求是科技

实效编程百例

Java 实效编程百例

★求是科技　潘传邦　杨瑞峰　王建军　编著

★学习编程技巧

★积累编程经验

★剖析功能模块

★突出应用实效

源代码光盘
CD-ROM

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

实效编程百例

Java 实效编程百例

★求是科技　潘传邦　杨瑞峰　王建军　编著

人民邮电出版社

图书在版编目（CIP）数据

Java 实效编程百例 / 求是科技编著. —北京：人民邮电出版社，2003.4
ISBN 7-115-10795-5

Ⅰ. J… Ⅱ. 求… Ⅲ. JAVA 语言—程序设计 Ⅳ. TP312
中国版本图书馆 CIP 数据核字（2003）第 015137 号

内容提要

本书通过 100 多个精选的实例讲解了利用 Java 进行应用程序开发的各个方面，涵盖了控件、界面、多媒体控制、图像处理、操作系统、磁盘文件、数据库、网络应用、邮件和通信、Java Beans、国际化和本地化等方面的内容。

本书内容突出了实用性，85%以上的实例模仿较常见的优秀软件的相关功能，余下实例大多为帮助读者理解重点、难懂概念所做。本书的另一个特点在于给出了不少实用性很强的“方案实例”（以往的百例图书内容基本上都属于“功能实例”），其内容多为典型或通用的功能模块的解决方案，包括界面设计、操作流程以及代码控制等内容。

本书适用于已经初步掌握 Java 编程概念、方法的读者阅读，本书可以帮助读者迅速掌握实际应用中的各种经验、技巧。

实效编程百例

Java 实效编程百例

◆ 编　　著　求是科技　潘传邦　杨瑞峰　王建军
责任编辑　张立科

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132692
北京汉魂图文设计有限公司制作
北京密云春雷印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：21.25
字数：659 千字　　2003 年 4 月第 1 版
印数：1-6 000 册　　2003 年 4 月北京第 1 次印刷

ISBN7-115-10795-5/TP・3174

定价：35.00 元（附光盘）

前 言

精确掌握编程语言的语法概念并不意味着可以开发功能强大的应用程序，相反这才是万里长征的第一步。编程水平的提高需要在实际应用中积累点滴经验，本书通过100多个新颖别致的应用实例，详细讲解了如何利用Java的强大功能以及API函数开发应用程序。希望能为读者提高Java的编程水平有一些帮助。

本书实例按照功能来分类，尽量将相关的内容归纳在一章中。通过每章的例子讲解控件、界面、多媒体控制、图像处理、操作系统、磁盘文件、数据库、网络应用、邮件和通信、Java Beans、国际化和本地化等内容，使读者对计算机应用的各个方面有所了解。

本书强调实用性，85%以上的实例模仿较常见的优秀软件的相关功能，余下实例大多为帮助读者理解重点、难懂概念所做。本书的另一个特点在于给出了不少实用性很强的“方案实例”，其内容多为典型或通用的功能模块的解决方案，包括界面设计、操作流程以及代码控制等内容。

每个实例的讲解分为3个步骤：

- 实例目的——讲解本例的功能所在，指出本例要到达的目的和效果，让读者做到心中有数。
- 实现方法——讲解技术原理/设计思路，给出技术原理的合理解释、规范的算法和流程描述，便于读者阅读代码、学习程序设计方法。
- 程序代码——给出具体的实现过程，包括界面设计、编写代码和注释，读者可参照实现。

本书的主要作者包括潘邦传、杨瑞峰、王建军等。此外，以下人员也参与了本书的资料搜集和写作工作，他们是王锐、王凯峰、刘冰玉、杨柯岚、王东、杨珏、赵昊彤、程凡、程卫峰、范桂山、赵微微、宋征、吴频、严庆子、文华、袁玎、岳进、钟明、黄成昆、王远、吴晓超、肖永顺、钱力鹏、马小来、李松、王琴芳、王国红等。以上人员对本书的顺利完成付出了辛勤的汗水和心血，在此一并表示感谢。

