第15章

用 Swing 开发 GUI 程序

GUI，即图形用户界面，可以为我们提供丰富多彩的程序。本章讲解 javax.swing 中的一些 API，主要涉及窗口开发、控件开发、颜色、字体和图片开发，最后讲解了一些常见的其他功能。

本章术语

GUI _____

Swing _____

JFrame _____

Component _____

JButton _____

Color _____

Font _____

Image _____

Icon _____

15.1 认识 GUI 和 Swing

Note

15.1.1 什么是 GUI

GUI，是 Graphics User Interface 的简称，即图形用户界面。

我们以前编写的程序，其操作基本是在控制台上进行，称为文本用户界面，或字符用户界面，用户操作很不方便。而图形用户界面可以让用户看到什么就操作什么，而不是通过文本提示来操作。Windows 中的计算器，就是一个典型的图形用户界面：

图 15-1

很明显，和控制台程序相比，图形用户界面操作更加直观，能够提供更加丰富的功能。

注意

能否说任何软件都必须用图形用户界面呢？这也不一定。和控制台程序相比，图形用户界面比较消耗资源，并且需要更多的硬件支持。虽然就日常电脑用户来说，这是一件很常见的事情，但是某些精密系统的操作，并不一定需要优美的界面。

从本章开始，我们将讲解用 Java 语言开发图形用户界面。

15.1.2 什么是 Swing

在 Java 中，GUI 操作的支持 API，一般保存在 java.awt 和 javax.swing 包中。所以，本章的内容，主要基于这两个包进行讲解。

Java 对 GUI 的开发，有两套版本的 API。

1. java.awt 包中提供的 AWT(Abstract Window Toolkit，抽象窗口工具包)界面开发 API，适合早期 Java 版本。

2. javax.swing 包中提供的 Swing 界面开发 API，功能比 AWT 更加强大，是 Java2 推出的，成为 JavaGUI 开发的首选。其中，javax 中的“x”是扩展的意思。

本书的讲解主要针对 Swing 展开。

特别提醒

界面开发的学习过程中，一定要多看文档，死记硬背是没有用的，实际上也不是学习的好习惯。

例如，曾经有学生发邮件问笔者：多行文本框中的内容如何自动回车，我给他回了邮件，告诉他调用哪个函数；几天之后，他又问：多行文本框如何加滚动条，我觉得该学生这样下去，估计无法学会 Java，我回的邮件是：用一天的时间，将文档中多行文本框中的所有成员函数都用一遍，不懂再来问。

我们必须注意，不要去刻意记住某个功能如何实现，而是要养成查文档的习惯。只有那种毫无品味的面试官，才会去考学生“多行文本框中的内容如何自动回车”。

Note

15.2　使用窗口

制作图形用户界面，首要的问题是：如何显示一个窗口，哪怕这个窗口上什么都没有，至少这是所有图形界面的基础。

15.2.1　用 JFrame 类开发窗口

一般情况下，我们使用 javax.swing.JFrame 类来进行窗口显示。

JFrame 类提供了窗口功能，打开文档，找到 javax.swing.JFrame 类，最常见的构造函数是：

```
public JFrame(String title) throws HeadlessException
```

传入一个界面标题，实例化 JFrame 对象。

我们可以调用 JFrame 类里面的函数来进行窗口操作，主要功能有：

1. 设置标题：

```
public void setTitle(String title)
```

2. 设置在屏幕上的位置：

```
public void setLocation(int x, int y)
```

其中，x 为窗口左上角在屏幕上的横坐标，y 为左上角在屏幕上的纵坐标。屏幕最左上角的横纵坐标为 0。

3. 设置大小：

```
public void setSize(int width, int height)
```

参数为宽度和高度。

4. 设置可见性：

```
public void setVisible(boolean b)
```

根据参数 b 的值显示或隐藏此窗口。

其他内容，大家可以参考文档。

以下代码显示一个窗口：

Note

FrameTest1.java

```
package window;
import javax.swing.JFrame;
public class FrameTest1 {
    public static void main(String[] args) {
        JFrame frm = new JFrame("这是一个窗口");
        frm.setLocation(30,50);
        frm.setSize(50,60);
        frm.setVisible(true);
    }
}
```

运行，显示窗口：

图 15-2

注意

点击该窗口右上角的“关闭”按钮，窗口消失，但是程序并没有结束。解决方法是，调用方法：frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

阶段性作业

和 JFrame 类似，JWindow 也可以生成窗口，没有标题栏、窗口管理按钮。请查文档，在桌面显示一个 JWindow 对象。

15.2.2 用 JDialog 类开发窗口

用 JDialog 类，也可以开发窗口，此时，创建的窗口是对话框。

打开文档，找到 javax.swing.JDialog 类，最常见的构造函数是：

public JDialog(Frame owner, String title,boolean modal) throws HeadlessException

其参数解释如下：owner 表示显示该对话框的父窗口；title 为该对话框的标题；modal 为 true 表示是模态对话框。

什么是父窗口和模态对话框呢？我们在 Windows 操作中经常遇到一个情况：从窗口 A 中打开窗口 B，此时，窗口 A 可以叫做窗口 B 的父窗口。

在打开窗口 B 时，可能出现一种情况：窗口 B 没有关闭时，窗口 A 不能使用，比如记事本中的“字体”对话框：

图 15-3

该对话框不关闭，记事本界面不能使用，此时，“字体”对话框就是一个模态对话框，否则就是一个非模态对话框。

调用 JDialog 类里面的函数来进行窗口操作，主要功能和 JFrame 类似。

以下代码在一个 JFrame 的基础上产生一个模态对话框：

DialogTest1.java

```
package window;
import javax.swing.JDialog;
import javax.swing.JFrame;
public class DialogTest1 {
    public static void main(String[] args) {
        JFrame frm = new JFrame("这是一个窗口");
        frm.setSize(200,100);
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.setVisible(true);

        JDialog dlg = new JDialog(frm,"这是一个对话框",true);
        dlg.setSize(100,50);
        dlg.setVisible(true);
    }
}
```

运行，显示两个窗口，前面的对话框不关闭，后面的窗口不能使用：

图 15-4

15.3 使用控件

Note

15.3.1 什么是控件

控件，实际上不是一个专有名词，而是一个俗称。比如，我们使用的按钮、文本框统称为控件，在 Java 中，有时又称 Component(组件)。控件一般都有相应的类来实现，比如，最常见的控件是按钮，在 Java 中就是用 JButton 类来实现的。

本节讲解的控件，基本上都是 javax.swing.JComponent 类的子类。

我们要将控件加到窗口上，为了对控件更好地组织，通常将控件加到面板(JPanel)上，再加到窗口上去。

以下代码就是将一个按钮加到面板上，然后加到窗口上去。代码如下：

ComponentTest1.java

```
package component;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class ComponentTest1 extends JFrame{
    private JButton jbt = new JButton("按钮");
    private JPanel jpl = new JPanel();
    public ComponentTest1(){
        jpl.add(jbt);
        this.add(jpl);
        this.setSize(300,500);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new ComponentTest1();
    }
}
```

运行，效果为：

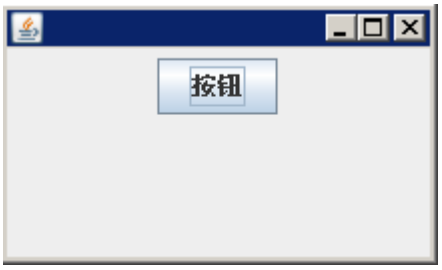


图 15-5

特别说明

1. 面板和窗口，我们也称为容器对象，在容器上可以添加容器，也可以添加控件，使用的是 add 方法。

2. 由于界面有可能比较复杂，所以我们一般不将界面的生成过程写在主函数中，而是写一个类继承 JFrame，在其构造函数中初始化界面。

Note

15.3.2　标签、按钮、文本框和密码框

我们使用比较常见的控件是标签、按钮、文本框和密码框。

1. 标签。

标签显示一段静态文本，效果如下：

图 15-6

我们可以用 JLabel 类开发标签，打开文档，找到 javax.swing.JLabel 类，最常见的构造函数是：

public JLabel(String text)

传入一个标题，实例化一个标签。

2. 按钮。

按钮的效果如下：

图 15-7

我们可以用 JButton 类开发按钮，打开文档，找到 javax.swing.JButton 类，最常见的构造函数是：

public JButton(String text)

传入一个标题，实例化一个按钮。

3. 文本框。

文本框效果如下：

图 15-8

我们可以用 JTextField 类开发文本框，打开文档，找到 javax.swing.JTextField 类，最常见的构造函数是：

public JTextField(int columns)

参数为 JTextField 的显示列数。

4. 多行文本框。

多行文本框效果如下：

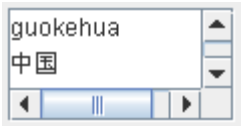


图 15-9

Note

我们可以用 JTextArea 类开发多行文本框，打开文档，找到 javax.swing.JTextArea 类，最常见的构造函数是：

public JTextArea(int rows, int columns)

参数为 JTextArea 显示的行数和列数。

注意

默认的文本框没有滚动条。如果要使用滚动条，需要使用 JScrollPane 类，将 JTextArea 对象传入其构造函数，然后在界面上添加 JScrollPane 对象。

5. 密码框。

密码框效果如下：

图 15-10

输入的内容以掩码形式显示。我们可以用 JPasswordField 类开发密码框，打开文档，找到 javax.swing.JPasswordField 类，最常见的构造函数是：

public JPasswordField(int columns)

参数为 JPasswordField 的显示列数。

以下代码在界面上显示标签、按钮、文本框和密码框。代码如下：

ComponentTest2.java

```
package component;
import javax.swing.*;
public class ComponentTest2 extends JFrame{
    private JLabel lblInfo = new JLabel("这是注册窗口");
    private JButton btReg = new JButton("注册");
    private JTextField tfAcc = new JTextField(10);
    private JPasswordField pfPass = new JPasswordField(10);
    private JTextArea taInfo = new JTextArea(3,10);
    private JScrollPane spTaInfo = new JScrollPane(taInfo);
    private JPanel jpl = new JPanel();
    public ComponentTest2(){
        jpl.add(lblInfo);
        jpl.add(btReg);
        jpl.add(tfAcc);
        jpl.add(pfPass);
        jpl.add(spTaInfo);
```

```
            this.add(jpl);
            this.setSize(150,220);
            this.setVisible(true);
        }
        public static void main(String[] args) {
            new ComponentTest2();
        }
    }
```

Note

运行，效果如下：

图 15-11

阶段性作业

查看文档:

1. 如何改变密码框的掩码，比如用#表示。
2. 如何让多行文本框输入时，自动换行。
3. 尝试用 setSize 方法修改按钮的大小，看看效果如何？

15.3.3　单选按钮、复选框和下拉列表框

单选按钮、复选框和下拉列表框(下拉菜单)也是使用较为常见的控件。

1. 单选按钮。

单选按钮提供多选一功能(比如性别)，效果如下：

图 15-12

我们可以用 JRadioButton 类开发单选按钮，打开文档，找到 javax.swing.JRadioButton 类，最常见的构造函数是：

public JRadioButton(String text, boolean selected)

参数 1 为单选按钮标题，参数 2 为选择状态。

注意

既然单选按钮支持的是多选一，如何将多个单选按钮看成一组呢。我们可以用 javax.swing.ButtonGroup 实现，该类有个 add 函数，能够将多个单选按钮加入，看成一组。但是 ButtonGroup 不能被加到界面上，我们还是要将单选按钮一个个加到界面上去。

Note

2. 下拉列表框。

下拉列表框也是提供多选一功能(适合选项较多的情况)，效果如下：

图 15-13

我们可以用 JComboBox 类开发下拉列表框，打开文档，找到 javax.swing.JComboBox 类，最常见的构造函数是：

public JComboBox()

实例化一个下拉列表框。其中的选项可用其 addItem 函数添加，大家可以参考文档。

3. 复选框。

复选框提供多选功能(可以不选，也可以全选，也可以选一部分)，效果如下：

图 15-14

我们可以用 JCheckBox 类开发复选框，打开文档，找到 javax.swing.JCheckBox 类，最常见的构造函数是：

public JCheckBox(String text, boolean selected)

实例化一个复选框。参数 1 为复选框标题，参数 2 为选择状态。

以下代码在界面上显示上述几种控件。代码如下：

ComponentTest3.java

```
package component;
import javax.swing.*;
public class ComponentTest3 extends JFrame{
    private JRadioButton rbSex1 = new JRadioButton("男",true);
    private JRadioButton rbSex2 = new JRadioButton("女",false);
    private JComboBox cbHome = new JComboBox();
    private JCheckBox cbFav1 = new JCheckBox("唱歌",true);
    private JCheckBox cbFav2 = new JCheckBox("跳舞");
```

Note

```
    private JPanel jpl = new JPanel();
    public ComponentTest3(){

        ButtonGroup bgSex = new ButtonGroup();
        bgSex.add(rbSex1);
        bgSex.add(rbSex2);

        cbHome.addItem("北京");
        cbHome.addItem("上海");
        cbHome.addItem("天津");

        jpl.add(rbSex1);
        jpl.add(rbSex2);
        jpl.add(cbHome);
        jpl.add(cbFav1);
        jpl.add(cbFav2);
        this.add(jpl);
        this.setSize(100,180);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new ComponentTest3();
    }
}
```

运行，效果如下：

图 15-15

阶段性作业

查看文档:

1. 如何获取单选按钮的选定状态?

2. 如何获取下拉菜单中的选定值？
3. 如何在下拉菜单中删除某项？
4. 如何获取复选框的选定状态？

Note

15.3.4 菜单

菜单也是一种常见的控件，效果如下：

图 15-16

如何开发菜单呢？实际上，菜单的开发，需要了解如下问题：

1．界面上首先需要放置一个菜单条，由 javax.swing.JMenuBar 封装。

打开文档，找到 javax.swing.JMenuBar 类，最常见的构造函数是：

public JMenuBar()

实例化一个菜单条。

注意

JFrame 的 setJMenuBar(JMenuBar menubar)方法来可以将菜单条加到界面上。

2. 上图中的“文件”，是一个菜单，由 javax.swing.JMenu 封装。JMenu 放在菜单条上。

打开文档，找到 javax.swing.JMenu 类，最常见的构造函数是：

public JMenu(String s)

参数是菜单文本。

注意

JMenuBar 的 add(JMenu menu)方法可以添加 JMenu。

3. 上图中的“保存”，是一个菜单项，由 javax.swing.JMenuItem 封装。JMenuItem 放在 JMenu 上。

打开文档，找到 javax.swing. JMenuItem 类，最常见的构造函数是：

public JMenuItem (String s)

参数是菜单项文本。

注意

JMenu 的 add(JMenuItem menuItem)方法来可以添加 JMenuItem。

因此，一言以蔽之，本例子中，界面上有个菜单条(JMenuBar)，菜单条上有个文件菜单(JMenu)，文件菜单中有三个菜单项(JMenuItem)。

以下代码在界面上显示上述控件。代码如下：

ComponentTest4.java

```
package component;
import javax.swing.*;
public class ComponentTest4 extends JFrame{
    private JMenuBar mb = new JMenuBar();
    private JMenu mFile = new JMenu("文件");
    private JMenuItem miOpen = new JMenuItem("打开");
    private JMenuItem miSave = new JMenuItem("保存");
    private JMenuItem miExit = new JMenuItem("退出");
    public ComponentTest4(){
        mFile.add(miOpen);
        mFile.add(miSave);
        mFile.add(miExit);
        mb.add(mFile);
        this.setJMenuBar(mb);

        this.setSize(200,180);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new ComponentTest4();
    }
}
```

Note

运行，效果如本节开头的菜单所示。

阶段性作业

查看文档:

1. 如何添加子菜单？如在“保存”菜单中，又分为“保存为 txt 文件”和“保存为 word 文件”。

2. 如何在菜单项之间加分隔线？

3. javax.swing 包中，有一个 JRadioButtonMenuItem 类，该类如何使用？

4. javax.swing 包中，有一个 JCheckBoxMenuItem 类，该类如何使用？

15.3.5　使用 JOptionPane

用 JOptionPane 类，也可以显示窗口，此时，一般可以使用其显示一些消息框、输入框、确认框等。

打开文档，找到 javax.swing.JOptionPane 类，我们一般使用其如下静态函数：

Note

1. 显示消息框：

public static void showMessageDialog(Component parentComponent,
Object message) throws HeadlessException

参数 1 表示父组件(可以为空，也可以为一个 Component)，参数 2 表示消息内容。比如：

图 15-17

2. 显示输入框：

public static String showInputDialog(Object message)
throws HeadlessException

参数表示输入框上的提示信息。输入之后的内容以字符串返回。比如：

图 15-18

3. 显示确认框：

public static int showConfirmDialog(Component parentComponent,
Object message) throws HeadlessException

参数 1 表示父组件(可以为空，也可以为一个 Component)，参数 2 表示确认框上的提示信息。比如：

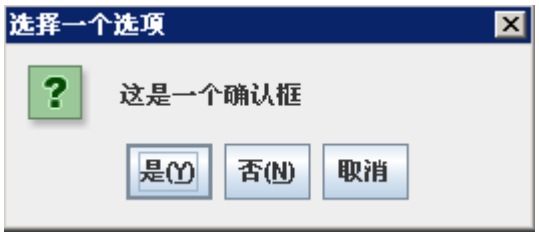


图 15-19

系统如何知道用户点了哪个按钮呢？答案是根据返回值来判断，返回值是一个整数，由 JOptionPane 类中定义的静态变量表达。比如，JOptionPane.YES_OPTION 表示点击了 YES 按钮，其他静态变量可以在文档中查到。

