J1He7
03 {
04     public static void main(String[] args)
05     {
06         //创建了 TreeSet 对象
07         TreeSet ts=new TreeSet();
08         //向 TreeSet 对象中依次添加了数字为 5、6、3、2、4 的字符串
09         ts.add(String.valueOf(5));
10         ts.add(String.valueOf(6));
11         ts.add(String.valueOf(3));
12         ts.add(String.valueOf(2));
13         ts.add(String.valueOf(4));
14         //移除了 HashSet 对象中内容为“5”的字符串
15         ts.remove(String.valueOf(5));
16         //将内容为“1”的字符串添加了进 HashSet 对象
17         ts.add(String.valueOf(1));
18         //打印输出 TreeSet 中的内容
19         System.out.print("TreeSet 操作后的结果为: ");
20         System.out.println(ts);
21     }
22 }
```

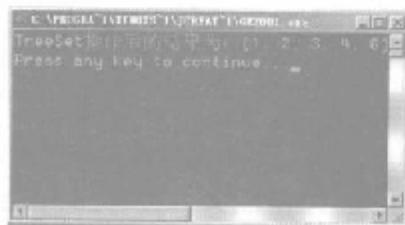


图 19-7 TreeSet 类

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-7 所示。

【代码解析】在本程序中，从第 9 行到第 13 行依次向 TreeSet 类中添加 5、6、3、2、4，然后将 5 这个字符串删除，再添加一个 1 字符串。在程序的第

20行要求显示 TreeSet 类中的所有元素。从运行结果中可以看出，元素的显示结果是按照自然的顺序来显示的，这个是和添加的顺序无关的。

TreeSet 中同样有很多方法，这些方法有些是实现 SortedSet 接口中的方法，也有自定义的方法。下面就来看一个使用这些方法的程序。

【范例 19-8】示例代码 19-8 是一个使用 TreeSet 类中方法的程序。

示例代码 19-8

```

01 import java.util.*;
02 public class JiHe8
03 {
04     public static void main(String[] args) throws Exception
05     {
06         TreeSet ts = new TreeSet(); //创建一个TreeSet 对象
07         for (int i = 5; i >= 0; i--)
08         {
09             ts.add(new Integer(i)); //循环加入 5 个整型对象
10         }
11         TreeSet anotherTs = (TreeSet) ts.clone(); //复制 TreeSet 对象
12         Iterator it = anotherTs.iterator();
13         while (it.hasNext())
14         {
15             System.out.println(it.next()); //循环输出 it 对象中的元素
16         }
17         System.out.println("创建 TreeSet 类");
18         TreeSet s = new TreeSet(new MyComparator()); //创建一个TreeSet 对象
19         s.add("aa"); //添加 4 个字符串元素
20         s.add("cc");
21         s.add("dd");
22         s.add("bb");
23         Iterator its = s.iterator(); //为对象 s 创建一个迭代器对象
24         while (its.hasNext())
25         {
26             System.out.println(its.next()); //遍历输出 s 中的对象
27         }
28         System.out.println("第一个元素为：" + s.first()); //输出 s 对象中的第一个
29         System.out.println("最后一个元素为：" + s.last()); //输出 s 对象中的最后一个
30         System.out.println("有几个元素：" + s.size()); //输出 s 对象中的元素数
31     }
32 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-8 所示。

【代码解析】在本程序中首先创建一个 TreeSet 对象。然后在第 11 行使用 clone 方法复制一个 TreeSet 对象，接下来循环显示元素。需要特别说明的是，在该程序的第 28 行和 29 行分别使用 first 方法和 last 方法来获取 TreeSet 对象中的第一个元素和最后一个元素。



图 19-8 TreeSet 类方法



19.3.4 HashSet 类

HashSet 类是 Set 接口最常用的实现之一，其既不是 Ordered 的也不是 Sorted 的，元素在其中存储不保证任何顺序。实际上，HashSet 存储对象引用时是按照哈希策略来实现的。另外，可以向 HashSet 中添加 null 值，但只能添加一次。表 19-7 列出了 HashSet 类的几个构造器。

表 19-7 HashSet类的几个构造器

构造器名	功 能
public HashSet()	该构造器将构造一个空的 HashSet 对象。该对象的初始容量为 16
public HashSet(int initialCapacity)	参数 initialCapacity 表示指定的初始容量，该构造器将构造一个具有指定容量的空 HashSet 对象
public HashSet(Collection c)	参数 c 为包含指定元素的 Collection，该构造器将构造一个以 c 中的元素为初始内容的 HashSet 对象

【范例 19-9】示例代码 19-9 是一个使用 HashSet 类的程序。

示例代码 19-9

```
import java.util.*;
public class JiHe9
01 {
02     public static void main(String[] args)
03     {
04         HashSet hs=new HashSet();           //创建 HashSet 对象
05         hs.add(String.valueOf(5));        //向其中添加 5 个元素
06         hs.add(String.valueOf(1));
07         hs.add(String.valueOf(3));
08         hs.add(String.valueOf(2));
09         hs.add(String.valueOf(4));
10         hs.remove(String.valueOf(5));    //删除 HashSet 对象中内容为“5”的字符串
11         hs.add(String.valueOf(1));       //再添加一个内容为 1 的字符串
12         hs.add(null);                 //将 null 值添加了进 HashSet 中
13         System.out.println("HashSet 中元素包括: ");
14         System.out.println(hs);
15     }
16 }
```

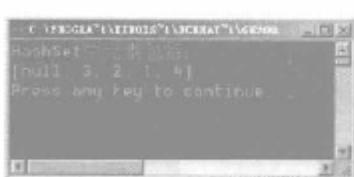


图 19-9 使用 HashSet 类

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-9 所示。

【代码解析】在本程序中，首先创建一个 HashSet 对象，然后按照不规则的方式输入 5 个元素，然后在程序的第 10 行，使用 remove 方法删除元素内容为“5”的元素，接下来在程序的第 11 行再添加一个“1”元素，在 12 行再向 HashSet 对象中添加一个 null 元素。从运行结果中可以看出，HashSet 中并不保证元素的存储顺序，既不是添加顺序，也不是元素自身的顺序。相同的元素在 HashSet 中只能有一次，同时 HashSet 中允许存放 null 值。



提示：任何对象都可以插入哈希表中，因为在 Java 中所有类都继承自根类 Object，而 Object 具有方法 hashCode，这个方法用于返回对象的哈希码。该哈希码由对象的内存地址派生而来。



19.4 映射

Map（映射）是一个存储关键字和值的关联，或者说是关键字/值对的集合。给定一个关键字，可以得到其相应的值。关键字和值都是对象。关键字必须是唯一的，但值是可以被复制的。而一个值对象可以是另一个 Map，依次类推，这样就可形成一个多级映射。对于键对象来说，像 Set 一样，一个 Map 容器中的键对象不允许重复，这是为了保持查找结果的一致性。

19.4.1 Map 接口

Map 也可以称之为键/值集合，因为在实现了该接口的集合中，元素都是成对出现的，一个称之为键，另一个称之为值。键和值都是对象，键对象用来在 Map 中唯一的标识一个值对象。键对象在 Map 中不能重复出现，就像 Set 中的元素不能重复一样。



注意：与 Collection 系列的集合一样，系统并不真正把对象放到 Map 中，Map 中存放的只是键和值对象的引用。

表 19-8 列出了 Map 接口中的常用方法。

表 19-8 Map 接口中的常用方法

方法名	功能
void clear()	从此映射中删除所有映射关系
boolean containsKey(Object key)	如果此映射包含指定键的映射关系，则返回 true
boolean containsValue(Object value)	如果此映射为指定值映射一个或多个键，则返回 true
Object get(Object key)	返回此映射中映射到指定键的值
boolean isEmpty()	该方法将测试 Map 中是否还存在映射关系，若存在则返回 false，否则返回 true
Object put(Object key, Object value)	将指定的键 key 与值 value 添加到 Map 中，并将其关联。如果之前有相同的 key 存在，则只将值 value 添加到 Map 中，并将其与之前相同的 key 关联。完成操作后将值返回
void putAll(Map t)	参数 t 为包含需要添加键值对的 Map，该方法将 t 中包含的元素添加进该方法所在的 Map
Object remove(Object key)	如果存在此键的映射关系，则将其从映射中删除
int size()	该方法将返回 Map 中键值对的个数



提示：通过上表可以看出，实现了 Map 接口的集合很注重键/值关系，若开发人员需要一个可以通过键搜索并查找值的实现，可以选用实现了 Map 接口的集合。



19.4.2 HashMap 类

HashMap 类是基于哈希表的 Map 接口的实现。该类提供所有可选的映射操作，并允许使用 null 值和 null 键。但是此类不保证映射的顺序。

HashMap 的实例有两个参数影响其性能，分别为初始容量和加载因子。容量是哈希表中桶的数量，初始容量只是哈希表在创建时的容量。加载因子是哈希表在其容量自动增加之前可以达到多满的一种尺度。当哈希表中的条目数超出了加载因子与当前容量的乘积时，通过调用 rehash 方法将容量翻倍。HashMap 类声明如下所示。

```
public class HashMap<K,V> extends AbstractMap<K,V> implements Map<K,V>, Cloneable, Serializable
```

HashMap 最常用的构造器如表 19-9 所示。

表 19-9 HashMap类的构造器

构造器名	功 能
public HashMap()	该构造器将构造一个空的 HashMap 对象，其初始容量为 16
public HashMap(Map m)	参数 m 为包含指定键值对的 Map。该构造器将构造一个以 m 中的键值对为初始内容的 HashMap 对象

HashMap 类可为基本操作（get 和 put）提供稳定的性能。迭代该集合视图所需的时间与 HashMap 实例的“容量”（桶的数量）及其大小（键-值映射关系数）之和成比例。所以，如果迭代性能很重要，则不要将初始容量设置得太高，或将加载因子设置得太低。