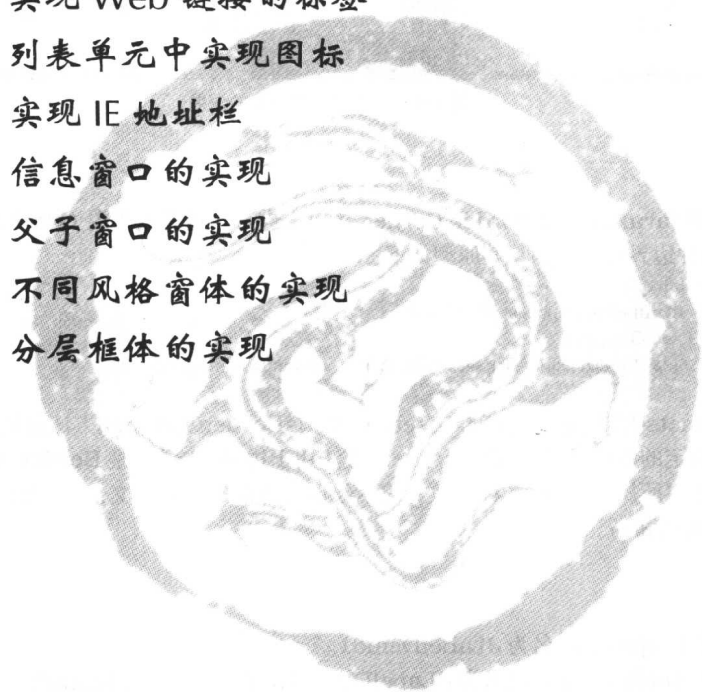
由于时间、水平限制，缺点和不足之处在所难免，敬请读者批评指正。

编者

目 录

第 1 章　控件与界面

- 自定义标签的边界效果
- 实现 Web 链接的标签
- 列表单元中实现图标
- 实现 IE 地址栏
- 信息窗口的实现
- 父子窗口的实现
- 不同风格窗体的实现
- 分层框体的实现



实例 1　自定义标签的边界效果

实例目的

Swing 中的标签是由 JLabel 类对象所表示的一个轻量级的组件，不能获得键盘输入焦点，主要用于显示一些简单的文字和图标。javax.swing.border 包中提供了一些用来美化 Swing 标签边界效果的类，如空边界（EmptyBorder）、斜切边界（BevelBorder）、蚀刻边界（EtchedBorder）、线条边界（LineBorder）、带标题的边界（TitledBorder）和复合边界（CompoundBorder）等边界效果。本例中，我们将编程实现自己定义标签的边界效果。如图 1-1 所示，窗体中显示了本例实现的几种自定义的标签边界效果。

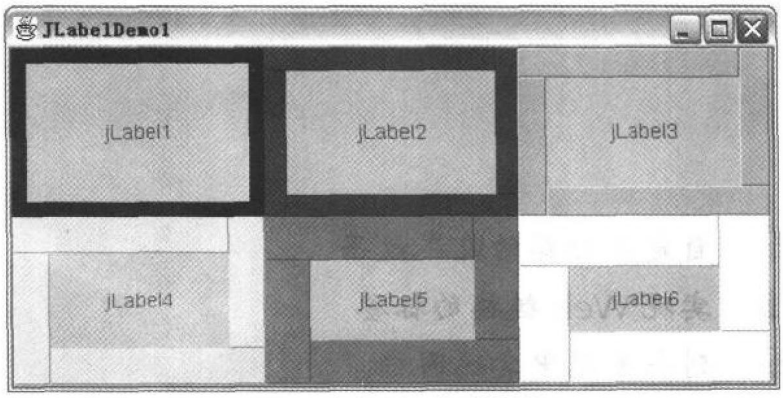


图 1-1　自定义标签边界的效果

实现方法

本例通过实现 javax.swing.border.Border 接口，进行自定义边界类 OwnBorder。Border 接口描述了 Swing 组件边界如何绘制，其原型如下：

```
public interface Border {
    Insets getBorderInsets(Component c);
    boolean isBorderOpaque();
    void paintBorder(Component c, Graphics g, int x, int y, int width, int height);
}
```

getBorderInsets()方法返回边界的 Insets 对象。isBorderOpaque()方法返回边界是不是透明，即边界是不是按照指定的颜色显示。paintBorder()方法用于边界的绘制，是 Border 接口中最重要的方法。其中参数 c 表示边界所属的组件对象，参数 g 表示绘制边界的图形上下文，x、y 表示边界的坐标，width、height 表示边界的宽和高。