以下代码使用了这三种函数：

OptionPaneTest1.java

```
package window;
import javax.swing.JOptionPane;
public class OptionPaneTest1 {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "这是一个消息框");
        JOptionPane.showInputDialog("这是一个输入框");
        int result = JOptionPane.showConfirmDialog(null,"这是一个确认框");
    }
}
```

运行，显示 3 个窗口。

Note

阶段性作业

JOptionPane 类显示窗口时，可以以各种不同的风格显示消息框、输入框和确认框，请查文档，阅读相应的函数的描述。

15.3.6　其他控件

前面我们说过，我们不可能对所有的控件死记硬背，因此，希望大家遇到想要使用的控件，自己去查文档，从构造函数看起，然后学习它们的成员函数。

以下列出一些常见的其他控件。

1. javax.swing.JFileChooser。文件选择框，用于文件打开或保存，效果如下：

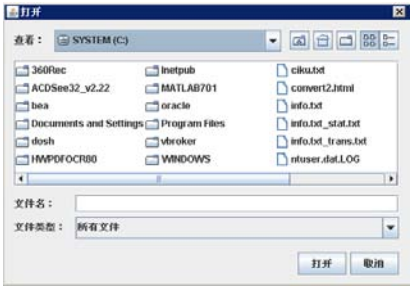


图 15-20

2. javax.swing.JColorChooser。颜色选择框，用于颜色的选择，效果如下：

图 15-21

3. javax.swing.JToolBar。用于在菜单条下方显示工具条，效果如下：

图 15-22

Note

4. javax.swing.JList。列表框，用于选择某些项目，效果如下：

图 15-23

5. javax.swing.JProgressBar。进度条，效果如下：

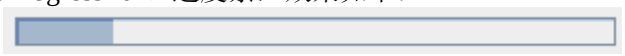


图 15-24

6. javax.swing.JSlider。滑块，用于设定某些数值，效果如下：

图 15-25

7. javax.swing.JTree。树形结构，效果如下：

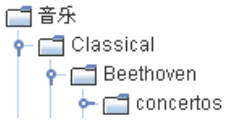


图 15-26

8. javax.swing.JTable。表格，效果如下：

名	姓
Mike	Albers
Mark	Andrews
Brian	Beck

图 15-27

9. javax.swing.JTabbedPane。选项卡，效果如下：

图 15-28

10. javax.swing.JInternalFrame。将窗口内容纳多个小窗口，效果如下：

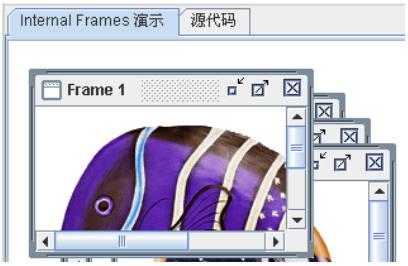


图 15-29

课外作业

结合文档和网上搜索，将此处列出的 10 种控件进行学习。

Note

15.4　颜色、字体和图像

15.4.1　如何使用颜色

GUI 编程中，颜色是我们经常要使用的内容。比如，将界面背景变为黄色，将按钮文字变为红色，等等。

在 Java 中，颜色是用 java.awt.Color 进行表达的。

打开文档，找到 java.awt.Color 类，最常见的构造函数是：

public Color(int r, int g, int b)

用红色、绿色和蓝色分量来初始化颜色。参数 r、g、b 必须在 0-255 之间。

小知识

我们生活中的任何颜色，都可以看做是红、绿、蓝三种颜色混合而成。如果三种颜色分量都为 0，则为黑色，都为 255，则为白色。

为了便于使用，在 Color 类中，提供了一些静态变量表示我们常见的颜色，如：Color.yellow 表示黄色，Color.red 表示红色，等等。

对于窗口和控件来说，我们可以设置两类颜色：

1. 设置背景颜色。

public void setBackground(Color c)

2. 设置前景颜色。

public void setForeground(Color c)

前景颜色主要是指控件上如文字等内容的颜色。

以下代码在界面上显示一个按钮，界面背景是黄色，按钮上的字是红色。代码如下：

ColorTest1.java

```
package color;
import java.awt.Color;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class ColorTest1 extends JFrame{
    private JButton jbt = new JButton("按钮");
    private JPanel jpl = new JPanel();
```

Note

```
        public ColorTest1(){
            jpl.add(jbt);
            this.add(jpl);
            jpl.setBackground(Color.yellow);
            jbt.setForeground(Color.red);
            this.setSize(100,80);
            this.setVisible(true);
        }
        public static void main(String[] args) {
            new ColorTest1();
        }
    }
```

运行，效果为：

图 15-30

阶段性作业

编写一个登录界面，含有文本框、密码框和登录按钮，要求界面背景和控件背景都是黄色，上面的字都是红色。

15.4.2 如何使用字体

GUI 编程中，字体也我们经常要使用的内容。比如，将文本框中的文字以一种醒目的字体显示。

在 Java 中，字体是用 java.awt.Font 进行表达的。打开文档，找到 java.awt.Font 类，最常见的构造函数是：

public Font(String name, int style, int size)

用字体名称、字体风格和字体大小初始化字体。

注意

1. 字体名称如果写错，则使用系统默认字体。在 Font 类中，也定义了一些静态变量表示系统提供的字体，如：Font.SANS_SERIF 等，具体可以参考文档。

2. 字体风格可以选用：Font.PLAIN(普通)，Font.BOLD(粗体)，Font.ITALIC(斜体)等，如果同时使用多种，则用“|”隔开，如：Font.BOLD|Font.ITALIC 表示粗斜体。

设置字体一般针对含有文字的控件，我们通过下面的方法设置字体：

public void setFont(Font f)

以下代码在界面上显示一个标签和一个文本框，标签字体为 20 号粗斜楷体，文

本框内的内容为 20 号斜黑体。代码如下：

FontTest1.java

```
package font;
import java.awt.Font;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class FontTest1 extends JFrame{
    private JLabel lblAcc = new JLabel("输入账号：");
    private JTextField tfAcc = new JTextField(10);
    private JPanel jpl = new JPanel();
    public FontTest1(){
        Font fontLblAcc =
            new Font("楷体_GB2312",Font.BOLD|Font.ITALIC,20);
        lblAcc.setFont(fontLblAcc);
        Font fontTfAcc = new Font("黑体",Font.ITALIC,20);
        tfAcc.setFont(fontTfAcc);
        jpl.add(lblAcc);
        jpl.add(tfAcc);
        this.add(jpl);
        this.setSize(250,80);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new FontTest1();
    }
}
```

Note

运行，效果为：

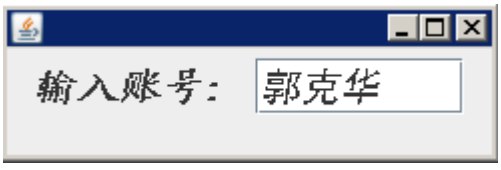


图 15-31

阶段性作业

1. 编写一个登录界面，含有文本框、密码框和登录按钮，要求界面背景和控件背景都是黄色，上面的字都是红色。字体都是 14 号楷体。

2. 在网上搜索：实例化字体对象时，如何精确确定该字体的名称？比如，“楷体_GB2312”，不能写成“楷体”。这个名称是如何确定的？

Note

15.4.3 如何使用图片

GUI 编程中，图片经常碰到。比如，我们通过在界面上画一幅图片，对界面进行美化。在 Java 中，图片的封装有两种方式：

1. 图像。

图像是用 java.awt.Image 来封装的。打开文档，找到 java.awt.Image 类，该类是一个抽象类，无法被实例化。

Image 对象一般使用如下方式得到：

```
Image img = Toolkit.getDefaultToolkit().createImage("图片路径");
```

Image 的使用，在界面画图中使用较多，后面的章节将有详细介绍。此处只介绍简单的功能。JFrame 有一个函数：

```
public void setIconImage(Image image)
```

通过该函数，可以设置此窗口要显示在最小化图标中的图像。

以下代码，将项目根目录下的 img.gif 设置为窗口的最小化图标。首先将该图片拷贝到项目根目录下，该图片内容如下：

图 15-32

代码如下：

ImageTest1.java

```
package image;
import java.awt.Image;
import java.awt.Toolkit;
import javax.swing.JFrame;
public class ImageTest1 extends JFrame{
    private Image img;
    public ImageTest1(){
        super("这是一个窗口");
        img = Toolkit.getDefaultToolkit().createImage("img.gif");
        this.setIconImage(img);
        this.setSize(250,80);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new ImageTest1();
```

```
    }
}
```

运行，效果为：

图 15-33

如果最小化，任务栏上显示为：

图 15-34

2. 图标。

图标是用 javax.swing.Icon 来封装的。打开文档，找到 java.awt.Icon，它是一个接口，无法被实例化。我们一般使用 Icon 的实现类 javax.swing.ImageIcon 来生成一个图标。

打开文档，找到 javax.swing.ImageIcon 类，最常见的构造函数是：

public ImageIcon(String filename)

传入一个路径，实例化 ImageIcon 对象。

设置图标，在 Swing 开发中非常常见。常见的控件，一般都提供了构造函数，传入一个图标。比如，JButton 类就具有如下构造函数：

public JButton(String text, Icon icon)

传入一个图标。此外，还有 setIcon 函数修改图标。

JLabel 等其他类也有相应的图标支持函数，请读者参考文档。

以下代码，将项目根目录下的 img.gif 设置为按钮的图标，代码如下：

ImageTest2.java

```
package image;
import javax.swing.*;
public class ImageTest2 extends JFrame{
    private Icon icon;
    private JButton jbt = new JButton("按钮");
    private JPanel jpl = new JPanel();
    public ImageTest2(){
        icon = new ImageIcon("img.gif");
        jbt.setIcon(icon);
        jpl.add(jbt);
        this.add(jpl);
        this.setSize(250,80);
```

```
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new ImageTest2();
    }
}
```

运行，效果为：

图 15-35

阶段性作业

1. 在界面上显示一个 JLabel 对象，JLabel 中含有一个图标。
2. 菜单项上也可以加图标，查看文档，看看如何实现？

15.5 几个有用的功能

15.5.1 如何设置界面的显示风格

前面我们编写的界面，风格似乎和 Windows 下的界面风格不太一致，能否让界面以某种操作系统的风格显示呢？

GUI 编程中，风格是由 javax.swing.UIManager 类来进行管理的。通过该类的如下函数来设置界面的显示风格：

```
public static void setLookAndFeel(String className)
                                    throws ClassNotFoundException,
                                           InstantiationException,
                                           IllegalAccessException,
                                           UnsupportedLookAndFeelException
```

我们可以用如下函数得到系统中已经支持的风格：

```
public static UIManager.LookAndFeelInfo[] getInstalledLookAndFeels()
```

以下代码，用系统支持的所有风格显示一个输入框，代码如下：

StyleTest.java

```
package others;
import javax.swing.*;
public class StyleTest {
```

Note

```
        public static void main(String[] args) {
            try{
                UIManager.LookAndFeelInfo[] infos =
                        UIManager.getInstalledLookAndFeels();
                for(UIManager.LookAndFeelInfo info:infos){
                    UIManager.setLookAndFeel(info.getClassName());
                    JOptionPane.showInputDialog(info.getName()+"风格");
                }
            }catch(Exception ex){}
        }
    }
```

运行，依次显示如下输入框：

图 15-36

15.5.2　如何获取屏幕大小

有时候，为了美观，我们希望在屏幕的正中央显示某个窗口，此时就必须事先知道屏幕的宽度和高度，才能对窗口的位置进行计算。如何知道屏幕的宽度和高度呢？

GUI 编程中，屏幕大小是由 java.awt.GraphicsEnvironment 类来获得的。下面的代码打印了当前的屏幕大小：

ScreenTest.java

```
package others;

import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;
public class ScreenTest {
    public static void main(String[] args) {
        GraphicsEnvironment ge =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        Rectangle rec =
            ge.getDefaultScreenDevice().getDefaultConfiguration().getBounds();
        System.out.println("屏幕宽度：" + rec.getWidth());
        System.out.println("屏幕高度：" + rec.getHeight());
    }
}
```

运行，控制台上打印如下：

```
屏幕宽度：1280.0
屏幕高度：800.0
```

图 15-37

Note

阶段性作业

编写一个界面，要求显示在屏幕中央。

15.5.3 如何用默认应用程序打开文件

JDK6.0 中增加了 java.awt.Desktop 类，该类最有意思的功能是用默认应用程序打开文件。比如，如果机器上装了 Acrobat，双击 pdf 文件，将会用 Acrobat 打开。此功能也可以用 Desktop 类实现。下面的代码打开 c:\test.pdf：

DesktopTest .java

```
package others;
import java.awt.Desktop;
import java.io.File;
public class DesktopTest    {
        public static void main(String[] args) throws Exception{
            Desktop.getDesktop().open(new File("C:\\test.pdf"));
        }
}
```

运行，自动打开这个文件，等价于双击这个文件。

15.5.4 如何将程序显示为系统托盘

JDK6.0 中增加了 java.awt.SystemTray 类，该类可以在任务栏上显示系统托盘，系统托盘用 java.awt.TrayIcon 封装。下面的代码将一个图片显示为系统托盘：

SystemTrayTest.java

```
package others;
import java.awt.Image;
import java.awt.SystemTray;
import java.awt.Toolkit;
import java.awt.TrayIcon;
public class SystemTrayTest{
        public static void main(String[] args) throws Exception {
            Image img = Toolkit.getDefaultToolkit().createImage("img.gif");
            TrayIcon ti = new TrayIcon(img);
            SystemTray.getSystemTray().add(ti);
```

```
        }
    }
```

运行，任务栏上显示效果如下：

图 15-38

在多媒体控制图标左边即为系统托盘，点击该图标，没有任何反应。这是因为我们还没有给其增加事件功能。

Note

本章知识体系

知识点	重要等级	难度等级
Swing 基本概念	★★★★	★★
窗口开发	★★★★	★★★
控件开发	★★★★	★★★★
颜色	★★★	★★
字体	★★	★★
图片	★★	★★★
其他功能	★★	★★★★

第16章

Java 界面布局管理

GUI 上控件的布局，能够让我们更好地控制界面的开发。本章将讲解几种最常见的布局：FlowLayout、GridLayout、BorderLayout、空布局以及其他一些比较复杂的布局方式。最后，用一个计算器程序将其进行了总结。

本章术语

布局 _____

FlowLayout _____

GridLayout _____

BorderLayout _____

空布局 _____

CardLayout _____

BoxLayout _____

GridBagLayout _____

16.1 认识布局管理

Note

16.1.1 为什么需要布局管理

在 JavaGUI 开发中，窗体上都需要添加若干控件。一般情况下是首先将控件加到面板上，然后加到窗体上。这样，控件在窗体上的排布就有一个方式，以下面的例子为例：

LayoutTest1.java

```
package layout;
import javax.swing.*;
public class LayoutTest1 extends JFrame{
    private JLabel lblInfo = new JLabel("这是注册窗口");
    private JButton btReg = new JButton("注册");
    private JPanel jpl = new JPanel();
    public LayoutTest1(){
        jpl.add(lblInfo);
        jpl.add(btReg);
        this.add(jpl);
        this.setSize(150,100);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new LayoutTest1();
    }
}
```

运行，效果为：

图 16-1

为什么控件会这样排布呢？这是 JPanel 默认的排布方式。这种排布方式可能有一些问题，比如，窗口改变大小，排布方式改变：

图 16-2

又如，如果我们通过 setSize 方法改变了按钮的大小，但是在界面上体现不出来。

因此，如果我们不使用较为科学的布局方式，界面在用户使用时，可能出现不同的样子。此时，就需要使用布局管理器。

布局管理器，可以让我们将加入到容器的组件按照一定的顺序和规则放置，使之看起来更美观。

在 Java 中，布局由布局管理器：java.awt.LayoutManager 来管理。

Note

16.1.2 认识 LayoutManager

打开文档，找到 java.awt.LayoutManager，会发现这是一个接口，并不能直接实例化。此时，我们可以使用该接口的实现类。

java.awt.LayoutManager 最常见的实现类有：

1. java.awt.FlowLayout：将组件按从左到右而后从上到下的顺序依次排列，一行放不下则到下一行继续放置。

2. java.awt.GridLayout：将界面布局为一个无框线的表格，每个单元格中放一个组件。

3. java.awt.BorderLayout：将组件按东、南、西、北、中五个区域放置，每个方向最多只能放置一个组件。

等等。

如何设置容器的布局方式？我们可以使用 JFrame、JPanel 等容器的如下函数：

public void setLayout(LayoutManager mgr)

注意

1. 在大多数情况下，综合运用好这些常见的布局管理器已可以满足需要。对于特殊的具体应用，可以通过实现 LayoutManager 或 LayoutManager2 接口来定义自己的布局管理器。

2. 以上几种常见的布局方式，组件的大小、位置都不能用 setSize 和 setLocation 方法确定，而是根据窗体大小自动适应。如果需要用 setSize 和 setLocation 方法确定组件的大小和位置，则可以采用空布局(null 布局)。

3. 应该指出，Java 中的布局方式还有很多，要想全部讲解，是不可能的，我们只能讲解最常见的几种，其他内容，大家举一反三，通过文档和网络，可以很容易学会。比如:

(1) java.awt.CardLayout: 将组件象卡片一样放置在容器中。

(2) java.awt GridBagLayout: 可指定组件放置的具体位置及占用单元格数目。

(3) javax.swing BoxLayout: 就像整齐放置的一行或者一列盒子，每个盒子中一个

组件。

此外，还有 javax.swing.SpringLayout 、 javax.swing.ScrollPaneLayout 、javax.swing.OverlayLayout、javax.swing.ViewportLayout 等。

