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：

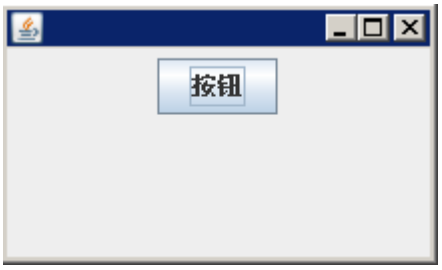

图 15-5

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：

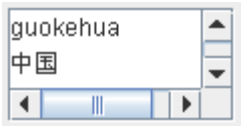

图 15-9

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：

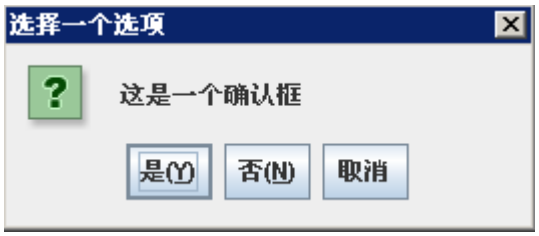

图 15-19

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：

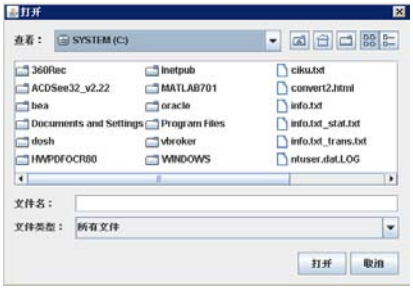

图 15-20

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：

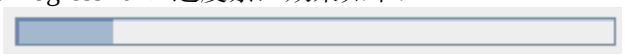

图 15-24

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：

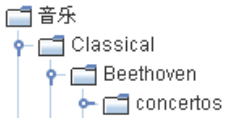

图 15-26

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：

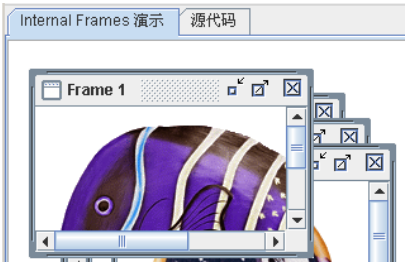

图 15-29

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：

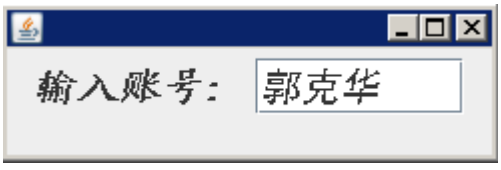

图 15-31

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。

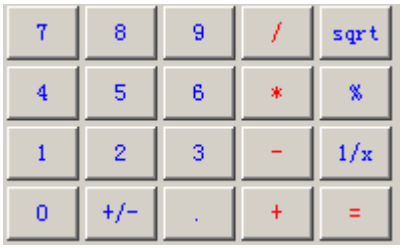

图 16-5

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：

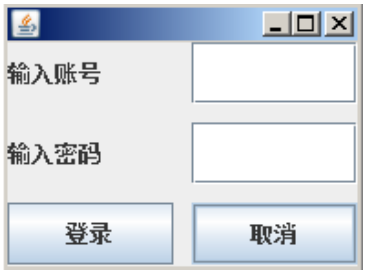

图 16-6

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：

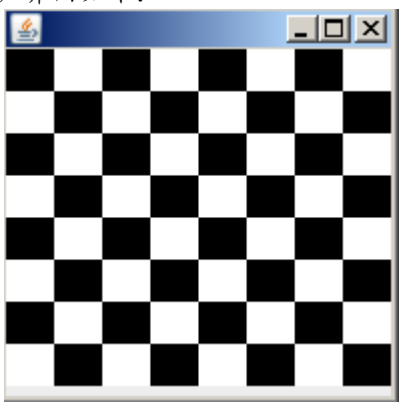

图 16-8

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。

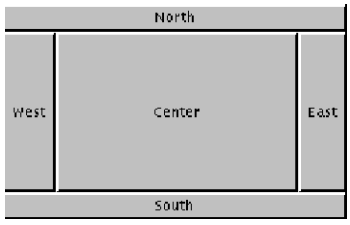

图 16-9

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：

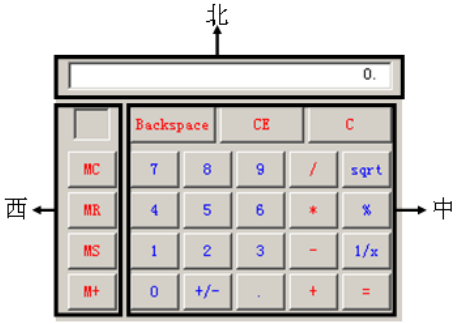

图 16-10

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：

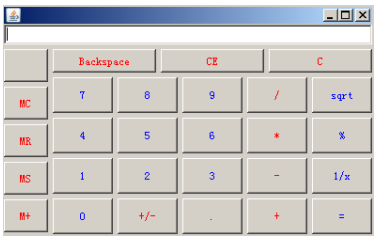

图 16-12

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：

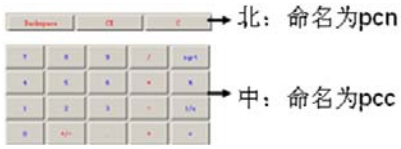

图 16-14

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	★★★	★★