【范例 19-10】示例代码 19-10 是一个使用 HashMap 类的程序。

示例代码 19-10

```
00 import java.util.*;
01 public class JiHe10
02 {
03     public static void main(String[] args)
04     {
05         //创建了 HashMap 对象
06         HashMap hm=new HashMap();
07         //向 HashMap 对象中添加内容不同的键值对
08         hm.put(1,"A");
09         hm.put(3,"B");
10         hm.put(4,"C");
11         hm.put(2,"D");
12         hm.put(5,"E");
13         System.out.print("添加元素后的结果为: ");
14         System.out.println(hm);
15         //移除了 HashMap 对象中键为 97001 的值
16         hm.remove(3);
17         //替换键 97002 对应的值
18         hm.put(4,"F");
19         //打印输出 HashMap 中的内容
20         System.out.print("删除和替换元素后结果为: ");
21         System.out.println(hm);
22         //取出指定键对应的值
23         Object o=hm.get(2); //使用了自动打包功能
24         String s=(String)o;
```

```

25         System.out.println("键 2 对应的值为: " + s);
26     }
27 }

```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-10 所示。

【代码解析】在本程序中，首先在第 6 行创建了一个 HashMap 对象，然后使用 put 方法向该对象中添加元素，在向 HashMap 中添加元素时，不但要将元素添加，还要为每一个元素设置一个 Hash 码。在 HashMap 对象中同样能够使用 remove 方法删除元素和使用 put 方法为指定的 Hash 码来重新设置元素。在 HashMap 对象中还可以以 Hash 码为 get 方法的参数来获取指定的元素。



注意：Hash 码不仅可以为数字，同样它也可以为字符串，它们的操作都是一样的。

示例代码 10-11 就是一个不同 Hash 码形式的程序。

【范例 19-11】示例代码 19-11 是一个不同 Hash 码形式的程序。

示例代码 19-11

```

01 import java.util.*;
02 import java.util.Map.Entry;
03 public class JiWei1
04 {
05     public static void main(String[] args)
06     {
07         // 创建一个 HashMap 类对象，参数与返回值都必须为 String 类型
08         HashMap<String, String> mapHash = new HashMap<String, String>();
09         mapHash.put("one", "A"); // 使用 put 方法添加键值对
10     HashMap 对象
11         mapHash.put("two", "B");
12         mapHash.put("three", "C");
13         mapHash.put("four", "D");
14         mapHash.put("five", "E");
15         mapHash.put("six", "F");
16         mapHash.put("seven", "G");
17         String query = "six";
18         System.out.println("键值为" + query);
19         String resultString = (String) mapHash.get(query); // 使用 get 方法通过
19                                         // 键获取值的方法
20         System.out.println("对应的值为: " + resultString);
21         Set<Entry<String, String>> hsm = mapHash.entrySet(); // 通过 entrySet
21                                         // 方法获取 Set 视图
22         Iterator<Entry<String, String>> i = hsm.iterator();
23         for (; i.hasNext();)
24         {
25             Entry en = i.next();
26             System.out.println(" 键为: " + en.getKey() + " 对应的值为: " +
26             en.getValue());
27         }
28     }

```

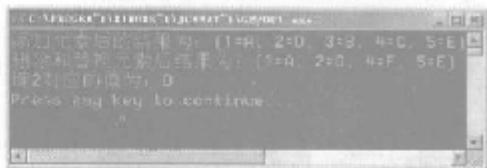


图 19-10 使用 HashMap 类



```
29      }
30  }
```

The screenshot shows a Java application window titled "21 天学通 Java" with the following content:

```

键值为 SIX
键值的键为: F
键值: two 对应的值为: B
键值: seven 对应的值为: G
键值: five 对应的值为: E
键值: one 对应的值为: A
键值: three 对应的值为: C
键值: four 对应的值为: D
键值: six 对应的值为: F
Press any key to continue ...
  
```

图 19-11 字符串为 Hash 码

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-11 所示。

【代码解析】 在示例代码 19-10 中使用的 Hash 码为数字的形式，而本程序中使用的 Hash 码是字符串的形式。本程序和示例代码 19-10 很相似。在本程序的最后使用 hasNext 方法来循环显示所有的元素，并同时将键和对应的值都显示出来。

19.4.3 TreeMap 类

TreeMap 类是 SortedMap 接口的基于红黑树的实现(红黑树是一种特定类型的二叉树，是一种自平衡二叉查找树，读者可以查阅相关资料)。此类保证了映射按照升序顺序排列关键字，根据使用的构造方法不同，可能会按照键的类的自然顺序进行排序(参见 Comparable)，或者按照创建时所提供的比较器进行排序。TreeMap 类的声明如下所示。

```
public class TreeMap<K,V> extends AbstractMap<K,V> implements SortedMap<K,V>, Cloneable, Serializable
```

TreeMap 类提供了 4 个构造方法，具体构造方法如表 19-10 所示。

表 19-10 TreeMap 类的几个构造器

构造器名	功 能
public TreeMap()	该构造器将构造一个空的 TreeMap 对象
public TreeMap(SortedMap s)	参数 s 为包含指定键值对的 SortedMap。该构造器将构造一个以 s 中的键值对为初始内容的 TreeMap 对象
public TreeMap(Map c)	参数 c 为包含指定键值对的 Map。该构造器将构造一个以 c 中的元素为初始内容的 TreeMap 对象
TreeMap(Comparator comparator)	Comparator 为指定的比较器，与 TreeSet 相同，如果想指定键的排序规则，则可以使用此构造器

public TreeMap(SortedMap<K,? extends V> m)方法构造一个新的映射，包含的映射关系与给定的 SortedMap 相同，该映射按照相同的排序方式进行排序。此方法以线性的时间运行。

【范例 19-12】示例代码 19-12 是一个使用 TreeMap 类的程序。

示例代码 19-12

```
01 import java.util.*;
02 public class JiHe12
03 {
04     public static void main(String[] args)
05     {
06         // 创建一个 TreeMap 类对象
07         TreeMap<String, String> tm = new TreeMap<String, String>();
08         tm.put("1", "A");           // 使用 put 方法添加键值对 TreeMap 对象
09         tm.put("2", "B");
```

```

11     tm.put("3", "C");
12     tm.put("4", "D");
13     tm.put("5", "E");
14     tm.put("6", "F");
15     System.out.println("所有的元素的值为: " + tm); // 输出 tm 对象中的所有元素
16     Iterator iter = tm.keySet().iterator(); // 为 tm 对象中的键值添加一个
17     // 迭代器
18     for (; iter.hasNext();)
19     {
20         System.out.println("元素值为: " + tm.get(iter.next()));
21     }
22 }

```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-12 所示。

【代码解析】 在示例代码 19-12 中，同样首先使用 put 方法将一些基本的元素添加到 TreeMap 对象中。同样在最后使用 for 循环来显示所有的元素。



图 19-12 使用 TreeMap 类

19.5 综合练习

1. 如何使用集合中的 sort 方法对集合中的元素进行排序？

【提示】

```

01 import java.util.*;
02 public class LianXii
03 {
04     public static void main(String[] args)
05     {
06         ArrayList al=new ArrayList(); // 创建 ArrayList 对象
07         // 随机创建 10 个整数，并添加到集合中
08         for(int i=0;i<10;i++)
09         {
10             al.add(Integer.valueOf((int)(Math.random()*100)));
11         }
12         // 打印初始化后 ArrayList 中的元素
13         System.out.println("排序前 ArrayList 中的元素");
14         System.out.println(al);
15         Collections.sort(al); // 使用 sort 方法对元素进行排序
16         // 打印排序后的 ArrayList 对象
17         System.out.println("排序后 ArrayList 中的元素");
18         System.out.println(al);
19     }
20 }

```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-13 所示。

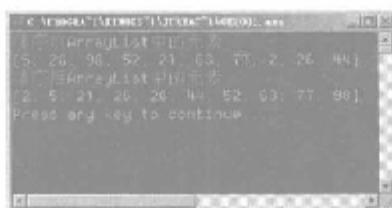


图 19-13 对元素进行排序

2. 如何使用 binarySearch 方法搜索集合中的元素?

【提示】在使用 binarySearch 方法搜索元素之前，首先要对元素进行排序。

```
import java.util.*;
public class LianXi2
{
    public static void main(String[] args)
    {
        ArrayList al=new ArrayList();           //创建ArrayList 对象
        //随机创建 10 个整数，并添加到集合中
        for(int i=0;i<100;i++)
        {
            al.add(Integer.valueOf((int)(Math.random()*100)));
        }
        Collections.sort(al);                  //使用 sort 方法对元素进行排序
        //使用 binarySearch 方法查找指定元素。
        int index=Collections.binarySearch(al, Integer.valueOf(15));
        if(index<0)
        {
            System.out.println("查找失败");
        }
        else
        {
            System.out.println("索引值为: "+index);
        }
    }
}
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 19-14 所示。

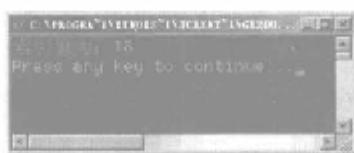


图 19-14 对元素进行搜索



19.6 小结

本章主要对 Java 中的集合框架进行了详细的讲解。集合框架主要包括列表、集合和映射，同时在各个方面中还包括一些具体的接口和类。如果读者想了解更多的关于集合框架



的知识，可以参考电子工业出版社出版的《Java 程序设计经典教程：融合上机操作实例》一书进行学习。

19.7 习题

一、填空题

1. 集合框架的主要优点在于提高了_____、_____和_____。
2. 在这些接口中_____接口是层次结构中的根接口，在该接口中具有开发中经常用到的_____接口、_____接口和_____接口。
3. Set 接口与 List 接口最大的区别在于：_____。
4. 集合是某一类对象的通称呼，这类对象代表以某种方式组合到一起的一组对象；它是将多个元素组合为一个单元的对象，用于_____，_____，_____和_____数据。
5. List 比 Collection 多了 10 个方法，这些方法可以分为_____方法、_____方法、_____方法和_____、_____方法。
6. java.util.List 的几种实现中，有三种最为常用的实现类，这三个类分别是_____类、_____类和_____类。
7. Map 也可以称之为_____集合，因为在实现了该接口的集合中，元素都是成对出现的。
8. Set 集合是一种_____的 Collection，即任意的两个元素 e1 和 e2 比较，结果都不相等。
9. HashMap 的实例有两个参数影响其性能：_____和_____。

二、选择题

1. 选择下面程序的运行结果（_____）。