Note

16.2 使用 FlowLayout

16.2.1 什么是 FlowLayout

FlowLayout 是最常见的布局方式，它的特点是：将组件按从左到右而后从上到下的顺序依次排列，一行放不下则到下一行继续放置。

上节中的代码，所展现的就是 FlowLayout。由此可见，JPanel 的默认布局方式就是 FlowLayout。

注意

FlowLayout 也称“流式布局”，非常形象，像流水一样，一个方向流不过去就会拐弯，控件在一行放不下就到下一行。

我们使用 java.awt.FlowLayout 类来进行流式布局的管理。打开文档，找到 java.awt.FlowLayout 类，其构造函数是：

1. public FlowLayout()：实例化 FlowLayout 对象，布局方式为居中对齐，默认的控件之间水平和垂直间隔是 5 个单位(一般为像素)。

2. public FlowLayout(int align)：实例化 FlowLayout 对象，默认的水平和垂直间隔是 5 个单位。

align 表示指定的对齐方式，常见的选择如下：

(1)FlowLayout.LEFT：左对齐。

(2)FlowLayout.RIGHT：右对齐。

(3)FlowLayout.CENTER：居中对齐。

3. public FlowLayout(int align, int hgap, int vgap)：不仅指定对齐方式，而且指定控件之间水平和垂直间隔。

16.2.2 如何使用 FlowLayout

以下案例，使用 FlowLayout 开发一个登录界面，控件居中对齐，水平和垂直间隔为 10 个像素。代码如下：

FlowLayoutTest1.java

```
package flowlayout;
import java.awt.FlowLayout;
import javax.swing.*;
public class FlowLayoutTest1 extends JFrame{
```

Note

```
    private FlowLayout flowLayout =
            new FlowLayout(FlowLayout.CENTER,10,10);
    private JLabel lblAcc = new JLabel("输入账号");
    private JTextField tfAcc = new JTextField(10);
    private JLabel lblPass = new JLabel("输入密码");
    private JPasswordField pfPass = new JPasswordField(10);
    private JButton btLogin = new JButton("登录");
    private JButton btExit = new JButton("取消");
    private JPanel jpl = new JPanel();
    public FlowLayoutTest1(){
        jpl.setLayout(flowLayout);//设置布局方式
        jpl.add(lblAcc);
        jpl.add(tfAcc);
        jpl.add(lblPass);
        jpl.add(pfPass);
        jpl.add(btLogin);
        jpl.add(btExit);
        this.add(jpl);
        this.setSize(200,150);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new FlowLayoutTest1();
    }
}
```

运行，效果为：

图 16-3

但是，这只是界面大小经过精心调整的结果，如果界面调整大小，效果为：

图 16-4

这说明 FlowLayout 的使用功能有些限制。

阶段性作业

1. 我们也可以设置让界面的大小不可改变，来使得 FlowLayout 变得更加实用。查询文档，如何让一个 JFrame 的大小固定，不可改变呢？

2. 在网上查询，JFrame 的默认布局是什么？

16.3 使用 GridLayout

16.3.1 什么是 GridLayout

GridLayout 也是比较常见的布局方式，它的特点是：将界面布局为一个无框线的表格，每个单元格中放一个组件。一行一行放置，一行放满，放下一行。

我们常见的计算器上的按钮，就可以采用这个布局。

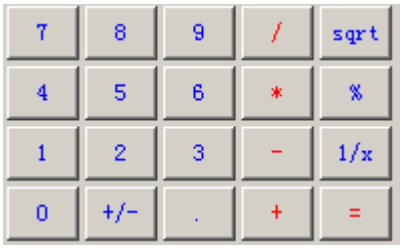


图 16-5

面板分为 4 行 5 列，放置一些按钮。

注意

GridLayout 也称“网格布局”。

我们使用 java.awt.GridLayout 类来进行网格布局的管理。打开文档，找到 java.awt.GridLayout 类，最常见的构造函数是：

1. public GridLayout(int rows, int cols)：创建具有指定行数和列数的网格布局，给布局中的所有组件分配相等的大小。默认情况下，行列之间没有边距。

2. public GridLayout(int rows, int cols, int hgap, int vgap)：创建具有指定行数和列数的网格布局，给布局中的所有组件分配相等的大小，此外，将水平和垂直间距设置为指定值。水平间距将置于列与列之间，垂直间距将置于行与行之间。

16.3.2 如何使用 GridLayout

以下案例，使用 GridLayout 开发一个登录界面，水平和垂直间隔为 10 个像素。代码如下：

Note

GridLayoutTest1.java

```
package gridlayout;
import java.awt.GridLayout;
import javax.swing.*;
public class GridLayoutTest1 extends JFrame{
    private GridLayout gridLayout = new GridLayout(3,2,10,10);
    private JLabel lblAcc = new JLabel("输入账号");
    private JTextField tfAcc = new JTextField(10);
    private JLabel lblPass = new JLabel("输入密码");
    private JPasswordField pfPass = new JPasswordField(10);
    private JButton btLogin = new JButton("登录");
    private JButton btExit = new JButton("取消");
    private JPanel jpl = new JPanel();
    public GridLayoutTest1(){
        jpl.setLayout(gridLayout);//设置布局方式
        jpl.add(lblAcc);
        jpl.add(tfAcc);
        jpl.add(lblPass);
        jpl.add(pfPass);
        jpl.add(btLogin);
        jpl.add(btExit);
        this.add(jpl);
        this.setSize(200,150);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new GridLayoutTest1();
    }
}
```

运行，效果为：

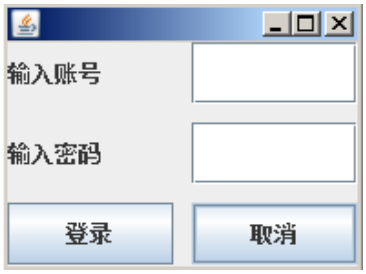


图 16-6

虽然界面难看了点，但是，如果界面调整大小，效果为：

图 16-7

这说明 GridLayout 至少可以保证界面不变形。

阶段性作业

开发一个国际象棋棋盘，界面如下：

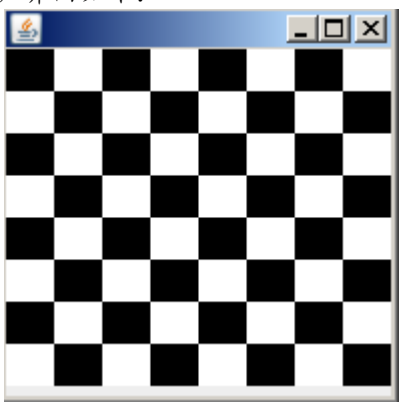


图 16-8

16.4 使用 BorderLayout

16.4.1 什么是 BorderLayout

BorderLayout 也是比较常见的布局方式，它的特点是：将组件按东、南、西、北、中五个区域放置，每个方向最多只能放置一个组件。

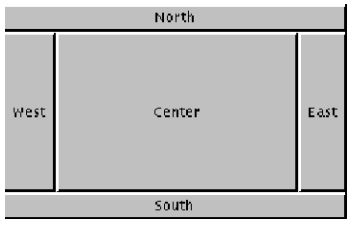


图 16-9

注意

读者可能会问，这种布局方式有什么用呢？实际上，我们常见的软件界面，很多都是用这种布局，比如：

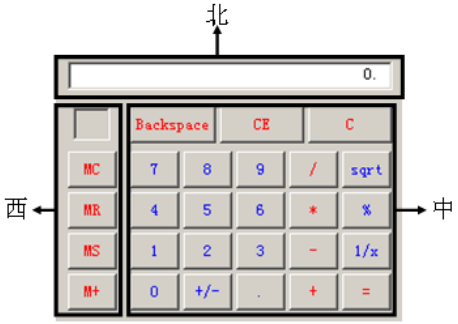


图 16-10

计算器程序的界面上，大致可以分为北、西、中三个部分，每个部分可以是一个面板。其中，“中”部分实际上还可以细分。

注意

1. BorderLayout 也称“边界布局”。

2. 如果东西南北某个部分没有添加任何内容，则其他内容会自动将其填满；如果中间没有添加任何内容，则会空着。

我们使用 java.awt.BorderLayout 类来进行边界布局的管理。打开文档，找到 java.awt.BorderLayout 类，最常见的构造函数是：

1. public BorderLayout()：构造一个组件之间没有间距的边界布局。

2. public BorderLayout(int hgap, int vgap)：构造一个具有指定组件间距的边界布局。水平间距由 hgap 指定，垂直间距由 vgap 指定。

不过，使用了边界布局之后，将组件加到容器上去，就不能直接用 add 函数了，还必须指定加到哪个位置。一般我们可以在 add 函数的第二个参数指定添加的位置，可以选择：

1. BorderLayout.NORTH：表示加到北边。

2. BorderLayout.SOUTH：表示加到南边。

3. BorderLayout.EAST：表示加到东边。

4. BorderLayout.WEST：表示加到西边。

5. BorderLayout.CENTER：表示加到中间。

比如，下面的代码就是将一个按钮添加到面板南边。

```
JPanel p = new JPanel();
p.setLayout(new BorderLayout());
p.add(new JButton("Okay"), BorderLayout.SOUTH);
```

Note

16.4.2 如何使用 BorderLayout

以下案例，使用 BorderLayout 开发一个很简单的界面，在东、西、南边各添加一个按钮。代码如下：

BorderLayoutTest1.java

```
package borderlayout;
import java.awt.BorderLayout;
import javax.*;
public class BorderLayoutTest1 extends JFrame{
    private BorderLayout borderLayout = new BorderLayout();
    private JButton btEast = new JButton("东");
    private JButton btWest = new JButton("西");
    private JButton btSouth = new JButton("南");
    private JPanel jpl = new JPanel();
    public BorderLayoutTest1(){
        jpl.setLayout(borderLayout);
        jpl.add(btEast,BorderLayout.EAST);
        jpl.add(btWest,BorderLayout.WEST);
        jpl.add(btSouth,BorderLayout.SOUTH);
        this.add(jpl);
        this.setSize(200,150);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new BorderLayoutTest1();
    }
}
```

运行，效果为：

图 16-11

16.5　一个综合案例：计算器

Note

16.5.1　案例需求

本节将制作一个简单的计算器界面，效果如下：

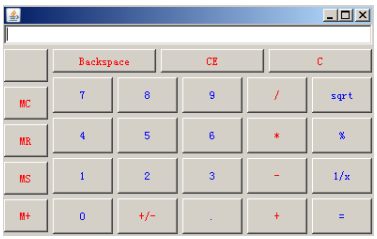


图 16-12

当然，这个界面还是没有 Window 中的计算器界面漂亮，不过，做界面也不是 Java 的强项。

16.5.2　关键技术

1. 面板的组织。

我们学习了各个布局，这里进行分析一下：总体来说，界面是个边界布局，分为北、西、中三个部分，方法如下：

图 16-13

其中，pn 中包括一个文本框，pw 中包括一个面板，分为 5 行 1 列，包含 5 个按钮。pc 又分为 2 个部分：

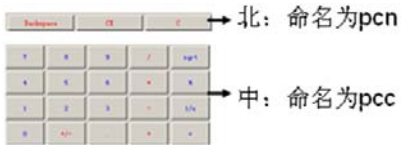


图 16-14

其中，pcn 中包括一个面板，分为 1 行 3 列，包含 3 个按钮，pcc 中包括一个面板，分为 4 行 5 列，包含 20 个按钮。

因此，各个面板的生成可以写成单独的函数。

2. 按钮的生成。

本界面中要生成很多按钮，如果一个个实例化，比较麻烦。我们可以使用循环，具体可见后面的程序代码。

Note

16.5.3 代码编写

本案例代码如下：

Calc.java

```
package calc;
import java.awt.*;
import javax.swing.*;
public class Calc extends JFrame{
    //北边的文本框
    public JPanel createPN() {
        JPanel pn = new JPanel();
        pn.setLayout(new BorderLayout(5,5));
        JTextField tfNumber = new JTextField();
        pn.add(tfNumber,BorderLayout.CENTER);
        return pn;
    }
    //西边的 5 个按钮
    public JPanel createPW() {
        JPanel pw = new JPanel();
        pw.setLayout(new GridLayout(5,1,5,5));
        JButton[] jbts = new JButton[5];
        String[] labels = new String[]{"","MC","MR","MS","M+"};
        for(int i=0;i<jbts.length;i++){
            JButton jbt = new JButton(labels[i]);
            jbt.setForeground(Color.red);
            pw.add(jbt);
        }
        return pw;
    }
    //中间面板
    public JPanel createPC() {
        JPanel pc = new JPanel();
        pc.setLayout(new BorderLayout(5,5));
        pc.add(createPCN(),BorderLayout.NORTH);
```

Note

```
        pc.add(createPCC(),BorderLayout.CENTER);
        return pc;
    }
    //中间面板中的北边 3 个按钮
    public JPanel createPCN() {
        JPanel pcn = new JPanel();
        pcn.setLayout(new GridLayout(1,3,5,5));
        JButton[] jbts = new JButton[3];
        String[] labels = new String[]{"Backspace","CE","C"};
        for(int i=0;i<jbts.length;i++){
            JButton jbt = new JButton(labels[i]);
            jbt.setForeground(Color.red);
            pcn.add(jbt);
        }
        return pcn;
    }
    //中间面板中的中间 20 个按钮
    public JPanel createPCC() {
        JPanel pcc = new JPanel();
        pcc.setLayout(new GridLayout(4,5,5,5));
        JButton[] jbts = new JButton[20];
        String[] labels = new String[]{"7","8","9","/","sqrt",
                                       "4","5","6","*","%",
                                       "1","2","3","-","1/x",
                                       "0","+/-",".","+","="};
        for(int i=0;i<jbts.length;i++){
            JButton jbt = new JButton(labels[i]);
            if(labels[i].endsWith("+")||labels[i].endsWith("-")||
                labels[i].endsWith("*")||labels[i].endsWith("/")){
                jbt.setForeground(Color.red);
            }else{
                jbt.setForeground(Color.BLUE);
            }
            pcc.add(jbt);
        }
        return pcc;
    }
```

Note

```
        //构造函数
        public Calc(){
            this.setLayout(new BorderLayout(5,5));
            this.add(createPN(),BorderLayout.NORTH);
            this.add(createPW(),BorderLayout.WEST);
            this.add(createPC(),BorderLayout.CENTER);
            this.setSize(400,250);
            this.setVisible(true);
        }
        public static void main(String[] args)throws Exception {
            //使用 Windows 风格
            String win="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
            UIManager.setLookAndFeel(win);
            Calc calcFrm = new Calc();
        }
    }
```

运行，即得到相应效果。

阶段性作业

思考：在前面的代码中，createPW 函数、createPCN 函数、createPCC 函数中含有大量重复代码，你能否想出办法解决或者部分解决这个问题？

16.6 使用空布局

16.6.1 什么是空布局

空布局实际上不算一种单独的布局种类，只是表示我们不在容器中使用任何布局。由于一般情况下，容器都有个默认布局(比如，JPanel 的默认布局是 FlowLayout)，因为，我们不在容器中使用任何布局，需要显示调用函数：setLayout(null)。

空布局有什么作用呢？我们知道，前面我们使用了布局，控件的大小和位置是随着界面的变化而变化的，我们不能通过 setSize 方法设置大小，也不能通过 setLocation 方法设置位置。但是，使用了空布局之后，就可以实现这个功能。

16.6.2 如何使用空布局

以下案例，使用空布局在界面上放置几个按钮，代码如下：

NullLayoutTest1.java

Note

```
package nulllayout;
import javax.swing.*;
public class NullLayoutTest1 extends JFrame{
    private JButton bt1 = new JButton("按钮 1");
    private JButton bt2 = new JButton("按钮 2");
    private JButton bt3 = new JButton("按钮 3");
    private JPanel jpl = new JPanel();
    public NullLayoutTest1(){
        jpl.setLayout(null);//设置空布局
        bt1.setSize(100,25);
        bt1.setLocation(10,20);
        jpl.add(bt1);
        bt2.setSize(80,40);
        bt2.setLocation(30,60);
        jpl.add(bt2);
        bt3.setSize(70,25);
        bt3.setLocation(15,45);
        jpl.add(bt3);

        this.add(jpl);
        this.setSize(200,150);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new NullLayoutTest1();
    }
}
```

运行，效果为：

图 16-15

注意

1. 在本例中，setLocation 方法实际上设置了按钮左上角距界面左上角的横、纵方向的距离。

2. 空布局虽然让我们很容易地进行界面开发，但是也要谨慎使用。由于在不同系统下的坐标概念不一定相同，纯粹用坐标来定义大小和位置，可能会产生不同的效果。

Note

阶段性作业

用空布局，结合多线程完成：界面上有一个包含图标的 JLabel，从界面顶部掉下来。

本章知识体系

知识点	重要等级	难度等级
布局基本概念	★★★★	★★
FlowLayout	★★★★	★★
GridLayout	★★★	★★
BorderLayout	★★★	★★
空布局	★★★	★★

第17章

Java 事件处理

JavaGUI 的事件，能够让我们真正完善程序的功能。本章将首先讲解事件的基本原理，然后讲解事件的开发流程，最后讲解几种最常见的事件的处理：ActionEvent、FocusEvent、KeyEvent、MouseEvent、WindowEvent，最后讲解用 Adapter 简化事件的开发。

本章术语

Event _____

Listener _____

ActionEvent _____

FocusEvent _____

KeyEvent _____

MouseEvent _____

WindowEvent _____

Adapter _____

17.1 认识事件处理

Note

17.1.1 什么是事件

在前面的程序中，我们可以在窗体上添加若干控件。比如，添加按钮，以下面的例子为例：

EventTest1.java

```
package event;
import javax.swing.*;
public class EventTest1 extends JFrame{
    private JButton btHello = new JButton("Hello");
    public EventTest1(){
        this.add(btHello);
        this.setSize(30,50);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new EventTest1();
    }
}
```