程序代码

1. 新建一个 Project，取名为 JLabelDemo1。
2. 新建一个 Application，取名为 JLabelDemo1；主窗口取名为 MainFrame（也就是主窗口的类名），标题为 JLabelDemo1。
3. 新建一个新类 OwnBorder，并实现 javax.swing.border.Border 接口。代码如下：

```
import java.awt.*;
import javax.swing.border.*;

class OwnBorder implements javax.swing.border.Border
```

```
{
    private int thickness; //边界线条的厚度
    private Color color;   //边界的颜色

    /*构造方法
    *传入两个参数：边界的厚度和颜色
    */
    public OwnBorder(int thickness,Color color){
        this.thickness=thickness;
        this.color=color;
    }
    /*实现 Border 接口中的第一个方法
    *该方法用于绘制自定义的边界
    */
    public void paintBorder(Component c,     //边界所属的组件对象
                            Graphics g,  //得到图形上下文引用，用于绘制
                            int x, int y,  //边界的坐标
                            int width, int height) {   //边界的宽度、长度
        g.setColor(this.color);                          //设定颜色
        g.fill3DRect(x,y,width-thickness,thickness,true); //绘制上边界
        g.fill3DRect(x,y+thickness,thickness,height-thickness,true); //绘制左边界
        g.fill3DRect(x+thickness,y+height-thickness,width-thickness,thickness,true); //绘制下边界
        g.fill3DRect(x+width-thickness,y,thickness,height-thickness,true); //绘制右边界
    }
    /*实现 Border 接口的第二个方法
    *返回一个 Insets 对象的引用
    */
    public Insets getBorderInsets(Component c){
        return new Insets(thickness,thickness,thickness,thickness);
    }
    /*实现 Border 接口的第三个方法
    *这里返回 true,表明按照制定的颜色显示边界
    */
    public boolean isBorderOpaque(){
        return true;
    }
}
```

4. 向 MainFrame 设计窗口中添加 6 个 JLabel 组件，并设置 contentPane 对象的布局结构为 GridLayout 类的实例

```
public class MainFrame extends JFrame {
.....
  private GridLayout gridLayout1 = new GridLayout();
  private JLabel jLabel1 = new JLabel();
  private JLabel jLabel2 = new JLabel();
  private JLabel jLabel3 = new JLabel();
  private JLabel jLabel4 = new JLabel();
  private JLabel jLabel5 = new JLabel();
  private JLabel jLabel6 = new JLabel();
.....
}
```

5. 在 jbInit()方法中设置 GridLayout、JLabel 对象的属性，并添加 JLabel 对象的边界为 OwnBorder 类的实例

```
  private void jbInit() throws Exception  {
    .....
    //设置 gidLayout1 的属性 2 行 3 列
    gridLayout1.setColumns(3);
    gridLayout1.setRows(2);
    //设置 JLabel 对象的属性
    jLabel1.setFont(new java.awt.Font("Dialog", 0, 14));
```

```
jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
jLabel1.setText("jLabel2");
jLabel2.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel2.setHorizontalAlignment(SwingConstants.CENTER);
jLabel2.setText("jLabel2");
jLabel3.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel3.setHorizontalAlignment(SwingConstants.CENTER);
jLabel3.setText("jLabel3");
jLabel4.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel4.setHorizontalAlignment(SwingConstants.CENTER);
jLabel4.setText("jLabel4");
jLabel5.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel5.setHorizontalAlignment(SwingConstants.CENTER);
jLabel5.setText("jLabel5");
jLabel6.setFont(new java.awt.Font("Dialog", 0, 14));
jLabel6.setHorizontalAlignment(SwingConstants.CENTER);
jLabel6.setText("jLabel6");

//是设置 JLabel 对象的边界
jLabel1.setBorder(new OwnBorder(10,Color.blue));
jLabel2.setBorder(new OwnBorder(15,Color.red));
jLabel3.setBorder(new OwnBorder(20,Color.orange));
jLabel4.setBorder(new OwnBorder(25,Color.yellow));
jLabel5.setBorder(new OwnBorder(30,Color.green));
jLabel6.setBorder(new OwnBorder(35,Color.white));
.....
}
```