```

01 import java.util.*;
02 public class Hello
03 {
04     public static void main(String args[])
05     {
06         TreeSet ts=new TreeSet();
07         ts.add("a");
08         ts.add("c");
09         ts.add("d");
10         ts.add("b");
11         ts.add("c");
12         Iterator it =new ts.iterator();
13         while(it.hasNext())
14         {
15             System.out.println(it.next());
16         }
17     }
18 }
```

- A. abcd B. acdb C. acdbc D. 编译发生错误
2. 在 Map 接口中表示从此映射中删除所有映射关系的方法是（_____）。
 - A. clear()方法
 - B. containsKey 方法



- C. isEmpty()方法 D. size()方法
3. 返回集合的大小的方法是();
A. clear()方法 B. containsKey 方法
C. isEmpty()方法 D. size()方法
4. 返回 SortedSet 接口中第一个元素的方法是();
A. first()方法 B. last()方法 C. headSet 方法 D. subSet 方法

三、简答题

1. 什么是集合框架，集合框架是怎么工作的？
2. 集合框架的最顶层接口是什么接口，它和 List 接口、Set 接口和 Map 接口是什么关系？

四、编程题

1. 编写简单的学生系统，在该系统中使用集合框架来表示学生。

第 20 章 网 络 编 程

随着 Internet 的逐渐普及，越来越多的软件项目应用涉及到网络。而 Java 似乎专门为网络设计的，它提供了很多方法供程序员使用，使能够很方便地开发网络应用软件。本章，将介绍 Java 网络编程的基础知识，以及网络编程的特点和方法。通过本章的学习，读者应该实现如下几个目标。

- 了解什么是协议，有哪些协议。
- 了解网络编程的模型。
- 熟练掌握使用 Socket 进行网络编程。



20.1 网络编程基础

简单来看，网络编程的目标就是计算机之间相互进行数据通信。Java SDK 提供了一系列 API 来完成这些工作，例如 Socket。对于程序员来说，这些 API 被存放在 java.net 包里面，因此只要导入这个包就可以进行网络编程。

20.1.1 TCP/IP 协议

现在的 Internet 或 Intranet 大部分都是使用 TCP/IP 协议进行网络通信的，实际上 TCP/IP 协议是一组以 TCP 与 IP 协议为基础的相关协议的集合。



注意：该协议并不完全符合 OSI 的七层参考模型，而是采用的 4 层结构。

IP 协议是 TCP/IP 协议族的核心，也是互连网络层中最重要的协议。其接收由更低层发来的数据包，并将该数据包发送到更高层，即 TCP 或 UDP 层；此外 IP 层也可以将从 TCP 或 UDP 层接收来的数据包传送到更低层。IP 是面向无连接的数据报传送，所以 IP 协议将报文传送到目的主机后，无论传送正确与否都不进行检验、不会送确认以及不保证分组的正确顺序。

TCP 协议位于传输层，其提供面向有连接的数据包传送服务，保证数据包能够被正确传送与接收，包括内容的校验与包的顺序，损坏的包可以被重传。要注意的是，由于提供的是有保证的数据传送服务，因此传送效率要比没有保证的服务低，一般适合工作在广域网中，对网络状况非常好的局域网不是很合算。当然，是否采用 TCP 也取决于具体的应用需求。

20.1.2 网络编程模型

对于网络编程来说，目前主要有两种编程模型，分别是 C/S 结构和 B/S 结构。C/S 结构是指客户机/服务器结构。所谓客户机/服务器结构，指的是在客户端需要安装客户端软



件，由客户端软件负责与服务器端的数据通信，将任务合理分配到客户端和服务器端来实现，降低了网络的负载开销。

B/S 结构是指浏览器/服务器结构。客户端只需要安装有网页浏览器，不需要安装客户端软件，大部分逻辑事务处理在服务器端完成，客户端浏览器只完成少量的事务处理，减少了客户端的计算机负载，减轻了系统维护与升级的成本和工作量。在 Java 这样的跨平台语言出现之后，B/S 架构管理软件更是方便、快捷、高效。

20.1.3 网络传输协议

在前面已经讲解了 TCP 协议的知识，网络传输协议除了 TCP 协议外，还有 UDP 协议。UDP（User Datagram Protocol）是用户数据报协议的英文缩写，UDP 是不可靠的无连接数据报服务，它不需要像 TCP 那样建立连接，它有点类似电报的形式，直接发送，而不管接收端是否收到了数据，所以效率高，发送时速度快，但是不可靠。

UDP 是一种无连接的协议，每个数据包都是一个独立的信息，包括完整的源地址或目的地址。UDP 数据包在网络上传往目的地的路径是未知的，因此能否到达目的地，到达目的地的时间以及内容是否正确都是不能确定的。使用 UDP 无须要建立发送方和接收方的连接。而 TCP 协议，由于它是一个面向连接的协议，在 Socket 之间进行数据传输之前首先要建立连接，所以在利用 TCP 协议传输的过程中增加了连接建立的时间。

使用 UDP 传输数据时是有大小限制的，每个被传输的数据报必须限定在 64KB 之内。而 TCP 没有这方面的限制，一旦连接建立起来，双方的 Socket 就可以按统一的格式传输大量的数据。再次需要强调的是，UDP 是一个不可靠的协议，发送方所发送的数据报并不一定以相同的次序到达接收方。而 TCP 是一个可靠的协议，它确保接收方完全正确地获取发送方所发送的全部数据。



提示：网络编程的目的是指通过网络协议与其他计算机进行通信。网络编程中主要有两个的问题：一是如何准确地定位网络上一台或多台主机，二是如何找到主机后可靠高效地进行数据传输。

20.1.4 端口和套接字

端口被规定为一个在 0~65535 之间的整数。HTTP 服务一般使用 80 端口，FTP 使用的是 21 端口，那么客户必须通过 80 端口才能连接到服务器的 HTTP 服务，而通过 21 端口，才能连接到服务器的 Ftp 服务器上。

在所有的端口中 1~1023 之间的已经被系统所占用了，因此在定义自己的端口时，不能使用这一段的端口号，而应该使用 1024~65535 之间的任意端口号，以免发生端口冲突。

网络程序中的套接字用来将应用程序与端口连接起来，套接字是一个软件实现，也是一个假想的装置。在 Java 中，将套接字抽象化为类，所以程序只需创建 Socket 类的对象，就可以使用套接字。那么 Java 是如何实现数据传递的呢？答案是使用 Socket 的流对象进行数据传输，Sokcet 类中有输入和输出流。使用 Socket 进行的通信都称为 Socket 通信。将编写的 Socket 类，用在 Socket 通信程序中，这就称为 Socket 网络程序设计。



20.2 基于 TCP/IP 协议的网络编程

通过前面一节的介绍，读者已经对网络编程基础的知识有了一个简单的了解。本节将继续向读者介绍如何利用 Java 进行基于 TCP Socket（套接字）连接的网络应用的开发。TCP/IP 是用于计算机通信的一组协议，它包括 TCP、UDP 等许多协议，这些协议一起称为 TCP/IP 协议。

20.2.1 Socket 套接字

Socket 套接字是基于 TCP/IP 协议的编程接口，用于描述 IP 地址和端口，是一个通信链的句柄。应用程序通常通过 Socket 套接字向网络发出请求或应答网络请求。

Socket 有两种主要的操作方式，包括面向连接的和无连接的。面向连接的 Socket 操作就像人们打电话，必须先拨号码，建立一个连接，然后再对话。数据包在到达接收端时的顺序与它们出发时的顺序一样，就像一个人在电话中对另一个人说话一样，每一个字到达另一端的时候与它出发时的顺序一样。

无连接的 Socket 操作就像是邮递员送信，邮递员只负责把信送出，至于何时发出，最后能不能到达收信人手中不能保证，无连接的 Socket 操作也一样，负责发出，但不保证数据包的传输质量，数据包到达目的地的顺序可能与出发时的顺序不一样。

20.2.2 ServerSocket 类

Java 中的网络编程是通过 ServerSocket 类和 Socket 类结合使用来完成的。下面首先讲解一下 ServerSocket 类，ServerSocket 类是应用在服务器端的类。在服务器端，由 ServerSocket 类负责实现服务器套接字。ServerSocket 类位于 java.net 包中。由 ServerSocket 对象监听指定的端口，端口可以任意指定。但是要注意 1024 以下的端口通常属于系统保留端口，因此不可以随便使用，应该使用大于 1024 的端口号，开始监听后，服务器就等待客户端连接请求，客户端连接后，会话开始；在完成会话后，关闭连接。

ServerSocket 类对象工作在服务器端，用来监听指定的端口并接收客户端的连接请求，其一共提供了 4 个构造器，如表 20-1 所示。

表 20-1 ServerSocket 类的构造器

构造器名	功 能
public ServerSocket() throws IOException	创建一个不绑定到任何端口的空 ServerSocket 对象
public ServerSocket(int port) throws IOException	创建一个绑定到特定端口 port 上的 ServerSocket 对象，若端口 port 的值为 0，则表示使用任何空闲端口
public ServerSocket(int port,int backlog) throws IOException	创建一个绑定到特定端口上的 ServerSocket 对象，参数 port 为指定的端口，参数 backlog 表示请求等待队列的最大长度
public ServerSocket(int port, int backlog, InetAddress bindAddr) throws IOException	创建一个绑定到本机指定 IP 地址指定端口上的 ServerSocket 对象，参数 port 为指定的端口，参数 backlog 表示请求等待队列的最大长度，参数 bindAddr 表示要绑定到的本机 IP 地址。请读者注意，如果本机有几个不同的 IP 地址可以采用此构造器指定绑定到其中的哪一个



注意:同一台主机上的同一端口号只能分配给一个特定的 ServerSocket 对象，不能两个 ServerSocket 对象监听同一个端口。端口号的理论范围为 0~65535，但前 1024 个中的大部分已经分配给了特定的应用协议，所以不能选用。

【范例 20-1】示例代码 20-1 是一个使用 ServerSocket 类的程序。

示例代码 20-1