运行，效果为：

图 17-1

界面上有一个按钮。但是，当点击按钮时，却没有任何反应。显然，真正丰富多彩的程序中，我们点击按钮，至少需要能做点事情。比如，点击按钮，在控制台上打印一个字符串：Hello。该功能如何实现呢？这就需要使用本章讲解的事件处理。

什么是事件？简单讲，事件是指用户为了交互而产生的键盘和鼠标动作。比如，点击按钮，就可以认为发出了一个“按钮点击事件”。

注意

1. 以上事件的定义，不太严谨，只是最直观的说法。实际上，事件不一定在用户交互时产生。比如，程序运行出了异常，也可以认为是一个事件。

2. 事件是有种类的。比如，按钮点击，是一种事件；鼠标在界面上移动，也是一种事件；等等。要处理某事件，首先必须搞清楚事件的种类。后面的篇幅我们有详细讲解。

17.1.2 事件处理代码的编写

Note

在了解事件处理之前，我们先举生活中的一个例子：

在生活中，也会出现很多出现“事件”的场合。比如，上课铃响了，小王听到了铃声，走进教室。

在这个事件中，上课铃响，相当于发出了一个事件，类似于上节的“点击按钮”，小王走进教室，相当于处理这个事件，类似于上节的“打印 Hello”。

实际上，这个看似简单的日常生活例子，其顺利执行的条件并不简单，至少需要以下条件：

1. 铃声必须由响铃的地方传到小王耳朵里，因此，铃声必须进行封装。
2. 小王必须长着耳朵，否则他听不到，铃声再响也是徒劳。
3. 小王必须执行“走进教室”这个动作，否则相当于事件没有处理。
4. 必须规定，上课铃响，让小王进教室。如果没有这个规定，小王如何知道要走进教室？

我们来进行类比，实际上，上面的几个条件可以解释如下：

1. 事件必须用一个对象封装。
2. 事件的处理者必须具有监听事件的能力。
3. 事件的处理者必须编写事件处理函数。
4. 必须将事件的发出者和事件的处理者对象绑定起来。

这四个步骤，就是我们编写事件代码的依据。这里，我们来实现“点击按钮，打印 Hello”。

1. 事件必须用一个对象封装。

点击按钮，系统自动将发出的事件封装在 java.awt.event.ActionEvent 对象内。

问答

问：如何知道某个事件封装在什么样的对象内？

答：由于事件是有种类的，因此，不同的事件封装在不同的对象内，在后面的章节中，我们进行了详细的总结。此处只要知道点击按钮，发出的事件封装在 java.awt.event.ActionEvent 对象内即可。

2. 事件的处理者必须具有监听事件的能力。

在 java 中，ActionEvent 是由 java.awt.event.ActionListener 监听的。

因此，此步骤中，我们需要编写一个事件处理类，来实现 ActionListener 接口。

```
class ButtonClickOpe implements ActionListener{
    //处理事件
}
```

3. 事件的处理者必须编写事件处理函数。

在 java 中，实现一个接口，必须将接口中的函数重写一遍，该函数就是事件处理函数。查看文档，可以找到 ActionListener 中定义的函数：

Note

void actionPerformed(ActionEvent e)

对其进行重写并编写事件处理代码：

```
class ButtonClickOpe implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        System.out.println("Hello");
    }
}
```

注意

actionPerformed 函数中，参数 e 就封装了发出的事件。通过参数 e 的 getSource() 方法，可以知道事件是由谁发出的。后面我们将会用到。

4. 必须将事件的发出者和事件的处理者对象绑定起来。

该步骤中，必须规定按钮点击，发出的事件由 ButtonClickOpe 对象处理。方法是调用按钮的如下函数：

public void addActionListener(ActionListener l)

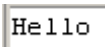
因此，整个代码就可以写成：

EventTest2.java

```
package event;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
public class EventTest2 extends JFrame{
    private JButton btHello = new JButton("按钮");
    public EventTest2(){
        this.add(btHello);
        //绑定
        btHello.addActionListener(new ButtonClickOpe());
        this.setSize(30,50);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new EventTest2();
    }
}
class ButtonClickOpe implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        System.out.println("Hello");
```

```
    }
}
```

运行，点击按钮，控制台上打印：



说明事件成功处理。

从本例可以看出，事件处理，关键是要弄清要处理什么样的事件，其他按照上面的流程编程即可。

Note

阶段性作业

已知：在 JTextField 中输入回车，发出的也是 ActionEvent，请编写：

在 JFrame 上放置一个文本框，文本框中输入一个数字，回车，在控制台上打印该文本框中数值的平方。

17.1.3 另外几种编程风格

前一章的例子中，我们需要编写两个类，实际上，有很多方法可以进行简化。比如，我们可以使用匿名处理对象的方法实现前一节的例子：

EventTest3.java

```
package event;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
public class EventTest3 extends JFrame{
    private JButton btHello = new JButton("按钮");
    public EventTest3(){
        this.add(btHello);
        //绑定
        btHello.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                System.out.println("Hello");
            }
        });
        this.setSize(30,50);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new EventTest3();
```

```
        }
    }
```

运行，点击按钮，打印“Hello”。不过，这种方法使用不多。

Note

由于一个 Java 类可以实现多个接口，我们通常用界面类直接实现接口的方法进行简化。本例中实现两个按钮，点击登录按钮，打印“登录”，点击退出按钮，程序退出。

EventTest4.java

```
package event;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
public class EventTest4 extends JFrame implements ActionListener{
    private JButton btLogin = new JButton("登录");
    private JButton btExit = new JButton("退出");
    public EventTest4(){
        this.setLayout(new FlowLayout());
        this.add(btLogin);
        this.add(btExit);
        //绑定
        btLogin.addActionListener(this);
        btExit.addActionListener(this);
        this.setSize(100,100);
        this.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btLogin){
            System.out.println("登录");
        }else{
            System.exit(0);
        }
    }
    public static void main(String[] args) {
        new EventTest4();
    }
}
```

运行，效果如下：

图 17-2

点击登录按钮，控制台打印：

图 17-3

点击退出按钮，程序退出。

注意

此处使用 e.getSource()判断事件是由谁发出的。

17.2 处理 ActionEvent

17.2.1 什么情况发出 ActionEvent

在 java.awt.event 包中，ActionEvent 是最常用的一种事件。一般情况下，ActionEvent 适合于对某些控件的单击(也有特殊情况)。常见发出 ActionEvent 的场合如下：

1. JButton、JComboBox、JMenu、JMenuItem、JCheckBox、JRadioButton等控件的的单击。

2. javax.swing.Timer 发出的事件。

3. 在 JTextField 等控件上按下回车、JButton 等控件上按下空格(相当于点击效果)等。

等等。ActionEvent 用 ActionListener 监听。其编程方法，采用上节流程即可。

17.2.2 使用 ActionEvent 解决实际问题

以下案例中，界面上包含一个下拉列表框，选择界面颜色，当选择之后，能够将界面背景自动变成相应颜色。代码如下：

ActionEventTest1.java

```
package actionevent;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
```

Note

```
import java.awt.event.ActionListener;
import javax.swing.JComboBox;
import javax.swing.JFrame;
public class ActionEventTest1 extends JFrame implements ActionListener{
    private JComboBox cbColor = new JComboBox();
    public ActionEventTest1(){
        this.add(cbColor,BorderLayout.NORTH);
        cbColor.addItem("红");
        cbColor.addItem("绿");
        cbColor.addItem("蓝");
        cbColor.addActionListener(this);
        this.setSize(30,100);
        this.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        Object color = cbColor.getSelectedItem();
        if(color.equals("红")){
            this.getContentPane().setBackground(Color.red);
        }else if(color.equals("绿")){
            this.getContentPane().setBackground(Color.green);
        }else{
            this.getContentPane().setBackground(Color.blue);
        }
    }
    public static void main(String[] args) {
        new ActionEventTest1();
    }
}
```

运行，效果为：

图 17-4

选择某种颜色，可以将界面背景变成相应颜色：

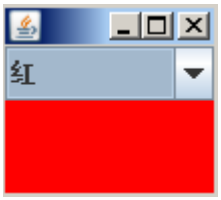


图 17-5

Note

注意

此代码中，我们改变的是 JFrame 的颜色，以变为红色为例，使用的代码段是：this.getContentPane().setBackground(Color.red);，改变颜色，必须得到 JFrame 上的 ContentPane，而不能直接使用 this.setBackground(Color.red);。

阶段性作业

1. 将上例中的下拉列表框改为单选按钮，选择颜色，能够将界面背景变成相应颜色。

2. 编写一个 JFrame 界面，上面含有一个菜单(JMenu)：打开文件。点击该菜单，出现文件选择框(JFileChooser)，选择一个文本文件，能够将文件内容显示在界面上的多行文本框(JTextArea)内。

3. 在 Swing 中，提供了一个定时器类 javax.swing.Timer，可以每隔一段时间执行一段代码。javax.swing.Timer 的构造函数为：

public Timer(int delay, ActionListener listener)

表示每隔一段时间(毫秒)，触发 ActionListener 内的处理代码。在实例化 Timer 对象之后，可以用 start()函数让其启动，可以用 stop()函数让其停止。

用 Timer 完成：界面上有一个按钮，从左边飞到右边。

4. 前面的章节中学习过，java.awt.SystemTray 可以向任务栏上添加一个托盘图标，托盘图标用 java.awt.TrayIcon 封装，但是将 TrayIcon 添加到任务栏上后，点击托盘图标，却没有任何反应。查询文档，实现：点击托盘图标，能够显示一个 JFrame 界面。

17.3 处理 FocusEvent

17.3.1 什么情况发出 FocusEvent

在 java.awt.event 包中，FocusEvent 也经常使用。一般情况下，FocusEvent 适合于对某些控件 Component 获得或失去输入焦点时，需要处理的场合。FocusEvent 用 java.awt.event.FocusListener 接口监听。该接口中有如下函数：

1. void focusGained(FocusEvent e)：组件获得焦点时调用。

2. void focusLost(FocusEvent e)：组件失去焦点时调用。

17.3.2 使用 FocusEvent 解决实际问题

以下案例中，界面上有一个文本框，要求，该文本框失去焦点时，内部显示：请您输入账号；当得到焦点时，该提示消失。代码如下：

Note

FocusEventTest1.java

```
package focusevent;
import java.awt.FlowLayout;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;
public class FocusEventTest1 extends JFrame implements FocusListener{
    private JButton btOK = new JButton("确定");
    private JTextField tfAcc = new JTextField("请您输入账号",10);
    public FocusEventTest1(){
        this.setLayout(new FlowLayout());
        this.add(btOK);
        this.add(tfAcc);
        tfAcc.addFocusListener(this);//绑定
        this.setSize(200,80);
        this.setVisible(true);
    }
    public void focusGained(FocusEvent arg0) {
        tfAcc.setText("");
    }
    public void focusLost(FocusEvent arg0) {
        tfAcc.setText("请您输入账号");
    }
    public static void main(String[] args) {
        new FocusEventTest1();
    }
}
```

运行，效果为：

图 17-6

鼠标移动到文本框中，效果如下：

图 17-7

Note

阶段性作业

在界面上放 2 个按钮: 登录和退出，焦点到达某个按钮上，该按钮背景变为黄色，文字变为红色。如果失去焦点，就显示为一个普通按钮。

17.4 处理 KeyEvent

17.4.1 什么情况发出 KeyEvent

在 java.awt.event 包中，KeyEvent 也经常使用。一般情况下，KeyEvent 适合于在某个控件上进行键盘操作时，需要处理事件的场合。KeyEvent 用 java.awt.event.KeyListener 接口监听。该接口中有如下函数：

1. void keyTyped(KeyEvent e)：键入某个键时调用此方法。
2. void keyPressed(KeyEvent e)：按下某个键时调用此方法。
3. void keyReleased(KeyEvent e)：释放某个键时调用此方法。

17.4.2 使用 KeyEvent 解决实际问题

下面用一个程序进行测试，在一个 JFrame 上敲击按下键盘，释放，打印按键的内容。代码如下：

KeyEventTest1.java

```
package keyevent;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JFrame;
public class KeyEventTest1 extends JFrame implements KeyListener{
    public KeyEventTest1(){
        this.addKeyListener(this);
        this.setSize(200,80);
        this.setVisible(true);
    }
```

Note

```
    public void keyPressed(KeyEvent e) {
        System.out.println(e.getKeyChar() + "按下");
    }
    public void keyReleased(KeyEvent e) {
        System.out.println(e.getKeyChar() + "释放");
    }
    public void keyTyped(KeyEvent e) {
        System.out.println(e.getKeyChar() + "敲击");
    }
    public static void main(String[] args) {
        new KeyEventTest1();
    }
}
```

运行，效果为：

图 17-8

如果按下键盘上的 a 键释放，控制台打印：

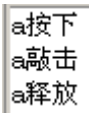


图 17-9

注意

1. 键盘事件，一定要在发出事件的控件已经获取焦点的情况下才能使用。比如，如果我们在 JFrame 上增加一个按钮，此时按钮获取了焦点，JFrame 的键盘事件就不会触发了，此时除非给按钮增加键盘事件。

2. KeyEvent 类中，封装了键的信息。主要函数有如下几个:

(1) public char getKeyChar():获取键的字符。

(2) public int getKeyCode(): 获取键对应的代码。代码可在文档 java.awt.event.KeyEvent 中查找，用静态变量表示，比如，左键对应的是: KeyEvent.VK_LEFT，左括弧对应的是: KeyEvent VK_LEFT_PARENTHESIS; 等等。

3. 键盘事件在游戏开发时经常用到，我们将在后面的篇幅中讲解。

阶段性作业

使用键盘事件完成: 界面上有一个含有卡通图标的 JLabel，可以通过键盘上的上下左右键控制其移动。

17.5 处理 MouseEvent

17.5.1 什么情况发出 MouseEvent

Note

在 java.awt.event 包中，MouseEvent 也经常使用。一般情况下，MouseEvent 在以下情况下发生：

1. 鼠标事件。

鼠标事件包括按下鼠标按键、释放鼠标按键、单击鼠标按键（按下并释放）、鼠标光标进入组件几何形状的未遮掩部分、鼠标光标离开组件几何形状的未遮掩部分。此时，MouseEvent 用 java.awt.event.MouseListener 接口监听。该接口中有如下函数：

(1) void mouseClicked(MouseEvent e)：鼠标按键在组件上单击（按下并释放）时调用。

(2) void mousePressed(MouseEvent e)：鼠标按键在组件上按下时调用。

(3) void mouseReleased(MouseEvent e)：鼠标按钮在组件上释放时调用。

(4) void mouseEntered(MouseEvent e)：鼠标进入到组件上时调用。

(5) void mouseExited(MouseEvent e)：鼠标离开组件时调用。

2. 鼠标移动事件

鼠标移动事件，包括移动鼠标和拖动鼠标。此时，MouseEvent 用 java.awt.event.MouseMotionListener 接口监听。该接口中有如下函数：

(1) void mouseDragged(MouseEvent e)：鼠标拖动时调用。

(2) void mouseMoved(MouseEvent e)：鼠标移动时调用。

17.5.2 使用 MouseEvent 解决实际问题

下面用一个程序进行鼠标事件测试，鼠标在 JFrame 上按下，将该处坐标设置为界面标题。代码如下：

MouseEventTest1.java

```
package mouseevent;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JFrame;
public class MouseEventTest1 extends JFrame implements MouseListener{
    public MouseEventTest1(){
        this.addMouseListener(this);
        this.setSize(300,100);
        this.setVisible(true);
    }
```

Note

```
    public void mouseClicked(MouseEvent e) {
        this.setTitle("鼠标点击：("+e.getX() + "," + e.getY() + ")");
    }
    public void mouseEntered(MouseEvent arg0) {}
    public void mouseExited(MouseEvent arg0) {}
    public void mousePressed(MouseEvent arg0) {}
    public void mouseReleased(MouseEvent arg0) {}
    public static void main(String[] args) {
        new MouseEventTest1();
    }
}
```

运行，效果为：

图 17-10

点击，界面标题变为：

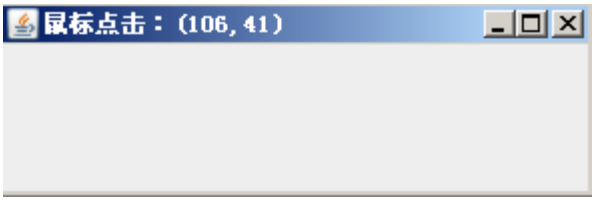


图 17-11

注意

1. 在本例中，MouseListener 接口中有 5 个函数，我们只用到一个：mouseClicked，其它函数是否可以不写呢？答案是不行，因为实现一个接口，必须将接口中的函数重写一遍。不用，也得写。不过该问题也可以通过其他手段得到解决，后面将会讲解。

2. MouseEvent 类中，封装了鼠标事件的信息。主要函数有如下几个：

(1) public int getClickCount()：返回鼠标单击次数。

(2) public int getX()和 public int getY()：返回鼠标光标在界面中的水平和垂直坐标。

其他的内容，大家可以参考文档。

3. 鼠标事件在画图软件开发时经常用到，我们将在后面的篇幅中讲解。

下面用一个程序进行测试鼠标移动事件，鼠标在 JFrame 上移动，当前坐标在界面上不断显示。代码如下：

MouseEventTest2.java

Note

```
package mouseevent;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import javax.swing.JFrame;
public class MouseEventTest2 extends JFrame implements MouseMotionListener{
    public MouseEventTest2(){
        this.addMouseMotionListener(this);
        this.setSize(300,100);
        this.setVisible(true);
    }
    public void mouseDragged(MouseEvent arg0) {}
    public void mouseMoved(MouseEvent e) {
        this.setTitle("鼠标位置：("+e.getX() + "," + e.getY() + ")");
    }
    public static void main(String[] args) {
        new MouseEventTest2();
    }
}
```