实例 2　实现 Web 链接的标签

实例目的

一些软件的帮助菜单中的“关于”对话框中有链接到软件公司的标签。本例将实现具有 Web 链接功能的标签。如图 1-2 所示，窗体中就是一个 Web 链接标签，单击此标签，可以打开 IE（或其他浏览器，但必须是系统默认的网页浏览器）并链接至指定的网址。

图 1-2　实现 Web 链接的标签

实现方法

运用 java.lang 包中的 Runtime 类可以实现标签的 Web 链接功能。Runtime 类提供了大量的方法用于应用程序与其运行环境的交互。本例要使用的方法原型如下：

```
static Runtime getRuntime();
Process exec(String command);
```

因为 Runtime 类中的 getRuntime()方法是静态方法，所以我们不必创建 Runtime 类的实例，直

接调用 Runtime.getRuntime()就可以得到一个 Runtime 对象的引用。接着，调用 exec()方法就可以执行参数 command 的命令。如果是 Windows NT、Windows 2000 和 Windows XP 操作系统，则 command = "cmd /c start http://www.pku.edu.cn"；如果是 Windows 98 或 Windows ME 操作系统，则只须将 cmd 改为 command 即可。具体实现代码如下：

```
Runtime.getRuntime.exec("cmd /c start http://www.pku.edu.cn");
```

程序代码

1. 新建一个 Project，取名为 JLabelDemo2。
2. 新建一个 Application，取名为 JLabelDemo2；主窗口取名为 MainFrame，标题为 JLabelDemo2。
3. 向 MainFrame 的设计窗口中，添加一个 JLabel 组件，并设置相关属性。

```
public class MainFrame extends JFrame {
......
    private JLabel jLabel1 = new JLabel();
......
}
private void jbInit() throws Exception  {
......
    //jLabel1 对象的基本属性设置
    jLabel1.setToolTipText("北京大学");
    jLabel1.setDisplayedMnemonic('0');
    jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel1.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR)); //当鼠标位于标签上时呈手形
    jLabel1.setText("<html><body><font size=10 ><a href=http://www.pku.edu.cn>北 京 大 学
                    </a></font></body></html>"); //为使得文字底下有横线，更像一个链接
   //添加鼠标监听器，当 mouseReleased 时触发事件
    jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {
      public void mouseReleased(MouseEvent e) {
        jLabel1_mouseReleased(e);
      }
    });
......
}
```

4. 编写 jLabel1 对象的鼠标事件处理方法。

```
void jLabel1_mouseReleased(MouseEvent e) {
    try{
      Runtime.getRuntime().exec("cmd /c start http://www.pku.edu.cn");   //打开链接的网页
    }catch(Exception err)
    {
      err.printStackTrace();
    }
  }
```

实例 3 列表单元中实现图标

实例目的

列表组件我们已经司空见惯了，通常在列表单元中的大都是文本，本例将制作一个可以在列表单元中显示图标的小程序。如图 1-3 所示，窗体中列表组件的每个单元既有图标又有文本。

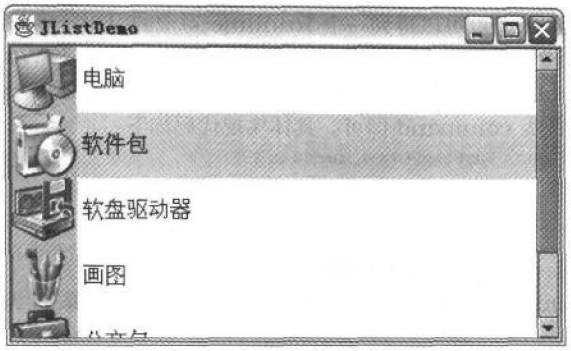


图 1-3 列表中实现图标

实现方法

本例通过实现 javax.swing.ListCellRenderer 接口来创建自定义的列表单元渲染类 OwnCellRenderer，并且为了使列表单元同时显示图标和文本，OwnCellRenderer 同时将继承 JLabel 类。如图 1-3 所示，图标显示在左、文本在右（当然，按照我们对 JLabel 的了解，图标文本的位置是可随意换的）。

ListCellRenderer 接口只提供了一个 getListCellRendererComponent()方法。因此，类 OwnCellRenderer 只需实现一个方法。方法的原型如下：

```
public Component getListCellRendererComponent(JList list,Object value,int index,Boolean isSelected,Boolean cellHasFocus);
```