```

01 import java.net.*;
02 public class WangLucl
03 {
04     public static void main(String args[])
05     {
06         try
07         {
08             ServerSocket ss=new ServerSocket(9876);
09         }
10         catch(Exception e)
11         {
12         }
13     }
14 }
```

【代码解析】示例代码 20-1 是一个应用在服务器端的程序，在该程序中的第 8 行，使用 ServerSocket 类创建了一个 ServerSocket 对象，为客户端开发的一个 9876 端口。需要注意的是，创建 ServerSocket 对象是可以发生异常的，必须要进行异常处理。

在 ServerSocket 类中有几个非常常用的方法。首先要讲的就是 accept 方法，使用该方法接收客户端的连接请求，并将与客户端的连接封装成一个 Socket 对象返回。



注意:此方法为阻塞方法，在没有接收到任何连接请求前调用此方法的线程将一直阻塞等待，直到接收到连接请求后此方法才返回，调用此方法的线程才继续运行。

除了 accept 方法外还有一个 close 方法来关闭 ServerSocket 对象。使用 getLocalPort 方法来获取设置的端口号。

20.2.3 Socket 类

Socket 类表示套接字。使用 Socket 时，需要指定待连接服务器的 IP 地址及端口号。客户机创建了 Socket 对象后，将马上向指定的 IP 地址及端口发起请求且尝试连接。于是，服务器套接字就会创建新的套接字对象，使其与客户端套接字连接起来。一旦服务器套接字与客户端套接字成功连接后，就可以获取套接字的输入/输出流，彼此进行数据交换。

Socket 类一共有 9 个构造器，表 20-2 列出了其中常用的两个。

表 20-2 Socket类的常用构造器

构造器名	功 能
public Socket(String host,int port) throws UnknownHostException, IOException	创建一个连接到指定主机指定端口的 Socket 对象，参数 host 为指定的主机名或 IP 地址字符串，参数 port 为指定的端口

续表

构造器名	功 能
public Socket(InetAddress address, int port) throws IOException	创建一个连接到指定主机指定端口的 Socket 对象, 参数 address 给出要连接主机的 IP 地址, 参数 port 为指定的端口

【范例 20-2】示例代码 20-2 是一个使用 Socket 类的程序。

示例代码 20-2

```

01 import java.net.*;
02 public class WangLuo2
03 {
04     public static void main(String args[])
05     {
06         try
07         {
08             Socket s=new Socket("192.168.1.119",9875);
09             s.close();
10         }
11         catch(Exception e)
12         {
13         }
14     }
15 }
```

【代码解析】在示例代码 20-2 中, 在程序的第 8 行使用 Socket 类来创建了一个 Socket 对象。这里使用的是 Socket 类的第一种构造器, 第一个参数表示要连接计算机的 IP 或主机名, 第二个参数表示要连接的端口。该端口必须是连接计算机中已经打开的端口, 也就是示例代码 20-1 中使用 ServerSocket 类的参数。

Socket 类中同样具有一些方法。其中 getPort 方法和 getLocalPort 方法分别是获取连接的远程端口和使用的本地端口。getInputStream 方法和 getOutputStream 方法分别是获取 Socket 对象的输入流和输出流, 这两个方法是经常被使用到的。



注意: close 方法虽然是很简单的, 表示关闭 Socket 对象。但是该方法在程序中是必须有的, 这是一个非常好的网络编程习惯。

20.2.4 网络编程 C/S 架构实例

C/S 架构的网络编程程序是由服务器端和客户端所组成的, 在开发时一定要先开发服务器端的程序, 再来开发客户端的程序。

【范例 20-3】示例代码 20-3 是一个运行在服务器端的程序。

示例代码 20-3

```

01 import java.io.*;
02 import java.net.*;
03 import java.util.*;
04 public class WangLuo3
05 {
06     public static void main(String[] args)
07     {
08         int count=0;//声明用来计数的 int 局部变量
09         try
```



```
10
11      //创建绑定到 9876 端口的 ServerSocket 对象
12      ServerSocket server=new ServerSocket(9876);
13      System.out.println("服务器对 9876 端口正在进行监听");
14      //服务器循环接收客户端的请求，为不同的客户端提供服务
15      while(true)
16      {
17          //接收客户端的连接请求，若有连接请求返回连接对应的 Socket 对象
18          Socket sc=server.accept();
19          //获取当前连接的输入流，并使用处理流进行封装
20          DataInputStream din=new DataInputStream(sc.getInputStream());
21          //获取当前连接的输出流，并使用处理流进行封装
22          DataOutputStream dout=new DataOutputStream(sc.getOutputStream());
23          //打印客户端的信息
24          System.out.println("这是第"+(++count)+"个客户访问");
25          System.out.println("客户端 IP 地址：" +sc.getInetAddress());
26          System.out.println("本地端口号：" +sc.getLocalPort());
27          System.out.println("客户端信息：" +din.readUTF());
28          //向客户端发送回应信息
29          dout.writeUTF("服务器的时间为：" +(new Date())+"。");
30          //关闭流
31          din.close();
32          dout.close();
33          //关闭此 Socket 连接
34          sc.close();
35      }
36  }
37  catch(Exception e)
38  {
39      e.printStackTrace();
40  }
41 }
42 }
```



图 20-1 启动服务器

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 20-1 所示。

【代码解析】图 20-1 是运行在服务器端的网络程序的第一次运行结果，只出现该结果是因为还没有客户端程序调用该程序。在该程序的第 12 行首先使用 ServerSocket 类创建了一个对象来对 9876 端口进行监听。当有客户端程序访问该程序时，就执行 while 循环，从而让服务器获取客户端的信息，并从服务器端向客户端发送当前时间信息。

开发完服务器端程序后，就需要继续来开发运行在客户端的程序。在客户端的程序需要使用 Socket 类来进行操作。

【范例 20-4】示例代码 20-4 是一个运行在客户端的程序。

示例代码 20-4

```
01 import java.io.*;
02 import java.net.*;
03 public class WangGuo4
04 {
05     public static void main(String[] args)
```

```

06     {
07         try
08         {
09             //创建连接到服务器的 Socket 对象
10             Socket sc=new Socket("192.168.1.119",9876);
11             //获取当前连接的输入流，并使用处理流进行封装
12             DataInputStream din=new DataInputStream(sc.getInputStream());
13             //获取当前连接的输出流，并使用处理流进行封装
14             DataOutputStream      dout=new      DataOutputStream(sc.getOutputStream());
15             dout.writeUTF("Hello");
16             System.out.println(din.readUTF());
17             din.close();
18             dout.close();
19             sc.close();
20         }
21         catch(Exception e)
22         {
23             e.printStackTrace();
24         }
25     }
26 }
27
28
29 }
30 }
```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 20-2 所示。

运行该程序后，服务器端程序的运行结果也会发生变化，变化后的运行结果如图 20-3 所示。

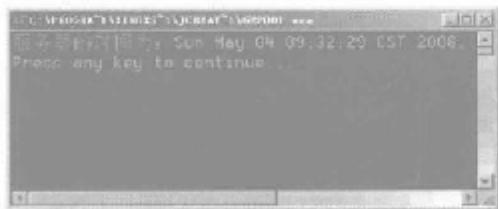


图 20-2 客户端程序



图 20-3 第一个客户

【代码解析】 在该客户端程序中首先使用流来获取服务器端向客户端发送的信息，该信息是指服务器端的时间。从运行结果中也可以看到这一点。客户端访问服务器端程序后，服务器端运行结果也会发生变化，出现图 20-3 的运行结果。从运行结果中可以看出显示客户端的 IP 地址，本地端口号和客户端向服务器端发送的信息。



注意：在进行网络编程时，有一点是需要特别注意的。服务器端程序运行后，运行结果是不可以关闭的，这样客户端才会访问到该服务器端程序。

如果一个服务器端只能被一个客户端程序访问，这显然是不合理的。如果客户端对服务器端进行第二次访问，就会出现如图 20-4 的运行结果。

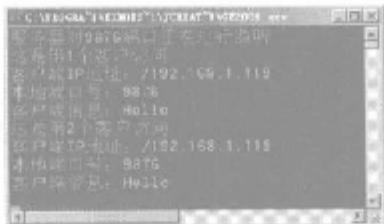


图 20-4 两个客户访问

从运行结果中可以看出，当有多个客户方法该服务器程序时将同时显示所有用户的信息。这里采用的是使用一个客户端程序进行多次访问，从而出现两次信息相同的情况。



20.3 综合练习

1. 开发一个局域网聊天程序。

【提示】这里给出一个简单的程序，读者可以自己扩展。首先是开发服务器端程序，代码如下所示。

```
01 import java.io.*;
02 import java.net.*;
03 import java.util.*;
04 public class LiamXiao
05 {
06     public static void main(String[] args)
07     {
08         int count=0;//声明用来计数的 int 局部变量
09         try
10         {
11             //创建绑定到 9876 端口的 ServerSocket 对象
12             ServerSocket server=new ServerSocket(9876);
13             System.out.println("服务器对 9876 端口正在进行监听");
14             //服务器循环接收客户端的请求，为不同的客户端提供服务
15             while(true)
16             {
17                 //接收客户端的连接请求，若有连接请求返回连接对应的 Socket 对象
18                 Socket sc=server.accept();
19                 //获取当前连接的输入流，并使用处理流进行封装
20                 DataInputStream din=new DataInputStream(sc.getInputStream());
21                 //获取当前连接的输出流，并使用处理流进行封装
22                 DataOutputStream dout=new DataOutputStream(sc.getOutputStream());
23                 System.out.println("欢迎你的访问");
24                 //发送客户端的信息
25                 dout.writeUTF("你好");
26                 //读取服务器的返回消息并打印
27                 System.out.println(din.readUTF());
28                 //关闭流
29                 din.close();
30                 dout.close();
31                 //关闭此 Socket 连接
32                 sc.close();
33             }
34 }
```