运行，效果为：

图 17-12

鼠标移动，界面标题不断变化：

图 17-13

阶段性作业

使用鼠标事件完成：在界面空白处某个位置点击鼠标，能够在该位置放置一个含有卡通图标的 JLabel，如果在该 JLabel 内拖动鼠标，则可以将该 JLabel 拖到另一个位置释放。

17.6 处理 WindowEvent

Note

17.6.1 什么情况发出 WindowEvent

在 java.awt.event 包中，WindowEvent 也经常使用。一般情况下，WindowEvent 适合窗口状态改变，如打开、关闭、激活、停用、图标化或取消图标化时，需要处理事件的场合。WindowEvent 一般用 java.awt.event.WindowListener 接口监听。该接口中有如下函数：

1. void windowOpened(WindowEvent e)：窗口首次变为可见时调用。
2. void windowClosing(WindowEvent e)：用户试图从窗口的系统菜单中关闭窗口时调用。
3. void windowClosed(WindowEvent e)：因对窗口调用 dispose 而将其关闭时调用。
4. void windowIconified(WindowEvent e) ：窗口从正常状态变为最小化状态时调用。
5. void windowDeiconified(WindowEvent e)：窗口从最小化状态变为正常状态时调用。
6. void windowActivated(WindowEvent e)：将 Window 设置为活动 Window 时调用。
7. void windowDeactivated(WindowEvent e)：当 Window 不再是活动 Window 时调用。

17.6.2 使用 WindowEvent 解决实际问题

下面用一个程序进行测试，在一个按下窗口上的关闭键，询问用户是否关闭该窗口。代码如下：

WindowEventTest1.java

```
package windowevent;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class WindowEventTest1 extends JFrame implements WindowListener{
    public WindowEventTest1(){
        //设置关闭时默认不做任何事
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        this.addWindowListener(this);
        this.setSize(200,80);
        this.setVisible(true);
    }
```

```
    public void windowClosing(WindowEvent arg0) {
        int result = JOptionPane.showConfirmDialog(this, "您确认关闭吗?",
                "确认",JOptionPane.YES_NO_OPTION);
        if(result==JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }
    public void windowActivated(WindowEvent arg0) {}
    public void windowClosed(WindowEvent arg0) {}
    public void windowDeactivated(WindowEvent arg0) {}
    public void windowDeiconified(WindowEvent arg0) {}
    public void windowIconified(WindowEvent arg0) {}
    public void windowOpened(WindowEvent arg0) {}
    public static void main(String[] args) {
        new WindowEventTest1();
    }
}
```

运行，效果为：

图 17-14

按下右上角的关闭按钮，显示确认框：

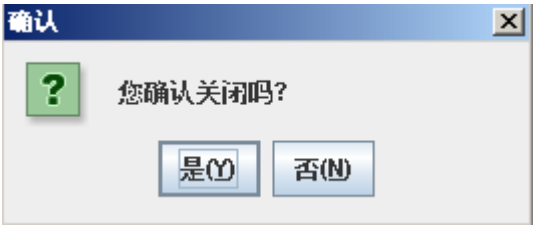


图 17-15

如果选择“是”，则关闭；如果选择“否”则不关闭。

注意

在本例中，一定要用：setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE); 设置窗口关闭时默认不做任何事，否则点击关闭按钮，界面都会关闭。

17.7 使用 Adapter 简化开发

Note

在前面的例子中，KeyEvent、MouseEvent、WindowEvent 的处理中，不约而同遇到了一个问题：Listener 接口中的函数个数较多，但是经常我们只用到一两个，由于实现一个接口，必须将接口中的函数重写一遍，因此，造成大量的空函数，用不着，不写又不行。

能否解决这个问题呢？我们知道，实现一个接口，必须将接口中的函数重写一遍，但是继承一个类，并不一定将类中的函数重写一遍，因此，java 中提供了相应的 Adapter 类来帮我们简化这个操作。

常见的 Adapter 类有：

1. KeyAdapter：内部函数和 KeyListener 基本相同。
2. MouseAdapter：内部函数和 MouseListener、MouseMotionListener 基本相同。
3. WindowAdapter：内部函数和 WindowListener 基本相同。

注意

在底层，这些 Adapter 已经实现了相应的 Listener 接口。

因此，我们编程时，就可以将事件响应的代码写在 Adapter 内。

比如，上节的 WindowEvent 的例子可以改为：

WindowAdapterTest1.java

```
package windowadapter;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class WindowAdapterTest1 extends JFrame    {
    public WindowAdapterTest1(){
        //设置关闭时不做任何事
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        this.addWindowListener(new WindowOpe());
        this.setSize(200,80);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new WindowAdapterTest1();
    }

    class WindowOpe extends WindowAdapter{
```

```
        public void windowClosing(WindowEvent arg0) {
            int result = JOptionPane.showConfirmDialog(null, "您确认关闭吗?",
                    "确认",JOptionPane.YES_NO_OPTION);
            if(result==JOptionPane.YES_OPTION){
                System.exit(0);
            }
        }
    }
}
```

运行，效果和上节相同。

注意

此代码中，由于 Adapter 是一个类，而 Java 不支持多重继承，因此，我们不得不将事件处理代码写在另一个类 WindowOpe 类中。

阶段性作业

将前几节中，和 KeyEvent、MouseEvent 有关的程序改为用 Adapter 实现。

Note

本章知识体系

知识点	重要等级	难度等级
事件原理	★★★★	★★★★
事件开发流程	★★★★	★★★★
处理 ActionEvent	★★★	★★★
处理 FocusEvent	★★	★★
处理 KeyEvent	★★★	★★★
处理 MouseEvent	★★★	★★★
处理 WindowEvent	★★	★★
使用 Adapter 简化开发	★★	★★

第18章

实践指导 4

前面学习了 JavaGUI 开发、JavaGUI 布局和 Java 事件处理，这些内容在 Java 界面编程中，属于非常重要的内容。本章将利用一个用户管理系统的案例，来对这些内容进行复习。

术语复习

GUI _____

Layout _____

Event _____

Listener _____

JFrame _____

JDialog _____

ActionEvent _____

ActionListener _____

Component _____

18.1 用户管理系统功能简介

Note

在本章中，我们将制作一个模拟的用户管理系统。用户能够将自己的账号、密码、姓名、部门存入数据库，由于没有学习数据库操作，因此，我们将内容存入文件。

该系统由 4 个界面组成。系统运行，出现登录界面：

图 18-1

该界面出现在屏幕中间。在这个界面中：

1. 点击登录按钮，能够根据输入的账号密码进行登录；如果登录失败，能够提示；如果登录成功，提示登录成功之后，能够到达操作界面。

2. 点击注册按钮，登录界面消失，出现注册界面。

3. 点击退出按钮，程序退出。

注册界面如下：

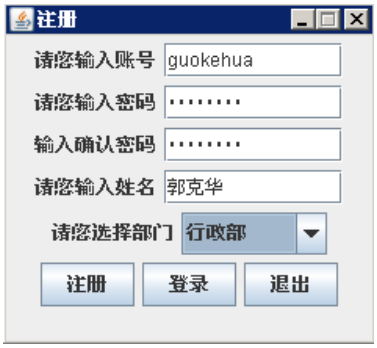


图 18-2

在这个界面中：

1. 点击注册按钮，能够根据输入的账号、密码、姓名、部门进行注册。两个密码必须相等，账号不能重复注册，部门选项如下：

图 18-3

2. 点击登录按钮，注册界面消失，出现登录界面。
3. 点击退出按钮，程序退出。

用户登录成功之后，出现操作界面，该界面效果如下：

图 18-4

在这个界面中：

1. 标题栏显示当前登录的账号。
2. 点击显示详细信息按钮，显示用户的详细信息：

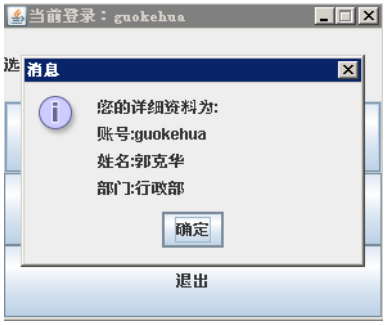


图 18-5

3. 点击退出按钮，程序退出。
4. 点击修改个人资料按钮，显示修改个人资料的对话框：

Note

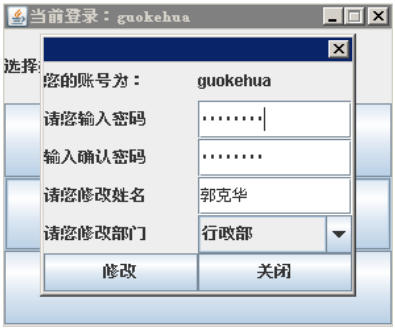


图 18-6

所有内容均初始化填入相应的控件。账号不可修改。

在这个界面中：

1. 点击修改按钮，能够修改用户信息。
2. 点击关闭按钮，能够关掉该界面。

18.2 关键技术

18.2.1 如何组织界面

在这个项目中，我们需要用到以下几个界面：登录界面，注册界面，操作界面和修改界面。很明显，这些界面各自有自己的控件和事件，四个界面应该分 4 个类，在各个类里面负责界面的界面元素和事件处理，是比较好的方法。

我们设计出来的类如下：

1.frame.LoginFrame：登录界面。

2.frame.RegisterFrame：注册界面。

3.frame.OperationFrame：操作界面。

4.frame.ModifyFrame：修改界面。

18.2.2 如何访问文件

但是，该项目有些特殊，主要是在好几个界面中都用到了文件操作，如果将文件操作的代码分散在多个界面类中，维护性较差。因此，这里有必要将文件操作的代码专门放在一个类中，让各个界面调用。

为了简化文件操作，我们将用户的信息用如下格式存储：

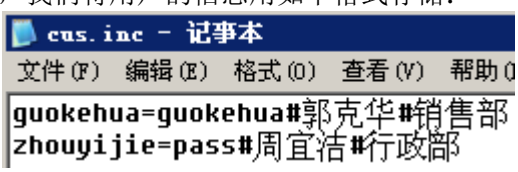


图 18-7

数据保存在 cus.inc 内，以“账号=密码#姓名#部门”的格式保存，便于用 Properties 类来读。

读文件的类设计如下：

util.FileOpe。负责读文件，将信息保存到文件。

因此，整个系统结构如图所示：

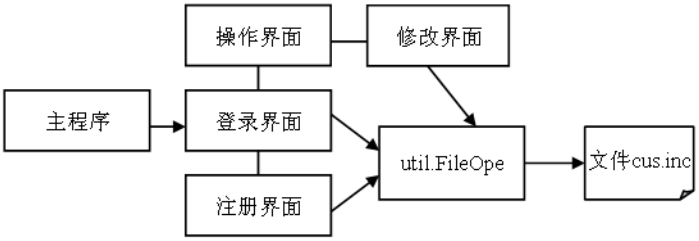


图 18-8

图 17-6 程序结构

18.2.3 如何保持状态

将项目划分为几个模块之后，模块之间的数据传递难度增大了。比如，在登录界面中，登录成功之后，系统就应该记住该用户的所有信息，否则到了操作界面，无法知道是谁在登录，到了修改界面，更无法显示其详细信息。

怎样保存其状态呢？有很多种方法，这里可以采用“静态变量法”。该方法就是将各个模块之间需要共享的数据保存在某个类的静态变量中，我们知道，静态变量一旦赋值，在另一个时刻访问，仍然是这个值，因此，可以用静态变量来传递数据。

我们设计的类如下：

util.Conf：内含 4 个静态成员：

1. public static String account;：保存登录用户的账号。
2. public static String password; ：保存登录用户的密码。
3. public static String name; ：保存登录用户的姓名。
4. public static String dept; ：保存登录用户的部门。

注意

在多线程的情况下，如果多个线程可能访问登录用户的数据，编程要十分谨慎，以免造成线程 A 将线程 B 的状态改掉的情况。不过，本项目中没有这个问题。

18.2.4 还有哪些公共功能

在本项目中，界面都要显示在屏幕中间，因此，我们可以编写一段公用代码来完成这个功能。该公用代码放在 util.GUIUtil 类中。

当然，还有一些资源文件，事先要建立好，比如，登录界面上的欢迎图片，数据文件 cus.inc 等。

Note

最终设计出来的项目结构如下：

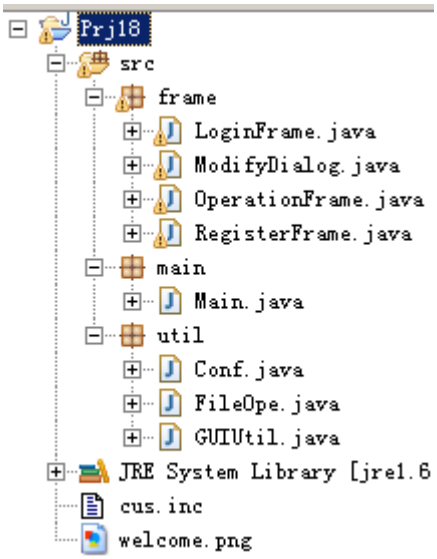


图 18-9

18.3 代码编写

18.3.1 编写 util 包中的类

首先是 Conf 类，比较简单：

Conf.java

```
package util;
public class Conf {
    public static String account;
    public static String password;
    public static String name;
    public static String dept;
}
```

然后是 FileOpe 类：

FileOpe.java

```
package util;
import java.io.FileReader;
import java.io.PrintStream;
import java.util.Properties;
import javax.swing.JOptionPane;
```

Note

```
public class FileOpe {
    private static String fileName = "cus.inc";
    private static Properties pps;
    static {
        pps = new Properties();
        FileReader reader = null;
        try{
            reader = new FileReader(fileName);
            pps.load(reader);
        }catch(Exception ex){
            JOptionPane.showMessageDialog(null, "文件操作异常");
            System.exit(0);
        }finally{
            try{
                reader.close();
            }catch(Exception ex){}
        }
    }
    private static void listInfo(){
        PrintStream ps = null;
        try{
            ps = new PrintStream(fileName);
            pps.list(ps);
        }catch(Exception ex){
            JOptionPane.showMessageDialog(null, "文件操作异常");
            System.exit(0);
        }finally{
            try{
                ps.close();
            }catch(Exception ex){}
        }
    }
    public static void getInfoByAccount(String account) {
        String cusInfo = pps.getProperty(account);
        if(cusInfo!=null){
            String[] infos = cusInfo.split("#");
            Conf.account = account;
```

Note

```
                Conf.password = infos[0];
                Conf.name = infos[1];
                Conf.dept = infos[2];
            }
        }
        public static void updateCustomer(String account,String password,
                                    String name,String dept) {
            pps.setProperty(account, password + "#" + name + "#" + dept);
            listInfo();
        }
    }
```

注意

本类中，静态代码负责载入 cus.inc 中的数据。

接下来是 FileOpe 类：

GUIUtil.java

```
package util;
import java.awt.Component;
import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;
public class GUIUtil {
    public static void toCenter(Component comp){
        GraphicsEnvironment ge =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        Rectangle rec =
            ge.getDefaultScreenDevice().getDefaultConfiguration().getBounds();
        comp.setLocation(((int)rec.getWidth()-comp.getWidth())/2,
                        ((int)rec.getHeight()-comp.getHeight())/2);
    }
}
```

注意

1. 本类中，toCenter(Component comp)函数传入的参数不是 JFrame，而是其父类 Component，完全是为了扩大本函数的适用范围，让其适用于所有 Component 的子类。

2.本类中使用了界面居中的坐标计算方法，请读者仔细理解。

18.3.2 编写 frame 包中的类

首先是登录界面类：

LoginFrame.java

Note

```
package frame;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import util.Conf;
import util.FileOpe;
import util.GUIUtil;
public class LoginFrame extends JFrame implements ActionListener{
    /**********************定义各控件*****************************/
    private Icon welcomeIcon = new ImageIcon("welcome.png");
    private JLabel lbWelcome = new JLabel(welcomeIcon);
    private JLabel lbAccount = new JLabel("请您输入账号");
    private JTextField tfAccount = new JTextField(10);
    private JLabel lbPassword = new JLabel("请您输入密码");
    private JPasswordField pfPassword = new JPasswordField(10);
    private JButton btLogin = new JButton("登录");
    private JButton btRegister = new JButton("注册");
    private JButton btExit = new JButton("退出");
    public LoginFrame(){
        /************************界面初始化************************/
        super("登录");
        this.setLayout(new FlowLayout());
        this.add(lbWelcome);
        this.add(lbAccount);
        this.add(tfAccount);
        this.add(lbPassword);
        this.add(pfPassword);
        this.add(btLogin);
        this.add(btRegister);
```

Note

```
        this.add(btExit);
        this.setSize(240, 180);
        GUIUtil.toCenter(this);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setVisible(true);
        /************************增加监听*************************/
        btLogin.addActionListener(this);
        btRegister.addActionListener(this);
        btExit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btLogin){
            String account = tfAccount.getText();
            String password = new String(pfPassword.getPassword());
            FileOpe.getInfoByAccount(account);
            if(Conf.account==null || !Conf.password.equals(password)){
                JOptionPane.showMessageDialog(this, "登录失败");
                return;
            }
            JOptionPane.showMessageDialog(this, "登录成功");
            this.dispose();
            new OperationFrame();
        }else if(e.getSource()==btRegister){
            this.dispose();
            new RegisterFrame();
        }else{
            JOptionPane.showMessageDialog(this, "谢谢光临");
            System.exit(0);
        }
    }
}
```