该方法返回的是一个 Component 对象（即列表单元渲染对象），本例中就是 JLabel 对象，所以将返回 this 对象。

getListCellRendererComponent（）方法的参数 list 表示使用此列表单元渲染对象的列表对象，有了这个引用，就可以使用列表（JList）的方法，如 list.getSelectionBackground(),list.getSelection Foreground()等。

参数 value 表示 list.getModel().getElementAt(index)返回的对象，本例中即列表单元的文本 String 对象；参数 index 表示列表单元的序数；参数 isSelected 表示指定的列表单元是否被选中；参数 cellHasFocus 表示指定的列表单元是否拥有焦点。

另外，本例中调用 java.lang.ClassLoader 类中的 getSystemResource()来引用列表单元中所需的图片。该方法原型如下：

```
static URL getSystemResource(String  name);
```

此方法是静态方法，故不必创建 ClassLoader 类的实例，直接调用 ClassLoader.getSystemResource(String name)即可返回（类文件所在目录）\（name 所指地址）的 URL。参数 name 可以是目录，也可以是文件。例如：class 文件在 c:\myProjects\JListDemo 目录下，而要引用 1-3-1.jpg 文件在此目录的子目录 images 下，则 1-3-1.jpg 文件的 URL 可调用如下代码得到：

```
URL  url=ClassLoader.getSystemResource("images/1-3-1.jpg");
```

程序代码

1. 新建一个 Project，取名为 JListDemo。
2. 新建一个 Application，取名为 JListDemo；主窗口取名为 MainFrame，标题为 JListDemo。
3. 新建一个新类 OwnCellRenderer，继承 javax.swing.JLabel，并利用 wizards|implements 实现 javax.swing.ListCellRenderer 接口，自动实现抽象方法。向 OwnCellRenderer 类中添加新的属性。

```
import java.awt.*;
import javax.swing.*;
```

```
class OwnCellRenderer extends JLabel implements ListCellRenderer
{
.....
    Icon[] icons;    // 列表中显示的图标数组
.....
}
```

4. 编写 OwnCellRenderer 类的构造方法。

```
/*构造方法，完成初始化工作*/
    public OwnCellRenderer(Icon[] icons){

        this.icons = icons;
        this.setOpaque(true);
        this.setFont(new Font("Dialog",Font.PLAIN,14));
    }
```

5. 编写 OwnCellRenderer 类唯一要实现的 getListCellRendererComponent（）方法。

```
/*实现接口中的唯一一个方法*/
    public Component getListCellRendererComponent(
        JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus){
            if(value!=null){
                String text = value.toString();
                this.setText(text);      //设置显示文本
            }
            this.setIcon(icons[index]);    //在标签中设置图标
            if(isSelected){
                //如果列表单元被选中，设置背景颜色、前景色分别为列表单元选中的默认背景色、前景色
                this.setBackground(list.getSelectionBackground());
                this.setForeground(list.getSelectionForeground());
            }
            else{
                //如果列表单元没被选中，设置背景颜色、前景色分别为列表单元没被选中默认的颜色
                this.setBackground(list.getBackground());
                this.setForeground(list.getForeground());
            }

            return this; // 返回本对象为列表单元的渲染对象
        }
```

6. 在 MainFrame 类的设计窗口中添加一个 JList 组件，并在本类中添加其他新的属性。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class JListDemo extends JFrame {

    String[] iconnames={"电脑","软件包","软盘驱动器","画图","公文包","网上邻居"}; //列表单元显示的文本
    JList list;
    // 列表单元显示的图标
    //ClassLoader.getSystemResource()返回 class 文件的目录,再加入"images/1-3-1.jpg"表示,class 文件所在目录
    //下 images 子目录下的 1-3-1.jpg 文件。
    Icon icon1=new ImageIcon(ClassLoader.getSystemResource("images/1-3-1.jpg"));
    Icon icon2=new ImageIcon(ClassLoader.getSystemResource("images/1-3-2.jpg"));
    Icon icon3=new ImageIcon(ClassLoader.getSystemResource("images/1-3-3.jpg"));
    Icon icon4=new ImageIcon(ClassLoader.getSystemResource("images/1-3-4.jpg"));
    Icon icon5=new ImageIcon(ClassLoader.getSystemResource("images/1-3-5.jpg"));
    Icon icon6=new ImageIcon(ClassLoader.getSystemResource("images/1-3-6.jpg"));
.....
}
```

7. 向 MainFrame 类的初始化方法 jbInit()中添加代码，使用 OwnCellRenderer 类来实例化 list 对象。