```

34     }
35     catch(Exception e)
36     {
37         e.printStackTrace();
38     }
39 }
40 }
```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 20-5 所示。

接下来开发客户端程序，代码如下所示。

```

01 import java.io.*;
02 import java.net.*;
03 public class Lianxi2
04 {
05     public static void main(String[] args)
06     {
07         try
08         {
09             //创建连接到服务器的 Socket 对象
10             Socket sc=new Socket("192.168.1.119",9076);
11             //获取当前连接的输入流，并使用处理流进行封装
12             DataInputStream din=new DataInputStream(sc.getInputStream());
13             //获取当前连接的输出流，并使用处理流进行封装
14             DataOutputStream dout=new DataOutputStream(sc.getOutputStream());
15             //向服务器发送消息
16             dout.writeUTF("你好");
17             //读取服务器的返回消息并打印
18             System.out.println(din.readUTF());
19             //关闭流
20             din.close();
21             dout.close();
22             //关闭此 Socket 连接
23             sc.close();
24         }
25         catch(Exception e)
26         {
27             e.printStackTrace();
28         }
29     }
30 }
```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 20-6 所示。

客户端向服务器端发送信息后，服务器端的运行结果也会发生变化，运行结果如图 20-7 所示。

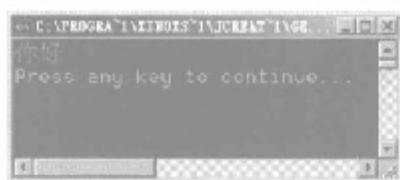


图 20-6 客户端

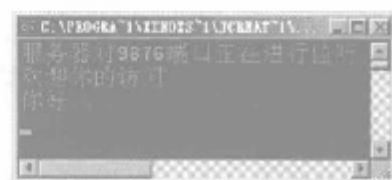


图 20-7 变化后



20.4 小结

在本章中对 Java 中如何进行网络编程进行了详细的讲解。首先讲解了一些网络编程的基础知识，包括 TCP/IP 协议、网络编程模型等。在后面又对基于 TCP/IP 协议的网络编程进行了详细的讲解，这里主要是使用 Socket 套接字来进行连接。该章介绍网络编程的内容并不是全面的，读者如果想了解更多的关于网络编程的内容，可以参考电子工业出版社出版的《Java 优化编程（第 2 版）》一书来进行更详细的学习。



20.5 习题

一、填空题

1. TCP 协议位于_____，其提供面向有连接的数据包传送服务，保证数据包能够被正确传送与接收，包括内容的校验与包的顺序，损坏的包可以被重传。
2. 对于网络编程来说，目前主要有两种编程模型，分别是_____和_____。
3. UDP 是一种无连接的协议，每个数据报都是一个独立的_____，包括完整的_____或_____。
4. TCP/IP 是用于计算机通信的一组协议，它包括_____、_____等许多协议，这些协议一起称为 TCP/IP 协议。
5. _____是基于 TCP/IP 协议的编程接口，用于描述 IP 地址和端口，是一个通信链的句柄。
6. 网络编程的目的是指_____。
7. Java 中的网络编程是通过_____类和_____类结合使用来完成的。

二、选择题

1. 端口号的理论范围为（ ）。
A. 1~1023 B. 1~1024 C. 1~65535 D. 1024~65535
2. _____是 TCP/IP 协议族的核心，也是互连网络层中最重要的协议（ ）。
A. IP 协议 B. TCP 协议
C. UDP 协议 D. HTTP 协议
3. _____是指浏览器/服务器结构（ ）。
A. A/S 结构 B. B/S 结构
C. C/S 结构 D. D/S 结构

三、简答题

1. 说明网络编程中的两种模型，并说出两种模型的不同点。
2. 简述应用 Socket 套接字进行网络编程的步骤。

四、编程题

1. 使用本章学习的内容，编写一个局域网聊天工具。

第 21 章 学生管理系统

学习完本书以后，读者可能会问学习 Java 有什么用。本章将使用一个综合案例来讲解如何使用 Java 进行实际开发。本章将开发一个读者最熟悉的学生管理系统，该系统在每一个学校中都有，也是非常容易开发的。学习完本章后读者应该实现如下几个目标。

- 掌握实际开发的步骤。
- 能够熟练开发和学生管理系统相类似的系统。
- 掌握 Java 中的界面开发。
- 掌握 Java 中如何连接数据库。



21.1 系统设计

首先确定学生管理系统的用户。学生管理系统的用户基本分为两类，分别是老师和学生。不管是哪种用户都必须经过登录才能进入学生管理系统，所以该系统必须有一个登录界面，并且在该界面中能够让用户选择用户是老师还是学生。该系统是不会对外开放的，所以也不存在注册界面。

因为用户分为两种，所以每一种用户进行操作的界面应该是不同的。首先是学生界面，在其中应该只有查询成绩和个人信息查询和插入。在本章中就来学习如何进行学生界面开发。

除了学生界面外，还要有一个老师界面。老师在老师界面中可以对学生信息进行管理，包括查询、修改和删除。同样也可以对学生的成绩进行管理，包括查询和插入，由于输入错误还要能够对学生的成绩进行修改，由于学生作弊还能够将学生的成绩进行删除。该系统的流程图如图 21-1 所示。



图 21-1 学生管理系统



21.2 数据库设计

本节将分析本系统中需要的数据库支持。首先数据库中应该有老师和学生这两个表，表中应该最少有用户名和密码两项，使用表中的这两项就可以进行登录。在学生表中还应该具有一些和学籍相关的信息，包括年龄、班级等内容，这样就可以在系统中对学生信息进行操作。

除此之外还需要一个成绩表，通过该表老师可以对学生的成绩进行查询、插入、修改和删除。学生也可以通过该表对自己的成绩进行查询。



21.3 登录界面开发

不管是老师和学生，都是从登录界面进入学生管理系统的。在登录界面中，应该先让用户选择自己的身份，然后系统将根据用户的选择来判断用户的身份并进行查询不同的数据库。这里为了让读者更容易理解登录界面，并没有连接数据库。读者可以将数据库的操作添加到程序中。

21.3.1 界面设计

这里的登录界面是采用最简单的形式，只需要用户输入用户名和密码就可以登录。并且在界面中定义了一个下拉列表让用户选择自己的身份。该界面的基本形式如图 21-2 所示。

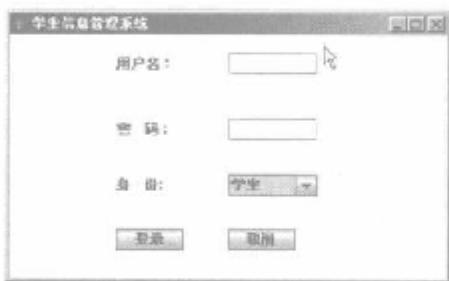


图 21-2 登录界面

21.3.2 程序开发

对界面设计好基本形式后，就可以进行程序开发。首先要定义两个标签和两个文本框，分别来表示用户名和密码，并且还需要定义一个下拉列表让用户来进行身份选择，其中选项包括“学生”和“老师”。在程序的最后还定义了两个按钮，从而让用户输入用户名和密码后进行登录。

【范例 21-1】示例代码 21-1 是用户登录界面程序。

示例代码 21-1

```

01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import java.util.*;
05 import javax.swing.event.*;
06 public class DengLuJieMian extends Frame implements ActionListener
07 {
08     JLabel JLUserName =new JLabel("用户名："); //使用文本创建一个标签对象
09     JLabel JLPaw =new JLabel("密    码："); //使用文本创建一个标签对象
10     JTextField JTUserName=new JTextField(); //创建一个文本框对象
11     JPasswordField JPsw =new JPasswordField(); //创建一个密码框对象
12     JButton JB1 =new JButton("登录"); //创建按钮对象
13     JButton JB2 =new JButton("取消");
14     JLabel JL1 =new JLabel("身    份："); //使用文本创建一个标签对象
15     JComboBox JC =new JComboBox(); //创建一个组合框对象
16
17     public DengLuJieMian()
18     {
19         this.setTitle("学生信息管理系统"); //设置窗口标题
20         this.setLayout(null); //设置窗口布局管理器
21         JLUserName.setBounds(100,40,100,20); //设置姓名标签的初始位置
22         this.add(JLUserName); //将姓名标签组件添加到容器
23         JTUserName.setBounds(200,40,80,20); //设置文本框的初始位置
24         this.add(JTUserName); //将文本框组件添加到容器

```