注意

本类中 this.dispose();表示让本界面消失，释放内存，但是程序不结束。System.exit(0);表示整个程序退出。

接下来是注册界面类：

RegisterFrame.java

Note

```
package frame;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import util.Conf;
import util.FileOpe;
import util.GUIUtil;
public class RegisterFrame extends JFrame implements ActionListener{
    /************************定义各控件*****************************/
    private JLabel lbAccount = new JLabel("请您输入账号");
    private JTextField tfAccount = new JTextField(10);
    private JLabel lbPassword1 = new JLabel("请您输入密码");
    private JPasswordField pfPassword1 = new JPasswordField(10);
    private JLabel lbPassword2 = new JLabel("输入确认密码");
    private JPasswordField pfPassword2 = new JPasswordField(10);
    private JLabel lbName = new JLabel("请您输入姓名");
    private JTextField tfName = new JTextField(10);
    private JLabel lbDept = new JLabel("请您选择部门");
    private JComboBox cbDept = new JComboBox();
    private JButton btRegister = new JButton("注册");
    private JButton btLogin = new JButton("登录");
    private JButton btExit = new JButton("退出");
    public RegisterFrame(){
        /************************界面初始化*************************/
        super("注册");
        this.setLayout(new FlowLayout());
        this.add(lbAccount);
        this.add(tfAccount);
        this.add(lbPassword1);
        this.add(pfPassword1);
```

Note

```
        this.add(lbPassword2);
        this.add(pfPassword2);
        this.add(lbName);
        this.add(tfName);
        this.add(lbDept);
        this.add(cbDept);
        cbDept.addItem("财务部");
        cbDept.addItem("行政部");
        cbDept.addItem("客户服务部");
        cbDept.addItem("销售部");
        this.add(btRegister);
        this.add(btLogin);
        this.add(btExit);
        this.setSize(240, 220);
        GUIUtil.toCenter(this);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setVisible(true);
        /************************增加监听************************/
        btLogin.addActionListener(this);
        btRegister.addActionListener(this);
        btExit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btRegister){
            String password1 = new String(pfPassword1.getPassword());
            String password2 = new String(pfPassword2.getPassword());
            if(!password1.equals(password2)){
                JOptionPane.showMessageDialog(this, "两个密码不相同");
                return;
            }
            String account = tfAccount.getText();
            FileOpe.getInfoByAccount(account);
            if(Conf.account!=null){
                JOptionPane.showMessageDialog(this, "用户已经注册");
                return;
            }
```

Note

```
                String name = tfName.getText();
                String dept = (String)cbDept.getSelectedItem();
                FileOpe.updateCustomer(account, password1, name , dept);
                JOptionPane.showMessageDialog(this, "注册成功");
            }else if(e.getSource()==btLogin){
                this.dispose();
                new LoginFrame();
            }else{
                JOptionPane.showMessageDialog(this, "谢谢光临");
                System.exit(0);
            }
        }
    }
```

接下来是操作界面类：

OperationFrame.java

```
package frame;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import util.Conf;
import util.GUIUtil;
public class OperationFrame extends JFrame implements ActionListener{
    /************************定义各控件*****************************/
    private String welcomeMsg = "选择如下操作:";
    private JLabel lbWelcome = new JLabel(welcomeMsg);
    private JButton btQuery = new JButton("显示详细信息");
    private JButton btModify = new JButton("修改个人资料");
    private JButton btExit = new JButton("退出");
    public OperationFrame(){
        /************************界面初始化*************************/
        super("当前登录：" + Conf.account);
        this.setLayout(new GridLayout(4,1));
        this.add(lbWelcome);
```

Note

```
        this.add(btQuery);
        this.add(btModify);
        this.add(btExit);
        this.setSize(300, 250);
        GUIUtil.toCenter(this);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.setVisible(true);
        /**********************增加监听**************************/
        btQuery.addActionListener(this);
        btModify.addActionListener(this);
        btExit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btQuery){
            String message = "您的详细资料为:\n";
            message += "账号:" + Conf.account + "\n";
            message += "姓名:" + Conf.name + "\n";
            message += "部门:" + Conf.dept + "\n";
            JOptionPane.showMessageDialog(this, message);
        }else if(e.getSource()==btModify){
            new ModifyDialog(this);
        }else{
            JOptionPane.showMessageDialog(this, "谢谢光临");
            System.exit(0);
        }
    }
}
```

最后是 ModifyDialog 类，注意，ModifyDialog 是个模态对话框。

ModifyDialog.java

```
package frame;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
```

Note

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import util.Conf;
import util.FileOpe;
import util.GUIUtil;
public class ModifyDialog extends JDialog implements ActionListener{
    /************************定义各控件*****************************/
    private JLabel lbMsg = new JLabel("您的账号为：");
    private JLabel lbAccount = new JLabel(Conf.account);
    private JLabel lbPassword1 = new JLabel("请您输入密码");
    private JPasswordField pfPassword1 = new JPasswordField(Conf.password,10);
    private JLabel lbPassword2 = new JLabel("输入确认密码");
    private JPasswordField pfPassword2 = new JPasswordField(Conf.password,10);
    private JLabel lbName = new JLabel("请您修改姓名");
    private JTextField tfName = new JTextField(Conf.name,10);
    private JLabel lbDept = new JLabel("请您修改部门");
    private JComboBox cbDept = new JComboBox();
    private JButton btModify = new JButton("修改");
    private JButton btExit = new JButton("关闭");
    public ModifyDialog(JFrame frm){
        /************************界面初始化************************/
        super(frm,true);
        this.setLayout(new GridLayout(6,2));
        this.add(lbMsg);
        this.add(lbAccount);
        this.add(lbPassword1);
        this.add(pfPassword1);
        this.add(lbPassword2);
        this.add(pfPassword2);
        this.add(lbName);
        this.add(tfName);
        this.add(lbDept);
        this.add(cbDept);
        cbDept.addItem("财务部");
```

Note

```
            cbDept.addItem("行政部");
            cbDept.addItem("客户服务部");
            cbDept.addItem("销售部");
            cbDept.setSelectedItem(Conf.dept);
            this.add(btModify);
            this.add(btExit);
            this.setSize(240, 200);
            GUIUtil.toCenter(this);
            this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            /************************增加监听*************************/
            btModify.addActionListener(this);
            btExit.addActionListener(this);
            this.setResizable(false);
            this.setVisible(true);
        }
        public void actionPerformed(ActionEvent e) {
            if(e.getSource()==btModify){
                String password1 = new String(pfPassword1.getPassword());
                String password2 = new String(pfPassword2.getPassword());
                if(!password1.equals(password2)){
                    JOptionPane.showMessageDialog(this, "两个密码不相同");
                    return;
                }
                String name = tfName.getText();
                String dept = (String)cbDept.getSelectedItem();
                //将新的值存入静态变量
                Conf.password = password1;
                Conf.name = name;
                Conf.dept = dept;
                FileOpe.updateCustomer(Conf.account, password1, name , dept);
                JOptionPane.showMessageDialog(this, "修改成功");
            }else{
                this.dispose();
            }
        }
    }
```

18.3.3 编写主函数所在的类

主函数所在的类，调用登录界面类：

Main.java

```
package main;
import frame.LoginFrame;
public class Main {
    public static void main(String[] args) {
            new LoginFrame();
    }
}
```

运行该类，则可以出现登录界面。

Note

18.4 思考题

本程序开发完毕，留下几个思考题，请大家思考：

1. 该程序中，需要用 Properties 类将整个文件读入进行处理，如果遇到文件较大的情况，会有什么问题？如何解决？

2. 将用户的登录信息用静态变量存储，在多线程情况下，有什么隐患？你能否举出一个例子？如何解决？

第22章

用 TCP 开发网络应用程序

从本章开始，将讲解网络编程，在网络编程框架内，我们主要针对比较重要的几种应用进行讲解，它们是：TCP 编程和 UDP 编程。

本章讲解 TCP 编程。TCP 编程是一种应用比较广泛的编程方式。我们将利用 TCP 编程，实现一个简单的聊天室。

本章术语

TCP _____

UDP _____

IP 地址 _____

端口 _____

ServerSocket _____

Socket _____

PrintStream _____

BufferedReader _____

22.1 认识网络编程

Note

22.1.1 什么是网络应用程序

在本书的前面几章中，我们的程序都是在一台单独的机器上运行，这一般称为单机版的软件。单机版软件不具备网络通信的功能；与此对应的是网络应用程序，能够通过网络，和另一台机器通信。比如下面的软件：

图 22-1

可以将另外机器上的一个文件，通过迅雷下载到自己的机器上。又如：

图 22-2

可以将一条聊天信息通过网络发到别人的机器。这些都属于网络应用程序。

本章讲解如何编写网络应用程序。

22.1.2 认识 IP 地址和端口

在编写网络应用程序之前，首先必须明白几个概念。

1. 通过什么来找到网络上的计算机。

要和别的机器通信，必须找到另一台机器在哪里。如何确定对方机器呢？很明显，用机器名称是不现实的，因为名称可能重复；实际上，我们是通过 IP 地址来确定一台计算机在网络上的位置的。

IP 地址被用来给 Internet 上的电脑一个编号。我们可以把一台机器比作一部手机，那么 IP 地址就相当于手机号码。

IP 地址是一个 32 位的二进制数，通常被分割为 4 个“8 位二进制数”。为了方便起见，IP 地址通常用“点分十进制”表示成（a.b.c.d）的形式，其中，a,b,c,d 都是 0~255 之间的十进制整数。比如，192.168.1.5，就是一个 IP 地址。

Note

问答

问：如何知道本机 IP 地址？

答：在 cmd 窗口中输入命令：ipconfig，即可显示 IP 地址：

图 22-3

或者右击“网上邻居”，选择“属性”，选取相应连接，右击，选择“属性”，出现如下界面：

图 22-4

双击“Internet 协议(TCP/IP)”，即可显示 IP 地址以及其他配置。

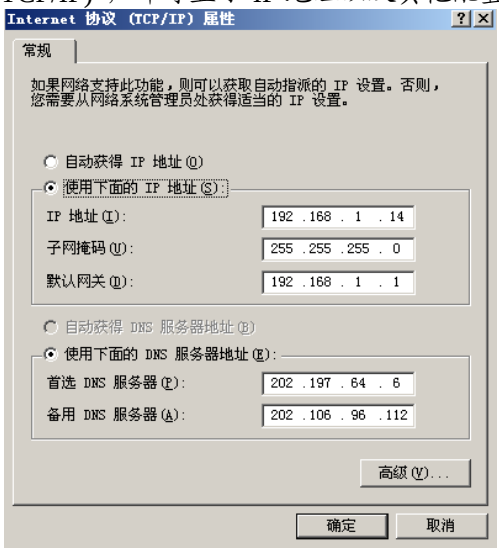


图 22-5

从上图可以看出，本机 IP 地址为 192.168.1.14，不过为了简便起见，统一可以用

127.0.0.1 表示本机 IP 地址。就好像每个人姓名不同，但是都可以自称“我”一样。

问：对方机器用 IP 地址确定，我们如何找到它？

答：某个 IP 地址的机器在网络的哪个地方，一般由路由器来判断，比如，我们要找 220.170.91.146，路由器会帮我们找到。具体找到过程，不是本课的内容。就好像我们要给对方打手机，不用关心移动公司怎样找到对方的一样。

Note

2. 通过什么来确定对方的网络通信程序。

找到计算机之后，就可以通信了吗？不一定。因为网络通信最终是软件之间的通信，还必须定位相应的软件。

我们首先来思考一个问题，一台联网的机器，只有一个网卡一根网线，为什么可以同时用多个程序上网？比如，我们可以用 FTP 下载文件，同时浏览网页，还可以 QQ 聊天，这些数据，是通过一根网线传过来的，为什么不会混淆呢？

实际上，我们是通过端口号(port)来确定一台计算机中特定的网络程序的。

我们可以将机器比作一栋办公楼，IP 就是这栋楼的地址，而端口就是办公楼中各个房间的房间号，虽然很多人都从大楼大门涌入，但是最后都进了不同的房间，每个房间负责完成不同的事情。

一台机器的端口号可以在 0-65535 之间。

这样，我们就可以理解，FTP 下载文件、浏览网页、QQ 聊天，这些程序应该对应不同的端口，信息传输到本机时，根据端口来进行分类，用不同的程序来处理数据。

问答

问：如何知道本机使用了哪些端口？

答：在 cmd 窗口中输入命令：netstat –an，即可显示本机使用了哪些端口：

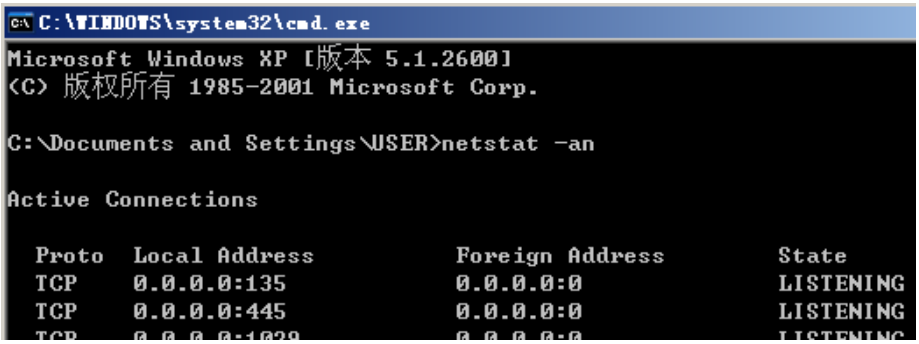


图 22-6

其中，IP 地址中，冒号后面你的数字(“0.0.0.0:135 中的 135”)，就是端口号。

问：常用应用程序的端口号有哪些？

答：21：FTP 协议；22：SSH 安全登录；23：Telnet；25：SMTP 协议；80：HTTP 协议。具体协议的意义，大家可以查看文档。

因此，在我们自己编写的网络应用程序中，要尽量避开这些常用的端口。一般情况下，0-1024 之间的端口最好不要使用。

3. 什么是 TCP？什么是 UDP？

TCP 和 UDP 是两种网络信息传输协议，都能够进行网络通信，你可以选择其中

的一种。

TCP 最重要的特点是面向连接，也就是说必须在服务器端和客户端连接上之后才能通信，它的安全性比较高。UDP 编程是面向非连接的，UDP 是数据报，只负责传输信息，并不能保证信息一定会被收到，虽然安全性不如 TCP，但是性能较好；TCP 基于连接，UDP 基于报文，具体大家可以参考计算机网络知识。

Note

你可以将 TCP 比喻成打电话，必须双方都拿起话机才能通话，并且连接要保持通畅。UDP 比喻成寄信，在寄信的时候，对方根本不知道有信要寄过去，信寄到哪里，靠信封上的地址。

没有必要讨论哪一种通信方式更好，这就像问打电话和寄信哪个好一样。

阶段性作业

上网搜索以下名词的全称：TCP、UDP、HTTP、FTP，它们有何区别？

22.1.3　客户端和服务器

客户端(Client)/服务器(Server)是一种最常见的网络应用程序的运行模式，简称 C/S。以网络聊天软件为例，在聊天程序中，各个聊天的界面叫做客户端，客户端之间如果要相互聊天，则可以将信息先发送到服务器端，然后由服务器端转发。因此，客户端先要连接到服务器端。

客户端连接到服务器端，需要知道一些什么信息呢？

显然，首先需要知道服务器端的 IP 地址，还要知道服务器端该程序的端口。如，知道服务器 IP 地址是 127.0.0.1，端口是 9999 等。

因此，服务器必须首先打开这个端口，等待客户端的连接，俗称打开并监听某个端口。

在客户端，必须要做到以下工作：根据服务器 IP，连接服务器的某个端口。

22.2　用客户端连到服务器

22.2.1　案例介绍

本节中我们开发一个聊天应用最基本的程序：客户端连接到服务器。首先，运行服务器，得到如图的界面：

图 22-7

Note

服务器运行完毕，界面上标题为：“服务器端，目前未见连接”。然后运行客户端，界面如图所示：

图 22-8

客户端运行完毕，界面上标题为：“客户端”。在上面有一个“连接”按钮，点击，连接到服务器端。服务器端界面变为如图所示的界面：

图 22-9

该界面上显示客户端的 IP 地址。

同时，客户端变为如图所示的界面，界面标题变为：“恭喜您，已经连上”：

图 22-10

22.2.2 如何实现客户端连接到服务器

前面说过，客户端连接到服务器端，首先需要知道服务器端的 IP 地址，还要知道服务器端该程序的端口。

服务器必须首先打开某个端口并监听，等待客户端的连接。客户端根据服务器 IP，连接服务器的某个端口。

本例中，服务器为本机，打开并监听的端口号是 9999。

1. 服务器端怎样打开并监听端口？

端口的监听是由 java.net.ServerSocket 进行管理的，打开 java.net.ServerSocket 的文档，这个类有很多构造函数，最常见的构造函数为：

public ServerSocket(int port) throws IOException

传入一个端口号，实例化 ServerSocket。

注意

实例化 ServerSocket，就已经打开了端口号并进行监听。

例如，如下代码就可以监听服务器上的 9999 端口，并返回 ServerSocket 对象 ss。

```
ServerSocket ss = new ServerSocket(9999);
```

2.客户端怎样连接到服务器端的某个端口？

客户端连接到服务器端的某个端口是由 java.net.Socket 进行管理的，打开 java.net.Socket 的文档，这个类有很多构造函数，最常见的构造函数为：

public Socket(String host, int port) throws UnknownHostException, IOException

传入一个服务器 IP 地址和端口号，实例化 Socket。

注意

实例化 Socket，就已经请求连接到该 IP 地址对应的服务器。

例如，如下代码就可以连接服务器 218.197.118.80 上的 9999 端口，并返回连接 Socket 对象 socket。

```
Socket socket = new Socket("218.197.118.80",9999);
```

3. 服务器怎么知道客户端连上来了？

既然客户端用 Socket 来向服务器请求连接，如果连接上之后，Socket 对象自然成为连接的纽带。对于服务器端来说，就应该得到客户端的这个 Socket 对象，并以此为基础来进行通信。

怎样得到客户端的 Socket 对象？在前面的篇幅中我们知道，服务器端实例化 ServerSocket 对象，监听端口。打开 ServerSocket 文档，会发现里面有一个重要函数：

public Socket accept() throws IOException

该函数返回一个 Socket 对象，因此，在服务器端可以用如下代码得到客户端的 Socket 对象：