```
/*构造方法，初始化工作*/
  private void jbInit() throws Exception  {
  .....
            Icon[] icons={icon1,icon2,icon3,icon4,icon5,icon6};
            list=new JList(iconnames);  //创建以 iconnames 数组元素为列表单元中文本的列表组件
            list.setCellRenderer(new OwnCellRenderer(icons)); // 设置 list 的单元渲染对象为自定义的
                                                              //OwnCellRenderer 对象
            this.getContentPane().add(new JScrollPane(list)); //将 list 对象放入到一个 JScrollPane 实例中
  ......
    }
```

实例 4　实现 IE 地址栏

实例目的

大家都用过 Microsoft Windows 自带的 Web 浏览器 IE。在 IE 的地址栏中输入网址，按下回车键就可以访问该网页（当然地址是正确的）。另外大家是否发现，用过的地址会在地址栏的下拉列表中记录下来，使下次访问时在下拉列表中选中此地址就行了。本例将使用 javax.swing 包中的 JComboBox 类来实现这样的地址栏。如图 1-4 所示，窗体中输入的网址（红色部分），会自动添加在下拉单中。

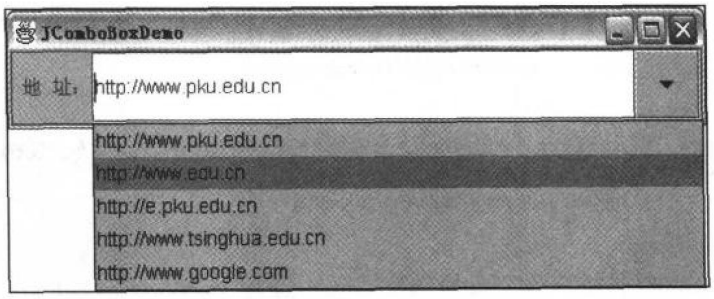


图 1-4　实现 IE 地址栏

实现方法

本例使用两个组件 JLabel 和 JComboBox 来构建地址栏。其中，JComboBox 对象的属性设为可编辑（默认是不可编辑的），另外还要实现 JComboBox 的 ActionEvent 事件处理程序。由于本例中使用 JComboBox 默认的编辑器，因此不用实现 javax.swing.ComboBoxEditor 创建自定义的编辑器类。

程序代码

1. 新建一个 Project，取名为 JComboBoxDemo。

2. 新建一个 Application，取名为 JComboBoxDemo；主窗口取名为 MainFrame，标题为 JComboBoxDemo。

3. 在 MainFrame 类的设计窗口中添加一个 JComboBox 组件，一个 JLabel 组件，并添加其他新的属性。

```
public class MainFrame extends JFrame {
  private JPanel contentPane;
  private BorderLayout borderLayout1 = new BorderLayout();
```

```
/*用于显示"地址："文本*/
  private JLabel jLabel1 = new JLabel();
/*下拉列表中默认得站点地址*/
  private String[] defaultSites={
        "http://www.pku.edu.cn",
        "http://www.edu.cn",
        "http://e.pku.edu.cn",
        "http://www.tsinghua.edu.cn",
        "http://www.google.com"
  };
/*/默认的编辑地址*/
  private String defaultEdit="http://www.pku.edu.cn";

/*创建一个 JComboBox 实例，并且每个单元的文本为 defaultSites 中的地址*/
  private JComboBox jComboBox1 = new JComboBox(defaultSites);
.....
}
```

4. 编写 MainFrame 的初始化方法 jbInit()、设置 jComboBox1 、jLabel1 的属性

```
  private void jbInit() throws Exception  {
    //setIconImage(Toolkit.getDefaultToolkit().createImage(MainFrame.class.getResource("[Your Icon]")));
    contentPane = (JPanel) this.getContentPane();

    //设置 jLabel1 的属性
    jLabel1.setFont(new java.awt.Font("Dialog", 0, 14));
    jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel1.setText("  地  址：");