```

25         JLPAw.setBounds(100,100,60,20);           //设置密码标签的初始位置
26         this.add(JLPAw);                         //将密码标签组件添加到容器
27         JPSw.setBounds(200,100,80,20);           //设置密码框的初始位置
28         this.add(JPSw );
29         JL1.setBounds(100,150,60,20);           //设置身份标签的初始位置
30         this.add(JL1);
31         JC.setBounds(200,150,80,20);           //将身份标签组件添加到容器
32         this.add(JC);
33         JC.addItem(new String("学生"));          //设置组合框的初始位置
34         JC.addItem(new String("老师"));          //将组合框组件添加到容器
35         JC1.setBounds(100,200,60,20);           //给组合框添加内容
36         this.add(JC1);
37         JC1.addActionListener(this);             //设置登录按钮的初始位置
38         JB2.setBounds(200,200,60,20);           //将登录按钮组件添加到容器
39         this.add(JB2);
40         JB2.addActionListener(this);             //给按钮添加监听器
41         this.setVisible(true);                  //设置取消按钮的初始位置
42         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //将取消按钮组件添加容器
43         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //给按钮添加监听器
44         this.setSize(10,10,400,250);            //设置窗口的可见性
45         addWindowListener(new WindowAdapter() { //设置窗口尺寸大小
46             public void windowClosing(WindowEvent e) {
47                 System.exit(0);
48             }
49         });
50     public static void main(String args[])
51     {
52         new DengLuJieMian();
53     }
54     public void actionPerformed(ActionEvent e)
55     {
56         if(e.getSource()==JB1)                  //处理登录事件
57         {
58             String name=JTUserName.getText();    //将文本框中包含的文本传给
59             String password=new String(JPSw.getPassword()); //将密码框中包含的
60             //password                                         //文本传给字符串
61             String box=(String)JC.getSelectedItem(); //将当前所选项传给字符串box
62             if((name!=null&&(name.equals("daishu")))&&(password!=null&&
63             (password.equals("0816"))))                //判断语句
64             {
65                 if(box.equals("学生"))                  //选择学生身份登录
66                 {
67                     //new StudentJieMian();               //调用学生信息主窗体
68                 }
69                 else if(box.equals("教师"))           //选择教师身份登录
70                 {
71                     //new TeacherJieMian();              //调用教师信息主窗体
72                 }
73             }
74         }
75     }
76 }

```

【运行结果】使用javac编译程序将产生一个和该程序对应的class程序，然后使用Java



图 21-3 登录界面

运行编译产生的 class 程序，运行结果如图 21-3 所示。

【代码解析】该程序虽然是比较长的，但是其中没有一点新内容，这些知识点都是在 Swing 一章中进行详细讲解的。如果读者有疑问，可以再次学习前面的内容。因为开发该程序时，还没有创建老师界面和学生界面，所以在程序的第 67 行和第 71 行是注释掉的。



21.4 学生界面开发

在本节中将来开发学生界面，在学生界面中，学生可以对自己的信息进行查询，第一次登录时还可以对自己的信息进行插入，并且学生能够查询自己的成绩。

21.4.1 界面设计

因为学生要完成对信息和成绩的操作，所以这里的设计是在界面中定义两个菜单，分别进行信息和成绩的操作。因为对信息的操作包括插入和查询，所以还需要在信息菜单下定义“插入”和“查询”两个子菜单。

21.4.2 程序开发

对界面进行设计后，就可以进行程序开发。同样首先是创建一个窗口，在窗口中要创建两个菜单，并且在信息菜单下还要创建“插入”和“查询”两个子菜单。

【范例 21-2】示例代码 21-2 是一个学生界面程序。

示例代码 21-2

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import java.util.*;
05 import javax.swing.event.*;
06 class StudentJieMian extends JFrame implements ActionListener
07 {
08     JMenuBar jm = new JMenuBar(); // 创建一个菜单栏对象
09     JMenu jm1 = new JMenu("信息"); // 创建一个菜单名为“信息”的菜单对象
10     JMenuItem jmi1=new JMenuItem("插入"); // 创建一个文字标签为“插入”的菜单项对象
11     JMenuItem jmi2=new JMenuItem("查询"); // 创建一个文字标签为“查询”的菜单项对象
12     JMenu jm2=new JMenu("成绩"); // 创建一个菜单名为“成绩”的菜单对象
13     JMenuItem jmi21=new JMenuItem("查询"); // 创建一个文字标签为“查询”的菜单项对象
14     public StudentJieMian()
15     {
16         this.setTitle("学生界面"); // 设置窗口标题
17         this.setLayout(new CardLayout()); // 设置窗口布局管理器
18         this.setJMenuBar(jm); // 将菜单栏组件添加容器
19         jm.add(jm1); // 将信息菜单添加到菜单栏
20         jm.add(jm2); // 将成绩菜单添加到菜单栏
21         jm1.add(jmi1); // 将插入信息菜单项添加到信息菜单
22         jm1.add(jmi2); // 将查询信息菜单项添加到信息菜单
```

```

23         jmi1.addActionListener(this);           //给插入信息菜单项添加监听器
24         jmi2.addActionListener(this);           //给查询信息菜单项添加监听器
25         jm2.add(jmi21);                      //将查询菜单项添加到查询菜单
26         jm2.addActionListener(this);           //给查询菜单添加监听器
27         jmi21.addActionListener(this);          //给查询菜单项添加监听器
28         this.setBounds(10,10,500,400);        //设置窗口尺寸大小
29         this.setVisible(true);                //设置窗口的可见性
30     }
31     public void actionPerformed(ActionEvent e)
32     {
33         if(e.getSource()==jmi1)                 //处理“添加信息”事件
34         {
35             //new AddStudent();
36         }
37         if(e.getSource()==jmi2)                 //处理“查询信息”事件
38         {
39             //new SelectStudent();
40         }
41         if(e.getSource()==jm21)                 //处理“成绩查询”事件
42         {
43             //new ChengJiStudent();
44         }
45     }
46     public static void main(String args[])
47     {
48         new StudentJieMian(); //创建一个对象
49     }
50 }

```

【运行结果】 使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 21-4 所示。

【代码解析】 在学生界面中，定义了两个菜单，分别是“信息”菜单和“成绩”菜单，在这些菜单下还有子菜单，并且为每一个子菜单都定义了事件。单击“插入信息”子菜单时，将进入插入信息界面。同样其他子菜单也是一样的。

21.4.3 开发插入学生界面

在学生界面中单击“信息”菜单下的“插入”子菜单，就会进入学生插入界面，在该界面中学生可以输入自己的信息。

【范例 21-3】 示例代码 21-3 是一个学生插入信息界面程序。

示例代码 21-3

```

01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import java.util.*;
05 import javax.swing.event.*;
06 import java.sql.*;
07 class AddStudent extends JFrame implements ActionListener
08 {
09     JLabel JL = new JLabel("添加基本信息",JLabel.CENTER); //使用文本创建一个标签
                                         //签对象

```

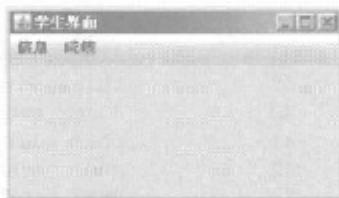


图 21-4 学生界面



```
10
11     JLabel    JLNumber=new JLabel("学号:");
12     JTextField JTNumber=new JTextField();           //使用文本创建一个标签对象
13     JLabel    JLName =new JLabel("姓名:");
14     JTextField JTName =new JTextField();           //使用文本创建一个标签对象
15     JLabel    JLClass =new JLabel("班级:");
16     JTextField JTClass =new JTextField();           //使用文本创建一个标签对象
17     JLabel    JLsex  =new JLabel("性别:");
18     ButtonGroup BG   =new ButtonGroup();           //创建一个 ButtonGroup 组件对象
19     JRadioButton JRB1 =new JRadioButton("男");      //创建一个单选按钮对象
20     JRadioButton JRB2 =new JRadioButton("女");
21     JLabel    JL1   =new JLabel("学院:");
22     JTextField JT1   =new JTextField();           //使用文本创建一个标签对象
23     JButton   JBAdd =new JButton("添加");          //创建按钮对象
24     JButton   JBNext =new JButton("重置");
25     String sql="";
26     public AddStudent()
27     {
28         this.setTitle("添加学生信息");
29         this.setLayout(null);
30         JL.setBounds(100,30,200,40);                //设置窗口标题
31         this.add(JL);                            //设置窗口布局管理器
32         JLNumber.setBounds(100,80,100,20);          //设置标签的初始位置
33         this.add(JLNumber);                      //将标签添加到容器
34         JTNumber.setBounds(200,80,80,20);          //设置学号标签的初始位置
35         this.add(JTNumber);                      //将学号标签添加到容器
36         JLName.setBounds(100,120,80,20);          //设置姓名标签的初始位置
37         this.add(JLName);                        //将姓名标签添加到容器
38         JTName.setBounds(200,120,80,20);          //设置文本框的初始位置
39         this.add(JTName);                        //将文本框添加到容器
40         JLsex.setBounds(100,160,100,20);          //设置性别标签的初始位置
41         this.add(JLsex);                         //将性别标签添加到容器
42         JRB1.setBounds(200,160,40,20);          //设置单选按钮的初始位置
43         JRB2.setBounds(300,160,40,20);
44         this.add(JRB1);                          //添加单选按钮组件
45         this.add(JRB2);                          //将单选按钮添加到 Button Group 组件
46         BG.add(JRB1);
47         BG.add(JRB2);
48         JTClass.setBounds(100,240,60,20);          //设置班级标签的初始位置
49         this.add(JTClass);                       //将班级标签添加到容器
50         JTClass.setBounds(200,240,80,20);          //设置文本框的初始位置
51         this.add(JTClass);                      //将文本框添加到容器
52         JL1.setBounds(100,280,60,20);          //设置学院标签的初始位置
53         this.add(JL1);                          //将学院标签添加到容器
54         JT1.setBounds(200,280,80,20);          //设置文本框的初始位置
55         this.add(JT1);                          //将文本框添加到容器
56         JBAdd.setBounds(80,320,90,20);          //设置添加按钮的初始位置
57         this.add(JBAdd);                        //将添加按钮添加到容器
58         JBAdd.addActionListener(this);           //给按钮添加监听器
59         JBNext.setBounds(190,320,90,20);          //设置重置按钮的初始位置
60         this.add(JBNext);                       //将重置按钮添加到容器
61         JBNext.addActionListener(this);           //给按钮添加监听器
62         this.setBounds(10,10,500,400);          //设置窗口尺寸大小
63         this.setVisible(true);                  //设置窗口的可见性
```

```

64        addWindowListener(new WindowAdapter()
65        {
66            public void windowClosing(WindowEvent e)
67            {
68                System.exit(0);
69            }
70        });
71    public void actionPerformed(ActionEvent e)
72    {
73
74        if(e.getSource()==JBAdd)           //处理“添加”事件
75        {
76            String snumber=JTNumber.getText(); //将文本框中包含的文本传给字符串
77                                         //snumber
77            String sname=JTName.getText();   //将文本框中包含的文本传给字符串
78                                         //sname
78            String sclass=JCClass.getText(); //将文本框中包含的文本传给字符串
79                                         //scClass
79            String ssex="女";             //返回单选按钮的值
80            if(JRB1.isSelected())
81                ssex="男";
82            String scollect=JT1.getText(); //将文本框中包含的文本传给字符串
82                                         //scollect
83            sql="select * from student where Id='"+snumber+"'"; //检索出 id
83                                         //等于 snumber 的学生的所有 84 信息
84            try
85            {
86                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //加载驱动程序
87                Connection cot=DriverManager.getConnection("jdbc:odbc:");
87                                         //student1","",""); //打开数据
88                                         //库 B9 连接,student1 为数据源名称
89                Statement stm=cot.createStatement(); //提交查询
90                ResultSet rs=stm.executeQuery(sql); //取得查询结果
91
92                if(rs.next())           //判断结果是否存在
93                    JOptionPane.showMessageDialog(null,"该号已经存在!"); //通过
93                                         // 95 showMessageDialog() 方法打印信息
94                else
95                {
96                    sql="insert into student 99 values('"+snumber+"',
96                                         '"+sname+"','"+sclass+"','"+ssex+"','"+scollect+"')"; //插入一组数据
97                    int i=stm.executeUpdate(sql); //对数据库进行更新
98                    if(i>0)
99                        JOptionPane.showMessageDialog(null,"添加成功!");
100                    else
101                        JOptionPane.showMessageDialog(null,"删除失败!");
102                }
103            }catch(Exception ee)
104            {
105            }
106        }
107        if(e.getSource()==JBNext)          //处理“重置”事件
108        {
109            JTNumber.setText(null);       //设置文本的 text 值为 null
110            JTName.setText(null);       //设置文本的 text 值为 null
111            JCClass.setText(null);      //设置文本的 text 值为 null
112            JT1.setText(null);         //设置文本的 text 值为 null
113        }
114    }
115}