```
Socket socket = ss.accept();
```

注意

值得一提的是，accept 函数是一个“死等函数”，如果没有客户端请求连接，它会一直等待并阻塞程序。为了说明这个问题，我们编写如下代码进行测试：

AcceptTest.java

```
package chat1;
import java.net.ServerSocket;
import java.net.Socket;
public class AcceptTest {
    public static void main(String[] args) throws Exception {
        // 监听 9999 端口
        ServerSocket ss = new ServerSocket(9999);
        System.out.println("未连接");
        // 等待客户端连接,如果没有客户端连接，程序在这里阻塞
        Socket socket = ss.accept();
        System.out.println("连接");
```

```
    }
}
```

运行这个程序，控制台上打印：

```
未连接
```

图 22-11

Note

没有打印“连接”，说明程序在 accept 处阻塞。

当然，如果此时有另一个客户端进行连接，阻塞就可以解除：

AcceptTest_Client.java

```
package chat1;
import java.net.Socket;
public class AcceptTest_Client {
    public static void main(String[] args)throws Exception {
        Socket socket = new Socket("127.0.0.1",9999);
    }
}
```

运行客户端，服务器端打印：

```
未连接
连接
```

图 22-12

说明服务器阻塞被解除。

4. 如何从 Socket 得到一些连接的基本信息？

了解了客户端怎样连接到服务器端，很显然，客户端和服务器端用 Socket 对象来进行通信。那么，从 Socket 能否得到一些连接的基本信息呢？

打开 Socket 文档，会发现有如下函数：

```
public InetAddress getInetAddress()
```

返回 Socket 内连接客户端的地址。

该返回类型是 java.net.InetAddress，查找 java.net.InetAddress 文档，可以用以下方法得到 IP 地址：

```
public String getHostAddress()
```

返回 IP 地址字符串（以文本形式）。

22.2.3 代码编写

综合以上叙述，建立如下服务器端代码：

Server.java

```
package chat1;
import java.net.ServerSocket;
import java.net.Socket;
```

Note

```
import javax.swing.JFrame;
public class Server extends JFrame{
    private ServerSocket ss;
    private Socket socket;
    public Server(){
        super("服务器端，目前未见连接");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(300,100);
        this.setVisible(true);
        try{
            //监听 9999 端口
            ss = new ServerSocket(9999);
            socket = ss.accept();
            String clientAddress = socket.getInetAddress().getHostAddress();
            this.setTitle("客户" + clientAddress + "连接");
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Server();
    }
}
```

运行这个程序，就可以得到服务器的效果。

接下来是客户端程序，代码如下：

Client.java

```
package chat1;
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.Socket;
import javax.swing.JButton;
import javax.swing.JFrame;
public class Client extends JFrame implements ActionListener{
    private JButton btConnect = new JButton("连接");
    private Socket socket;
    public Client(){
```

Note

```
        super("客户端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(btConnect,BorderLayout.NORTH);
        btConnect.addActionListener(this);
        this.setSize(300,100);
        this.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        try{
            socket = new Socket("127.0.0.1",9999);
            this.setTitle("恭喜您，已经连上");
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Client();
    }
}
```

运行，得到客户端界面，点击按钮，则可以连接到服务器。

注意

必须要先运行服务器端，再运行客户端。

阶段性作业

客户端连接到服务器，连接成功，双方的提示信息用消息框显示。

22.3 利用 TCP 实现双向聊天系统

22.3.1 案例介绍

上一节中已经讲述了客户端和服务器端的连接，接下来就可以让客户端和服务器端进行通信了。在本节中，服务器和客户端界面相同，都可以给对方发送信息，也能够自动收到对方发过来的信息。本节的效果如图所示：

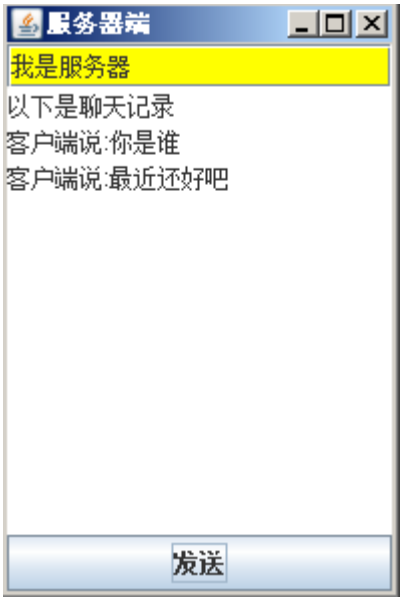


图 22-13

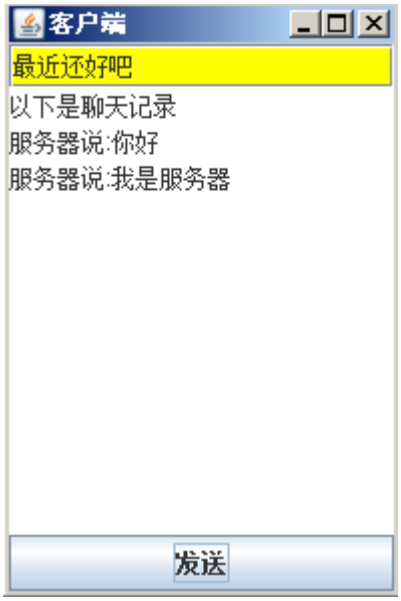


图 22-14

服务器端和客户端都有一个文本框，输入聊天信息。输入聊天信息之后，点击“发送”，就能够将信息发送给对方，对方也能够自动收到之后显示。

22.3.2　如何实现双向聊天

客户端与服务器端的通信过程，包括读信息和写信息，对于客户端和服务器端，如果将数据传给对方，就称为写，用到输出流；反之，如果从对方处得到数据，就为读，用到输入流。

TCP 编程中，客户端和服务器端之间的通信是通过 Socket 实现的。

1. 如何向对方发送信息？

从 java.net.Socket 文档中，会发现其中有一个重要函数：

public OutputStream getOutputStream() throws IOException

打开此 Socket 的输出流。

我们知道，OutputStream 功能并不强大，但是我们可以和 java.io.PrintStream 类配合使用，使之能够输出一行。如下代码：

```
Socket socket = new Socket("127.0.0.1",9999);
OutputStream os = socket.getOutputStream();
PrintStream ps = new PrintStream(os);
ps.println("消息内容");
```

就是用 Socket 向对方发出一个字符串。

2. 如何从对方处接收信息？

打开 java.net.Socket 文档，会发现其中有一个重要函数：

public InputStream getInputStream() throws IOException

打开此 Socket 的输入流。

我们知道，InputStream 功能并不强大，但是我们可以和 BufferedReader 函数配合使用，使之能够读取一行。如下代码：

Note

```
Socket socket = new Socket("127.0.0.1",9999);
InputStream is = socket.getInputStream(); // 得到输入流,InputStream 功能不强大
BufferedReader br = new BufferedReader(new InputStreamReader(is));
String str = br.readLine();// 读
System.out.println(str);
```

就是从 Socket 的输入流中读入字符串，并打印。

很明显，在本例中，客户端和服务器端的通信，既要用到读操作，又要用到写操作。

为了对这个功能进行测试，我们在项目中建立一个服务器程序和客户端程序。让客户端发送给服务器端一个“服务器，你好”，然后服务器端收到之后打印。服务器端程序为：

Server.java

```
package chat2;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Server{
    public static void main(String[] args) throws Exception{
        ServerSocket ss = new ServerSocket(9999);
        Socket s = ss.accept();
        //获取对方传过来的信息，并打印
        InputStream is = s.getInputStream();
        BufferedReader br = new BufferedReader(new
InputStreamReader(is));
        String str = br.readLine();//读
        System.out.println(str);
    }
}
```

然后编写客户端，代码为：

Client.java

```
package chat2;
import java.net.Socket;
```

Note

```
import java.io.OutputStream;
import java.io.PrintStream;
public class Client{
    public static void main(String[] args) throws Exception{
        Socket s = new Socket("127.0.0.1",9999);//连接到服务器
        OutputStream os = s.getOutputStream();//os 只能发字节数组
        PrintStream ps = new PrintStream(os);    //ps 功能更强大
        ps.println("服务器，你好!");//信息发送出去
    }
}
```

首先运行服务器端，然后运行客户端，在服务器端的控制台上会打印：

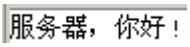


图 22-15

说明信息由客户端传输到了服务器端，并被服务器端收取。

注意

值得一提的是，在客户端与服务器端之间传递信息时，BufferedReader 的 readLine 函数也是一个“死等函数”，如果客户端连接上了，但是没有发送信息，readLine 函数会一直等待。为了说明这个问题，我们编写如下代码进行测试，服务器端代码为：

ReadLineTest.java

```
package chat2;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class ReadLineTest{
    public static void main(String[] args) throws Exception{
        ServerSocket ss = new ServerSocket(9999);
        Socket s = ss.accept();
        InputStream is = s.getInputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(is));
        System.out.println("未收到信息");
        String str = br.readLine();//读
        System.out.println("收到信息");
        System.out.println(str);
    }
}
```

客户端代码为：

ReadLineTest_Client.java

```
package chat2;
import java.net.Socket;
public class ReadLineTest_Client {
    public static void main(String[] args)throws Exception {
        Socket socket = new Socket("127.0.0.1",9999);
    }
}
```

Note

运行服务器端，再运行客户端，服务器控制台上打印：

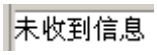


图 22-16

没有打印“收到信息”，说明程序在 readLine 处阻塞。

当然，如果客户端给服务器发送一条信息，阻塞就可以解除：

由以上情况可以看出，客户端和服务器端如果需要自动读取对方传来的信息，就不能将 readLine 函数放在主线程内，因为在不知道对方在什么时候会发出信息的情况下，readLine 函数的死等，可能会造成程序的阻塞。所以，最好的方法是将读取信息的代码写在线程内。

22.3.3 代码编写

综合以上叙述，建立如下服务器端代码：

Server.java

```
package chat3;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
public class Server extends JFrame implements ActionListener, Runnable {
    private JTextArea taMsg = new JTextArea("以下是聊天记录\n");
    private JTextField tfMsg = new JTextField("请您输入信息");
    private JButton btSend = new JButton("发送");
    private Socket s = null;
    public Server() {
        this.setTitle("服务器端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Note

```
        this.add(taMsg,BorderLayout.CENTER);
        tfMsg.setBackground(Color.yellow);
        this.add(tfMsg,BorderLayout.NORTH);
        this.add(btSend,BorderLayout.SOUTH);
        btSend.addActionListener(this);
        this.setSize(200, 300);
        this.setVisible(true);
        try {
            ServerSocket ss = new ServerSocket(9999);
            s = ss.accept();
            new Thread(this).start();
        } catch (Exception ex) {
        }
    }
    public void run() {
        try {
            while (true) {
                InputStream is = s.getInputStream();
                BufferedReader br = new BufferedReader(
                        new InputStreamReader(is));
                String str = br.readLine();// 读
                taMsg.append(str + "\n");// 添加内容
            }
        } catch (Exception ex) {
        }
    }
    public void actionPerformed(ActionEvent e) {
        try {
            OutputStream os = s.getOutputStream();
            PrintStream ps = new PrintStream(os);
            ps.println("服务器说:"+tfMsg.getText());
        } catch (Exception ex) {
        }
    }
    public static void main(String[] args) throws Exception {
        Server server5 = new Server();
    }
```

```
}
```

运行这个程序，就可以得到服务器的效果。

接下来是客户端程序，代码如下：

Client.java

```
package chat3;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
public class Client extends JFrame implements ActionListener,Runnable{
    private JTextArea taMsg = new JTextArea("以下是聊天记录\n");
    private JTextField tfMsg = new JTextField("请您输入信息");
    private JButton btSend = new JButton("发送");
    private Socket s = null;
    public Client(){
        this.setTitle("客户端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(taMsg,BorderLayout.CENTER);
        tfMsg.setBackground(Color.yellow);
        this.add(tfMsg,BorderLayout.NORTH);
        this.add(btSend,BorderLayout.SOUTH);
        btSend.addActionListener(this);
        this.setSize(200, 300);
        this.setVisible(true);
        try{
            s = new Socket("127.0.0.1",9999);
            new Thread(this).start();
        }catch(Exception ex){}
    }
    public void run(){
        try{
            while(true){
                InputStream is = s.getInputStream();
                BufferedReader br = new BufferedReader(
                        new InputStreamReader(is));
                String str = br.readLine();//读
```

Note

```
                    taMsg.append(str+"\n");//添加内容
                }
            }catch(Exception ex){}
        }
        public void actionPerformed(ActionEvent e){
            try{
                OutputStream os = s.getOutputStream();
                PrintStream ps = new PrintStream(os);
                ps.println("客户端说:"+tfMsg.getText());
            }catch(Exception ex){}
        }
        public static void main(String[] args) throws Exception{
            Client client5 = new Client();
        }
    }
```

运行，得到客户端界面，两者即可进行聊天。

注意

必须要先运行服务器端，再运行客户端。

阶段性作业

1. 将本例中的按钮去掉，改为：在文本框中回车，信息自动发出，文本框清空。

2. 完成一个网络远程控制系统，如果服务器给客户端发送的信息为：“关闭”二字，客户端能够自动关闭。

3. 完成一个简单的隐私窃取软件，如果客户端连到服务器，能够自动将其 C 盘下的所有文件名称传到服务器端显示。

22.4　利用 TCP 实现多客户聊天系统

22.4.1　案例介绍

上一节中已经讲述了客户端和服务器端的互相通信。但是，实际应用中，应该是客户端和客户端聊天，而不是客户端和服务器端聊天。客户端和客户端聊天的本质是信息由服务器端转发。因此，本节将开发一个支持多个客户端的程序。服务器端界面如图所示：

图 22-17

Note

以下是客户端界面，当客户端出现时，需要输入昵称：

图 22-18

点击“确定”，连接到服务器。如果连接成功，服务器回送一个信息：

图 22-19

确定，即可进行聊天。

为了体现多客户端效果，我们打开了 3 个客户端，如图所示：

图 22-20

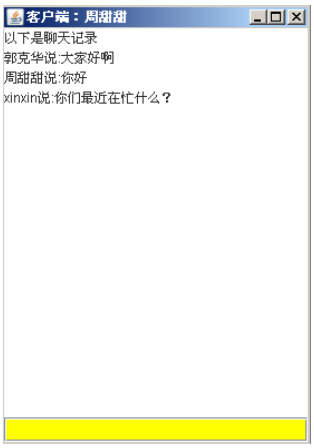


图 22-21

图 22-22

在界面下方可以输入消息，回车，消息发出。消息发出之后，文本框自动清空。消息发送之后，能够让各个客户端都收到聊天信息，聊天信息打印在界面上的多行文本框内，在打印聊天信息的同时，还能够打印这条聊天信息是谁说的。

22.4.2 编写服务器程序

在本例中，要让服务器端能够接受多个客户端的连接，需要注意以下几个问题：

1. 由于事先不知道客户端什么时候连过来，因此，服务器端必须首先有一个线程负责接受多个客户端连接；结构如下：

```
public class Server extends JFrame implements Runnable{
    public Server(){
        //服务器端打开端口
        //服务端开启线程，接受客户端连接
    }
    public void run(){
        //不断接受客户端连接
        while(true){
                //接收客户端连接
                 //开一个聊天线程给这个客户端
                //将该聊天线程对象添加进集合
               //聊天线程启动
        }
    }
}
```

Note

2．当客户端连接上之后，服务器端要等待这些客户端传送信息过来，而事先并不知道客户端什么时候会发信息过来。所以，每一个客户端连上之后，必须为这个客户端单独开一个线程，来读取它发过来的信息。因此，需要再编写一个线程类。

3. 服务器收到某个客户端信息之后，需要将其转发给各个客户端，这就需要在服务器端保存各个客户端的输入输出流的引用(实际上，这些引用可以保存在为客户端服务的线程中)。

因此，整个服务器端程序的基本结构如下：

```
public class Server extends JFrame implements Runnable{
    public Server(){
        //服务器端打开端口
        //服务端开启线程，接受客户端连接
    }
    public void run(){
        //不断接受客户端连接
        while(true){
                //接收客户端连接
                //开一个聊天线程给这个客户端
                //将该聊天线程对象添加进集合
                //聊天线程启动
        }
```

Note

```
        }
        /*聊天线程类，每连接上一个客户端，就为它开一个聊天线程*/
        class ChatThread extends Thread{
            //负责读取相应 SocketConnection 的信息
            public void run(){
                while(true){
                    //读取客户端发来的信息
                    //将该信息发送给所有其他客户端
                }
            }
        }
}
```

服务器端详细代码如下：

Server.java

```
package chat4;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.util.ArrayList;
import javax.swing.JFrame;
public class Server extends JFrame implements Runnable{
    private Socket s = null;
    private ServerSocket ss = null;
    private ArrayList clients = new ArrayList();//保存客户端的线程
    public Server() throws Exception{
        this.setTitle("服务器端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setBackground(Color.yellow);
        this.setSize(200,100);
        this.setVisible(true);
        ss = new ServerSocket(9999);//服务器端开辟端口，接受连接
        new Thread(this).start(); //接受客户连接的死循环开始运行
    }
    public void run(){
        try{
            while(true){
                s = ss.accept();
```