    //设置 jComboBox1 的属性
    jComboBox1.setFont(new java.awt.Font("Dialog", 0, 14));
/*本例的关键一步
*JComboBox 对象的可编辑属性设为 true，这样才能输入地址 URL
*/
    jComboBox1.setEditable(true);
    jComboBox1.setMaximumRowCount(5); //设置下拉列表的最大长度为 5
    jComboBox1.configureEditor(jComboBox1.getEditor(),defaultEdit); //设置默认编辑器的文本为 defaultEdit 字符串
    ComboBoxEditor comboxEditor=jComboBox1.getEditor();//以默认编辑器来获得 ComboBoxEditor 接口的引用
    Component editorComp = comboxEditor.getEditorComponent();//获得编辑器组件，我们在这个组件中输入文本地址
    //设置编辑器组件的背景色、前景色和字体
    editorComp.setBackground(Color.white);
    editorComp.setForeground(Color.red);
    editorComp.setFont(new Font("Dialog",Font.PLAIN,14));

    //添加 jComboBox1 的 ActionListener,ComboxListener 是自定义实现 ActionListener 接口的内部类
    //当选中下拉列表中的某个单元时触发事件
    jComboBox1.addActionListener(new  ComboxListener())

    contentPane.add(jLabel1, BorderLayout.WEST);
    contentPane.add(jComboBox1, BorderLayout.CENTER);
  }
```

5. 编写一个类 JComboBoxDemo 的内部类 ComboxListener，实现 ActionListener 接口，来处理事件。代码如下：

```
class ComboxListener implements ActionListener{
      /*实现 ActionListener 接口唯一的方法*/
      public void actionPerformed(ActionEvent e){
            boolean isItemPresent = false;
      //判断选中的单元是不是现在的单元，如果不是将 isItemPresent 属性置为 false
            for(int i=0;i<combox.getItemCount();i++)
            {
                  if(combox.getItemAt(i).equals(combox.getSelectedItem())){
                        isItemPresent=false;
```

```
                    break;
                }
            }
            //如果 isItemPresent 为假，则将选中的单元的地址插入到 JComboBox 对象的最前单元
            if(!isItemPresent){
                combox.insertItemAt(combox.getSelectedItem(),0);
                isItemPresent=false;
            }
        }
    }
```

实例 5　信息窗口的实现

实例目的

在开始运行一个程序时，经常可以看到会先弹出一个有关该软件或软件公司的窗口，然后才会正式进入程序的主界面，如常用的 Microsoft Word、 Borland JBuilder 等，这样的窗口称为信息窗口。使用信息窗口的好处是可以使用户在等待软件主界面出现前的一段时间中得知软件运行状态。本例将演示如何来实现信息窗口，效果如图 1-5 所示。当打开程序时，信息窗口先显示，并在窗口上倒计时，直到“waiting 0”时，关闭该窗口，显示程序的主窗口。

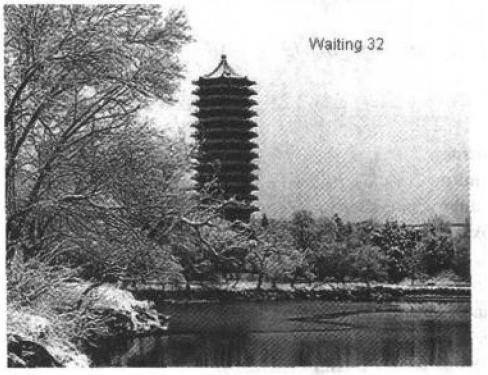


图 1-5　信息窗口的实现

实现方法

大多数的信息窗口是没有标题栏的，因此信息窗口不能由继承 JFrame 类来实现，一种简单的做法是通过继承 JWindow 来实现（当然继承 Window 类也可以，但一个原则是尽量使用 swing 中的界面类）。

另外，本例在加载图 1-5 所示的信息窗口上的图片时，用到 java.awt 包中的 MediaTracker 类。使用该类的好处是可以更好地管理程序中要使用的图片，同时还可以保证图片和界面同时显示，避免了窗口显示后很久才显示图片的缺点。

程序代码

1. 新建一个 Project，取名为 JSpleshWindowDemo，其他设置按默认值。

2. 新建一个 Application ，取名为 JSpleshWindowDemo，主窗口取名为 MainFrame，主窗口标题取名为 JSpleshWindowDemo。