```



21 天学通 Java

```
17      }
18      public static void main(String args[])
19      {
20          new AddStudent();
21      }
22  }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 21-5 所示。

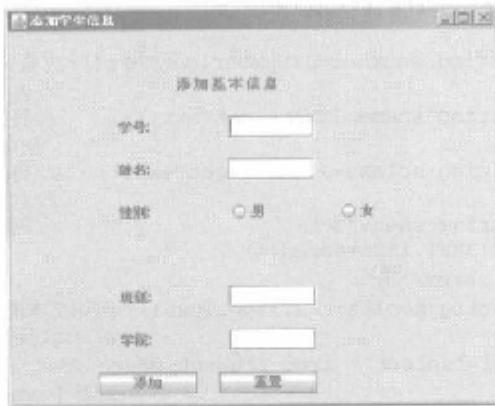


图 21-5 插入信息界面

【代码解析】该程序是一个学生插入信息的界面程序，在该程序中需要用户填写学生的学号、姓名、性别、班级和学院。用户填写后，单击“添加”按钮，就会将这些信息添加到数据库中。

21.4.4 查询学生信息界面

学生第一次插入信息后，老师是可以对学生的信息进行修改和删除的。除此之外，学生还可以查询自己被修改后的信息，在信息菜单下有一个查询子菜单，单击该菜单就触发事件，从而进入查询学生信息界面。

【范例 21-4】示例代码 21-4 是查询学生信息界面。

示例代码 21-4

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import java.util.*;
05 import javax.swing.event.*;
06 import java.sql.*;
07 import java.util.Vector;
08 class SelectStudent extends JFrame implements ActionListener
09 {
10     JLabel    JI    =new JLabel("查询学生信息",JLabel.CENTER);
11     JLabel    JLNumber=new JLabel("请输入学号:");
12     JTextField JTNumber=new JTextField();           //使用文本创建一个文本框对象
13     JLabel    JLName =new JLabel("姓名:");
14     JTextField JTName =new JTextField();           //使用文本创建一个文本框对象
15     JLabel    JLCClass =new JLabel("班级:");
16     JTextField JTClass =new JTextField();           //使用文本创建一个文本框对象
```

```

17     JLabel JLsex =new JLabel("性别:");           //使用文本创建一个标签对象
18     ButtonGroup BG =new ButtonGroup(); //创建一个ButtonGroup 组件对象
19     JRadioButton JRB1 =new JRadioButton("男");    //创建一个单选按钮对象
20     JRadioButton JRB2 =new JRadioButton("女");
21     JLabel JL1 =new JLabel("学院:");           //使用文本创建一个标签对象
22     JTextField JT1 =new JTextField();          //创建一个文本框对象
23     JButton JBSet =new JButton("查询");         //创建按钮对象
24     JButton JBNext =new JButton("重置");        //定义一个字符串
25     String sql="";
26     public SelectStudent()                      //创建 SelectStudent 构造函数
27     {
28         this.setTitle("查询学生信息");           //设置窗口标题
29         this.setLayout(null);                  //设置窗口布局管理器
30         JL.setForeground(Color.red);          //设置标签的前景色
31         JL.setFont(new java.awt.Font("宋体",Font.PLAIN,19)); //设置标签的字体
32         JL.setBounds(100,30,200,40);          //设置标签的初始位置
33         this.add(JL);                       //将标签添加到容器
34         JLNumber.setBounds(100,80,100,20);      //设置学号标签的初始位置
35         this.add(JLNumber);                 //将学号标签添加到容器
36         JTNumber.setBounds(200,80,80,20);      //设置文本框的初始位置
37         this.add(JTNumber);                 //将文本框添加到容器
38         JLName.setBounds(100,160,60,20);       //设置姓名标签的初始位置
39         this.add(JLName);                   //将姓名标签添加到容器
40         JTName.setBounds(200,160,80,20);      //设置文本框的初始位置
41         this.add(JTName);                   //将文本框添加到容器
42         JLsex.setBounds(100,200,100,20);      //设置性别标签的初始位置
43         this.add(JLsex);                   //将性别标签添加到容器
44         JRB1.setBounds(200,200,40,20);       //设置单选按钮的初始位置
45         JRB2.setBounds(300,200,40,20);
46         this.add(JRB1);                   //添加单选按钮组件
47         this.add(JRB2);                   //将单选按钮添加到 Button Group 组件
48         BG.add(JRB1);                   //设置班级标签的初始位置
49         BG.add(JRB2);                   //将班级标签添加到容器
50         JLClass.setBounds(100,280,60,20);     //设置文本框的初始位置
51         this.add(JLClass);                //将文本框添加到容器
52         JTClass.setBounds(200,280,80,20);     //设置文本框的初始位置
53         this.add(JTClass);                //将文本框添加到容器
54         JL1.setBounds(100,320,60,20);       //设置学院标签的初始位置
55         this.add(JL1);                   //将学院标签添加到容器
56         JT1.setBounds(200,320,80,20);       //设置文本框的初始位置
57         this.add(JT1);                   //将文本框添加到容器
58         JBSet.setBounds(80,120,90,20);      //设置查询按钮的初始位置
59         this.add(JBSet);                  //将查询按钮添加到容器
60         JBSet.addActionListener(this);      //给按钮添加监听器
61         JBNext.setBounds(190,120,90,20);     //设置重置按钮的初始位置
62         this.add(JBNext);                 //将重置按钮添加到容器
63         JBNext.addActionListener(this);     //给按钮添加监听器
64         this.setBounds(10,10,500,400);      //设置窗口尺寸大小
65         this.setVisible(true);            //设置窗口的可见性
66         addWindowListener(new WindowAdapter())
67         {
68             public void windowClosing(WindowEvent e)
69             {
70                 System.exit(0);
71             }
72         } //通过内部类重写关闭窗体的方法

```



```
72     }
73     public void actionPerformed(ActionEvent e)
74     {
75         if(e.getSource()==JButton) //处理“查询”事件
76         {
77             String snumber=JTNumber.getText(); //将文本框中包含的文本传给字符串
78                                         //snumber
79             String sname=JTName.getText(); //将文本框中包含的文本传给字符串 sname
80             String sclass=JTCClass.getText(); //将文本框中包含的文本传给字符串
81                                         //sclass
82             String sex="女"; //返回单选按钮的值
83             if(JRB1.isSelected())
84                 sex="男";
85             String scollect=JT1.getText(); //将文本框中包含的文本传给字符串
86                                         //scollect
87             sql="select * from student where Id='"+snumber+"'"; //检索出 Id
88                                         //等于 snumber 的学生的所有信息
89
90             try //异常处理
91             {
92                 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //加载驱动程序
93                 Connection cot=DriverManager.getConnection("jdbc:odbc:
94                     student1","",""); //打开数据库 30 //连接, student1 为数据源名称
95                 Statement stm=cot.createStatement(); //提交查询
96                 ResultSet rs=stm.executeQuery(sql); //取得查询结果
97                 if(rs.next()) //判断结果是否存在
98                 {
99                     String name=rs.getString(2); //获取当前行中指定列的值, 并将
                                         //返回的字符串对象赋给 name
100
101                     JTName.setText(name); //将字符串 name 显示在文本框中
102                     String clas=rs.getString(3); //获取当前行中指定列的值, 并将返
                                         //回的字符串对象赋给 clas
103
104                     JTCClass.setText(clas); //将字符串 clas 显示在文本框中
105                     String sex=rs.getString(4); //获取当前行中指定列的值, 并将返回
                                         //的字符串对象赋给 sex
106
107                     JRB1.setText(sex); //将字符串 sex 显示在文本框中
108                     String collect=rs.getString(5); //获取当前行中指定列的值, 并将返
                                         //回的字符串对象赋给 collect
109
110                     JT1.setText(collect); //将字符串 collect 显示在文本框中
111                     String bir=rs.getString(6); //获取当前行中指定列的值, 并将返回
                                         //的字符串对象赋给 bir
112
113                     int n=stm.executeUpdate(sql); //对数据库进行更新
114                     if(n>0)
115                         JOptionPane.showMessageDialog(null, "查询成功!");
116                     else
117                         JOptionPane.showMessageDialog(null, "查询失败!");
118
119                 } catch(Exception ee)
```

```

20         {
21     }
22     }
23     if(e.getSource()==JBNext) //处理“重置”事件
24     {
25         JTNumber.setText(null); //设置文本的text值为null
26         JTName.setText(null); //设置文本的text值为null
27         JTClass.setText(null); //设置文本的text值为null
28         JT1.setText(null); //设置文本的text值为null
29     }
30 }
31 public static void main(String args[])
32 {
33     new SelectStudent();
34 }
35 }
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 21-6 所示。

【代码解析】该程序是一个学生查询信息界面程序。在该界面中，学生只需要输入自己的学号就能够查询出自己的班级和学院等信息。这里只是一个简单的演示，读者可以向其中添加更多的信息。

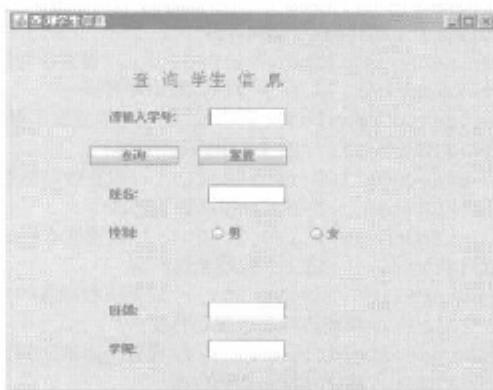


图 21-6 查询学生信息

21.4.5 查询成绩信息

在学生界面中还有一个“成绩”菜单，在学生的界面该菜单下只有一个“查询”子菜单。单击“查询”子菜单，将触发事件，进入到查询成绩界面。

【范例 21-5】示例代码 21-5 是一个查询成绩界面程序。

示例代码 21-5