Note

```
                //s 就是当前的连接对应的 Socket，对应一个客户端
                //该客户端随时可能发信息过来，必须要接收
                //另外开辟一个线程，专门为这个 s 服务，负责接受信息
                ChatThread ct = new ChatThread(s);
                clients.add(ct);
                ct.start();
            }
        }catch(Exception ex){}
    }
    class ChatThread extends Thread{//为某个 Socket 负责接受信息
        private Socket s = null;
        private BufferedReader br = null;
        public PrintStream ps = null;
        public ChatThread(Socket s) throws Exception    {
            this.s = s;
            br = new BufferedReader(
                    new InputStreamReader(s.getInputStream()));
            ps = new PrintStream(s.getOutputStream());
        }
        public void run(){
            try{
                while(true){
                    String str = br.readLine();//读取该 Socket 传来的信息
                    sendMessage(str);  //将 str 转发给所有客户端
                }
            }catch(Exception ex){}
        }
    }
    public void sendMessage(String msg){//将信息发给所有客户端
        for(int i=0;i<clients.size();i++){
            ChatThread ct = (ChatThread)clients.get(i);
            //向 ct 内的 Socket 内写 msg
            ct.ps.println(msg);
        }
    }
    public static void main(String[] args) throws Exception{
        Server server = new Server();
```

```
        }
    }
```

Note

22.4.3 编写客户端程序

客户端编程相对简单，只需要编写发送信息、连接服务器、接受服务器端传输的信息即可。代码如下：

Client.java

```
package chat4;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
import javax.swing.*;
public class Client extends JFrame implements ActionListener, Runnable {
    private JTextArea taMsg = new JTextArea("以下是聊天记录\n");
    private JTextField tfMsg = new JTextField();
    private Socket s = null;
    private String nickName = null;
    public Client() {
        this.setTitle("客户端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(taMsg,BorderLayout.CENTER);
        tfMsg.setBackground(Color.yellow);
        this.add(tfMsg,BorderLayout.SOUTH);
        tfMsg.addActionListener(this);
        this.setSize(280, 400);
        this.setVisible(true);
        nickName = JOptionPane.showInputDialog("输入昵称");
        try {
            s = new Socket("127.0.0.1", 9999);
            JOptionPane.showMessageDialog(this,"连接成功");
            this.setTitle("客户端：" + nickName);
            new Thread(this).start();
        } catch (Exception ex) {}
    }
    public void run() {
        try {
```

Note

```
                while (true) {
                    InputStream is = s.getInputStream();
                    BufferedReader br = new BufferedReader(
                            new InputStreamReader(is));
                    String str = br.readLine();// 读
                    taMsg.append(str + "\n");// 添加内容
                }
            } catch (Exception ex) {
            }
        }
        public void actionPerformed(ActionEvent e) {
            try {
                OutputStream os = s.getOutputStream();
                PrintStream ps = new PrintStream(os);
                ps.println(nickName + "说:" + tfMsg.getText());
                tfMsg.setText("");
            } catch (Exception ex) {
            }
        }
        public static void main(String[] args) throws Exception {
            Client client = new Client();
        }
    }
```

运行服务器端和客户端，就可以得到本案例需求中的效果。

阶段性作业

1. 在客户端增加一个下拉列表框，显示每个在线客户的昵称，如果某个客户下线，可以通知其他客户进行刷新，如何实现？

2. 客户可以在下拉列表中选择自己要发送信息的人，进行私聊，如何实现？

本章知识体系

知识点	重要等级	难度等级
网络编程若干概念	★★★★	★★
ServerSocket	★★★	★★
Socket	★★★★	★★★
客户端连到服务器	★★★	★★

客户端和服务器通信	★★★★	★★★
基于线程双向通信	★★★	★★★★
多客户端	★★★	★★★★★

Note

第25章

实践指导 6

前面学习了 TCP 网络编程、UDP 网络编程、URL 编程和 Applet 开发。本章将利用一个网络对战的打字游戏，来对网络编程内容进行复习。

术语复习

IP 地址 _____

端口 _____

TCP _____

ServerSocket _____

Socket _____

UDP _____

DatagramSocket _____

DatagramPacket _____

URL _____

Applet _____

25.1 网络打字游戏功能简介

Note

在本章中，我们将制作一个网络对战的打字游戏。首先运行服务器，界面如下：

图 25-1

服务器运行之后，客户可以加入打字游戏中。

运行客户端，首先显示：

图 25-2

用户能够输入昵称，确定，则连接到服务器。本章服务器运行在本机，端口为 9999。

连接成功，显示：

图 25-3

点击确定按钮，即可出现打字游戏界面。

实际上，多人可以加入打字对战。界面如下：

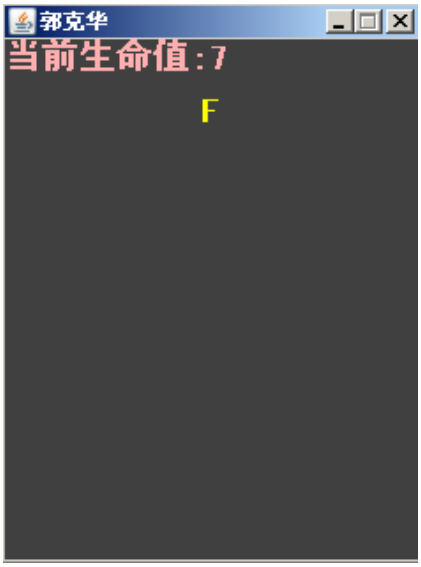


图 25-4

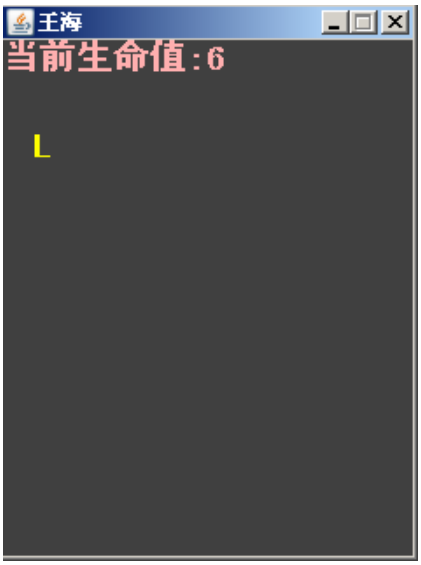


图 25-5

规则如下：

1. 初始生命值为 10 分，字母随机落下。
2. 用户按下键盘，如果输入的字符正确，则加 1 分，错误，减 1 分。
3. 如果用户加 1 分，将其他所有用户的分数减 1 分。
4. 字母掉到用户界面底部，用户减 1 分，重新出现新字母。
5. 如果生命值变为 0 分，则游戏退出。

图 25-6

注意

此案例是很多网络对战游戏的基础，比如网络打牌、网络赛车、网络五子棋等等。

25.2 关键技术

25.2.1 如何组织界面

在这个项目中，服务器端界面比较简单。客户端也只有一个界面，但是我们最好将游戏的工作写在一个面板内，然后将面板加到一个 JFrame 中。

设计出来的类如下：

1. client.GamePanel：客户端游戏所在的面板。
2. client.GameFrame：客户端游戏面板所在的界面。
3. server.Server：服务器界面。

Note

25.2.2 客户端如何掉下字母

我们可以通过画图技术在界面上画出字母。不过，在 JavaGUI 中，还有一种更加简单的方法，那就是将面板设置为空布局之后，将字母放在一个 JLabel 中。

字母的掉下，实际上相当于调整 JLabel 的位置。

代码如下：

```
……
public class GamePanel extends JPanel…… {
    ……
    //  掉下来的字母 Label
    private JLabel lbMoveChar = new JLabel();
    public GamePanel(){
        this.setLayout(null);
        ……
        this.add(lbMoveChar);
        lbMoveChar.setFont(new Font("黑体",Font.BOLD,20));
        lbMoveChar.setForeground(Color.yellow);
        this.init();
        ……
    }
    public void init(){//  字母的属性设置
        ……
        //  出现随机字母
        String str = String.valueOf((char)('A' + rnd.nextInt(26)));
        lbMoveChar.setText(str);
        lbMoveChar.setBounds(rnd.nextInt(this.getWidth()), 0, 20,20);
    }
    ……
    // Timer 事件对应的行为：实现掉下一个字母
    public void actionPerformed(ActionEvent e){
        ……
        lbMoveChar.setLocation(lbMoveChar.getX(),lbMoveChar.getY()+10);
    }
}
```

25.2.3 客户端如何实现加减分数

Note

由于本项目属于网络通信应用，因此，分数的加减可以通过服务器转发。方法如下：

1. 客户端输入正确，将自己加 2 分，然后将一个字符串“-1”发给服务器。
2. 服务器将“-1”发给所有在线客户端。
3. 所有客户端(包括自己)，获取“-1”之后，将相应的生命值减去 1 分。

代码如下：

```
……
public class GamePanel extends JPanel…… {
    // 生命值
    private int life = 10;
    // 按键按下的字母
    private char keyChar;
    // 掉下来的字母 Label
    private JLabel lbMoveChar = new JLabel();
    // 当前生命值状态显示 Label
    private JLabel lbLife = new JLabel();

    private Socket s = null;
    private Timer timer = new Timer(100,this);
    private Random rnd = new Random();
    private BufferedReader br = null;
    private PrintStream ps = null;
    private boolean canRun = true;
    ……
    //线程读取网络信息
    public void run() {
        try {
            while (canRun) {
                String str = br.readLine();// 读
                int score = Integer.parseInt(str);
                life += score;
                checkFail();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
```

Note

```
                javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
                System.exit(0);
            }
        }
        ……
        //键盘事件
        public void keyPressed(KeyEvent e){
            keyChar = e.getKeyChar();
            String keyStr = String.valueOf(keyChar).toUpperCase();
            try{
                if(keyStr.equals(lbMoveChar.getText())){
                    //注意，这里加 2 分，然后发送“-1”给所有客户端
                    //本客户端又会收到，结果为加 1 分
                    life += 2;
                    ps.println("-1");
                }else{
                    life --;
                }
                checkFail();
            }catch(Exception ex){
                canRun = false;
                javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
                System.exit(0);
            }
        }
    ……
}
```

25.2.4 客户端如何判断输了

判断输了，很简单，只需要判断生命值是否小于等于 0 即可：

```
……
public class GamePanel extends JPanel…… {
    ……
    public void checkFail(){
        init();
        if(life<=0){
            timer.stop();
```

Note

```
                javax.swing.JOptionPane.showMessageDialog(this,
                        "生命值耗尽，游戏失败！");
                System.exit(0);
            }
        }
        ……
}
```

最终设计出来的项目结构如下：

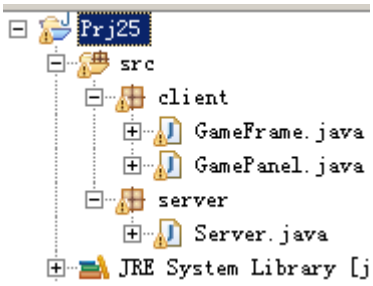


图 25-7

25.3 代码编写

25.3.1 服务器端

首先是服务器类，和前面讲解的服务器比较类似：

Server.java

```
package server;
import java.awt.Color;
import java.io.*;
import java.net.*;
import java.util.ArrayList;
import javax.swing.JFrame;
public class Server extends JFrame implements Runnable{
    private Socket s = null;
    private ServerSocket ss = null;
    //保存客户端的线程
    private ArrayList<ChatThread> clients = new ArrayList<ChatThread>();
    public Server() throws Exception{
        this.setTitle("服务器端");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Note

```
        this.setBackground(Color.yellow);
        this.setSize(200,100);
        this.setVisible(true);
        ss = new ServerSocket(9999);//服务器端开辟端口，接受连接
        new Thread(this).start(); //接受客户连接的死循环开始运行
    }
    public void run(){
        try{
            while(true){
                s = ss.accept();
                ChatThread ct = new ChatThread(s);
                clients.add(ct);
                ct.start();
            }
        }catch(Exception ex){
            ex.printStackTrace();
            javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
            System.exit(0);
        }
    }
    class ChatThread extends Thread{//为某个 Socket 负责接受信息
        private Socket s = null;
        private BufferedReader br = null;
        private PrintStream ps = null;
        private boolean canRun = true;
        public ChatThread(Socket s) throws Exception    {
            this.s = s;
            br = new BufferedReader(
                    new InputStreamReader(s.getInputStream()));
            ps = new PrintStream(s.getOutputStream());
        }
        public void run(){
            try{
                while(canRun){
                    String str = br.readLine();//读取该 Socket 传来的信息
                    sendMessage(str);  //将 str 转发给所有客户端
                }
```

Note

```
                }catch(Exception ex){
                    //此处可以解决客户异常下线的问题
                    canRun = false;
                    clients.remove(this);
                }
            }
        }
        //将信息发给所有其他客户端
        public void sendMessage(String msg){
            for(ChatThread ct : clients){
                ct.ps.println(msg);
            }
        }
        public static void main(String[] args) throws Exception{
            Server server = new Server();
        }
}
```

25.3.2 客户端

首先是游戏面板类：

GamePanel.java

```
package client;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
import java.util.Random;
import javax.swing.*;
public class GamePanel extends JPanel
        implements ActionListener,KeyListener,Runnable {
    // 生命值
    private int life = 10;
    // 按键按下的字母
    private char keyChar;
    // 掉下来的字母 Label
    private JLabel lbMoveChar = new JLabel();
    // 当前生命值状态显示 Label
```

Note

```
private JLabel lbLife = new JLabel();
private Socket s = null;
private Timer timer = new Timer(100,this);
private Random rnd = new Random();
private BufferedReader br = null;
private PrintStream ps = null;
private boolean canRun = true;
public GamePanel(){// 构造器
    this.setLayout(null);
    this.setBackground(Color.DARK_GRAY);
    this.setSize(240,320);

    this.add(lbLife);
    lbLife.setFont(new Font("黑体",Font.BOLD,20));
    lbLife.setBackground(Color.yellow);
    lbLife.setForeground(Color.PINK);
    lbLife.setBounds(0,0,this.getWidth(),20);

    this.add(lbMoveChar);
    lbMoveChar.setFont(new Font("黑体",Font.BOLD,20));
    lbMoveChar.setForeground(Color.yellow);
    this.init();
    this.addKeyListener(this);
    try {
        s = new Socket("127.0.0.1", 9999);
        JOptionPane.showMessageDialog(this,"连接成功");
        InputStream is = s.getInputStream();
        br = new BufferedReader(new InputStreamReader(is));
        OutputStream os = s.getOutputStream();
        ps = new PrintStream(os);
        new Thread(this).start();
    } catch (Exception ex) {
        javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
        System.exit(0);
    }
    timer.start();
}
```

Note

```
public void init(){// 字母的属性设置
    lbLife.setText("当前生命值:" + life);
    // 出现随机字母
    String str = String.valueOf((char)('A' + rnd.nextInt(26)));
    lbMoveChar.setText(str);
    lbMoveChar.setBounds(rnd.nextInt(this.getWidth()), 0, 20,20);
}
public void run() {
    try {
        while (canRun) {
            String str = br.readLine();// 读
            int score = Integer.parseInt(str);
            life += score;
            checkFail();
        }
    } catch (Exception ex) {
        canRun = false;
        javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
        System.exit(0);
    }
}
// Timer 事件对应的行为：实现掉下一个字母
public void actionPerformed(ActionEvent e){
    if(lbMoveChar.getY()>=this.getHeight()){
        life--;
        checkFail();
    }
    lbMoveChar.setLocation(lbMoveChar.getX(),lbMoveChar.getY()+10);
}
public void checkFail(){
    init();
    if(life<=0){
        timer.stop();
        javax.swing.JOptionPane.showMessageDialog(this,
                "生命值耗尽，游戏失败！");
        System.exit(0);
    }
```

Note

```
        }
        // 键盘操作事件对应的行为
        public void keyPressed(KeyEvent e){
            keyChar = e.getKeyChar();
            String keyStr = String.valueOf(keyChar).toUpperCase();
            try{
                if(keyStr.equals(lbMoveChar.getText())){
                    //注意，这里加 2 分，然后发送“-1”给所有客户端
                    //本客户端又会收到，结果为加 1 分
                    life += 2;
                    ps.println("-1");
                }else{
                    life --;
                }
                checkFail();
            }catch(Exception ex){
                ex.printStackTrace();
                javax.swing.JOptionPane.showMessageDialog(this,"游戏异常退出!");
                System.exit(0);
            }
        }
        public void keyTyped(KeyEvent e){}
        public void keyReleased(KeyEvent e){}
}
```

接下来是面板所在的界面类：

GameFrame.java

```
package client;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class GameFrame extends JFrame{
    private GamePanel gp;
    public GameFrame(){//构造器
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        String nickName = JOptionPane.showInputDialog("输入昵称");
        this.setTitle(nickName);
        gp = new GamePanel();
        this.add(gp);
```

```
        //获取焦点
        gp.setFocusable(true);
        this.setSize(gp.getWidth(), gp.getHeight());
        this.setResizable(false);
        this.setVisible(true);
    }
    //主函数入口：
    public static void main(String[] args){
        new GameFrame();
    }
}
```

运行该服务器，再运行客户端类，则可以进行网络游戏对战。

Note

Note

```
    public static String serverIP;
    public static int port;
    private static void loadConf() throws Exception{
        Properties pps = new Properties();
        pps.load(new FileReader("net.conf"));
        serverIP = pps.getProperty("serverIP");
        port = Integer.parseInt(pps.getProperty("port"));
    }
    public static void main(String[] args) throws Exception{
        loadConf();
        new LoginFrame();
    }
}
```

运行该类，即可运行客户端。

30.5 思考题

本程序开发完毕，留下几个思考题，请大家思考：

1. 如何传送文件，特别是大文件？
2. 如何像 QQ 一样，将用户界面和聊天界面分开？如：

图 30-17

3. 如何将软件在任务栏右下角显示为一个图标？如同 QQ 一样：

图 30-18

4. 能否用 UDP 编程来实现本章案例？

Note