```

01 import java.awt.*;
02 import java.awt.event.*;
03 import javax.swing.*;
04 import java.util.*;
05 import javax.swing.event.*;
06 import java.sql.*;
07 import java.util.Vector;
08 class ChengJiStudent extends JFrame implements ActionListener
```



21 天学通 Java

```
09  {
10      JLabel    JL     =new JLabel("查询成绩",JLabel.CENTER); //使用文本创建一个
11      //标签对象
12      JLabel    JLNumber=new JLabel("请输入学号:"); //使用文本创建一个标签对象
13      JTextField JTNumber=new JTextField(); //创建一个文本框对象
14      JLabel    JLName =new JLabel("姓名:"); //使用文本创建一个标签对象
15      JTextField JTName =new JTextField(); //创建一个文本框对象
16      JLabel    JLClass =new JLabel("语文:"); //使用文本创建一个标签对象
17      JTextField JTClass =new JTextField(); //创建一个文本框对象
18      JLabel    JL1     =new JLabel("数学:"); //使用文本创建一个标签对象
19      JTextField JT1     =new JTextField(); //创建一个文本框对象
20      JLabel    JL2     =new JLabel("班级:"); //使用文本创建一个标签对象
21      JTextField JT2     =new JTextField(); //创建一个文本框对象
22      JButton   JBSet  =new JButton("查询"); //创建按钮对象
23      JButton   JBNext =new JButton("重置");
24      String sql=""; //定义一个字符串
25      public ChengJiStudent() //创建 SetGrade 构造函数
26      {
27          this.setTitle("查询成绩"); //设置窗口标题
28          this.setLayout(null); //设置窗口布局管理器
29          JL.setForeground(Color.red); //设置标签的前景色
30          JL.setFont(new java.awt.Font("宋体",Font.PLAIN,19)); //设置标签的字体
31          JL.setBounds(100,30,200,40); //设置标签的初始位置
32          this.add(JL); //将标签添加到容器
33          JLNumber.setBounds(100,80,100,20); //设置学号标签的初始位置
34          this.add(JLNumber); //将学号标签添加到容器
35          JTNumber.setBounds(200,80,80,20); //设置文本框的初始位置
36          this.add(JTNumber); //将文本框添加到容器
37          JLName.setBounds(100,160,60,20); //设置姓名标签的初始位置
38          this.add(JLName); //将姓名标签添加到容器
39          JTName.setBounds(200,160,80,20); //设置文本框的初始位置
40          this.add(JTName); //将文本框添加到容器
41          JL2.setBounds(100,240,80,20); //设置班级标签的初始位置
42          this.add(JL2); //将班级标签添加到容器
43          JT2.setBounds(200,240,80,20); //设置文本框的初始位置
44          this.add(JT2); //将文本框添加到容器
45          JLClass.setBounds(100,280,60,20); //设置语文标签的初始位置
46          this.add(JLClass); //将语文标签添加到容器
47          JTClass.setBounds(200,280,80,20); //设置文本框的初始位置
48          this.add(JTClass); //将文本框添加到容器
49          JL1.setBounds(100,320,60,20); //设置数学标签的初始位置
50          this.add(JL1); //将数学标签添加到容器
51          JT1.setBounds(200,320,80,20); //设置文本框的初始位置
52          this.add(JT1); //将文本框添加到容器
53          JBSet.setBounds(80,120,90,20); //设置查询按钮的初始位置
54          this.add(JBSet); //将查询按钮添加到容器
55          JBSet.addActionListener(this); //给按钮添加监听器
56          JBNext.setBounds(190,120,90,20); //设置重置按钮的初始位置
57          this.add(JBNext); //将重置按钮添加到容器
58          JBNext.addActionListener(this); //给按钮添加监听器
59          this.setBounds(10,10,500,400); //设置窗口尺寸大小
60          this.setVisible(true); //设置窗口的可见性
61          addWindowListener(new WindowAdapter()
62          {
63              public void windowClosing(WindowEvent e)
64          }
65      }
66  }
```

```
62         {
63             System.exit(0);
64         }
65     });
66 }
67 public void actionPerformed(ActionEvent e)
68 {
69     if(e.getSource()==JBSot) //处理“查询”事件
70     {
71         String snumber=JTNumber.getText(); //将文本框中包含的文本传给字符串
72                                         //snumber
73         String sname=JTName.getText(); //将文本框中包含的文本传给字符串 sname
74         String sclass=JTClass.getText(); //将文本框中包含的文本传给字符串
75                                         //sclass
76         String sssex="女"; //返回单选按钮的值
77         String scollect=JT1.getText(); //将文本框中包含的文本传给字符串
78                                         //scollect
79         String sbir=JT2.getText(); //将文本框中包含的文本传给字符串 sbir
80         sql="select * from grade where Id='"+snumber+"'"; //检索出 id 等于
81                                         //snumber 的学生的所有信息
82
83     try //异常处理
84     {
85         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //加载驱动程序
86         Connection cot=DriverManager.getConnection("jdbc:odbc:
87             student1","",""); //打开数据库连接, student1 为数据源名称
88
89         Statement stm=cot.createStatement(); //提交查询
90         ResultSet rs=stm.executeQuery(sql); //取得查询结果
91         if(rs.next()) //判断结果是否存在
92         {
93             String name=rs.getString(2); //获取当前行中指定列的值, 并将
94                                         //返回的字符串对象赋给 name
95             JTName.setText(name); //将字符串 name 显示在文本框中
96
97             String clas=rs.getString(3); //获取当前行中指定列的值, 并将返
98                                         //回的字符串对象赋给 clas
99             JTClass.setText(clas); //将字符串 clas 显示在文本框中
100
101             String collect=rs.getString(5); //获取当前行中指定列的值, 并将返
102                                         //回的字符串对象赋给 collect
103
104             JT1.setText(collect); //将字符串 collect 显示在文本框中
105
106             String bir=rs.getString(6); //获取当前行中指定列的值, 并将返
107                                         //回的字符串对象赋给 bir
108
109             JT2.setText(bir); //将字符串 bir 显示在文本框中
110
111             int n=stm.executeUpdate(sql); //对数据库进行更新
112             if(n>0)
113                 JOptionPane.showMessageDialog(null, "查询成功!"); //通过
114                                         //showMessageDialog()方法打印信息
```



```
08             else
09                 JOptionPane.showMessageDialog(null, "查询失败！");
10            }
11        else
12        {
13            JOptionPane.showMessageDialog(null, "此用户不存在！");
14        }
15    }catch(Exception ee)
16    {
17    }
18}
19if(e.getSource()==JBNext)//处理“重置”事件
20{
21    JTNumber.setText(null);//设置文本的text值为null
22    JTName.setText(null);//设置文本的text值为null
23    JTClass.setText(null);//设置文本的text值为null
24    JT1.setText(null);//设置文本的text值为null
25    JT2.setText(null);//设置文本的text值为null
26}
27}
28public static void main(String args[])
29{
30    new ChengJiStudent();//实例化一个对象
31}
32}
```

【运行结果】使用 javac 编译程序将产生一个和该程序对应的 class 程序，然后使用 Java 运行编译产生的 class 程序，运行结果如图 21-7 所示。

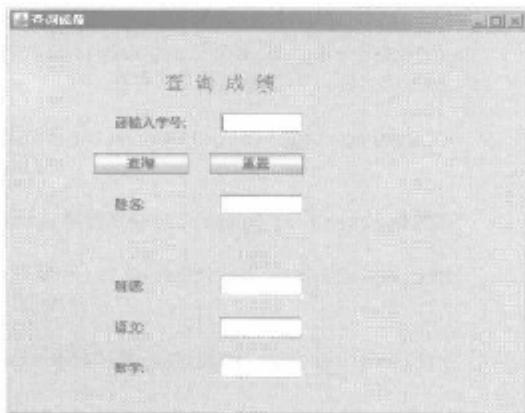


图 21-7 查询成绩

【代码解析】在该程序中，学生只需要输入学号后，单击“查询”按钮就能够查询出该学号学生的姓名、班级，以及数学和语文的成绩。



21.5 综合练习

参考本章中学生界面，开发学生系统中的老师界面。

【提示】老师界面和学生界面非常类似，但也存在很多不一样的地方。首先在老师界面上能够对学生的信息除了插入和查询之外，还能够对学生的信息进行修改和删除，这样



就需要在“信息”菜单下增加“修改”和“删除”两个子菜单，也需要开发修改学生信息界面和删除学生信息界面。

在学生界面中只能够进行对程序的查询，在老师界面中不但能够对学生的信息进行查询，还能够对学生的信息进行修改、插入和删除，同样也需要增加相应的子菜单和对应的界面。



21.6 小结

本章以一个实际开发的综合案例来讲解了如何使用 Java 进行开发。这里以学生管理系统为例，在该程序中主要应用了 Java 中的 Swing 知识和数据库连接知识。如果想了解更多的关于实际开发的内容，可以通过电子工业出版社出版的《Java 优化编程（第 2 版）》一书来进行更详细的学习。