

JC2002 Java 程序设计

第 7 天：图形用户界面（人工智能、计算机科学与技术）

11 月 8 日星期三/11 月 9 日星期四

JC2002 Java 程序设计

第 7 天，第 1 课时：图形用户界面介绍

参考文献和学习目标

- 今天的会议主要基于
 - Sierra 等人，《**Head First Java**》（第 2 版），O'Reilly，第 12、13 章
(可在图书馆获取)
 - <https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>
 - **Java 编程入门**》，第 2.14 章
<https://runestone.academy/runestone/books/published/csjava/index.html>
- 今天的课程结束后，您应该能够
 - 解释图形用户界面的主要概念和术语，以及

- 使用标准 Swing 组件实现简单的用户界面
- 为不同用途选择合适的图形用户界面组件

图形用户界面 (GUI)

- 图形用户界面 (GUI) 提供了一种方便用户的机制用于与应用程序互动
 - GUI (发音为 "GOO-ee") 赋予应用程序独特的外观和感觉
 - 图形用户界面为应用程序提供一致、直观的用户界面组件，让用户即使在使用新的应用程序时也有有一种熟悉感
- 图形用户界面由图形用户界面组件 (也称为控件或部件) 构建而成 (窗口小工具)
 - 图形用户界面组件是用户通过鼠标、键盘或其他输入方式，如语音识别

不同平台上的 Java 图形用户界面

- Java 代码与平台无关: Java 图形用户界面使用图形用户界面底层平台提供的组件
 - 不同的平台有不同的外观和感觉



Mac



OSLinux OS



Solaris (Unix) 操作
系统



Windows 操作系统

Java 图形用户界面库

- Java 有不同的图形用户界面库：
 - **抽象窗口工具包 (AWT)** 是 Java 最初的图形用户界面库（在所有 Java 图形用户界面中历史最悠久）。
 - AWT 是重量级的，依赖于平台
 - **Swing** 在 Java SE 1.2 中被添加到平台中
 - 直到最近，Swing 仍是主要的 Java 图形用户界面技术
 - 更轻便、不依赖平台、纯粹用于台式机
 - **JavaFX** 宣布于 2007 年，并于 2008 年发布，其竞争对手是

Adobe Flash 和 Microsoft Silverlight

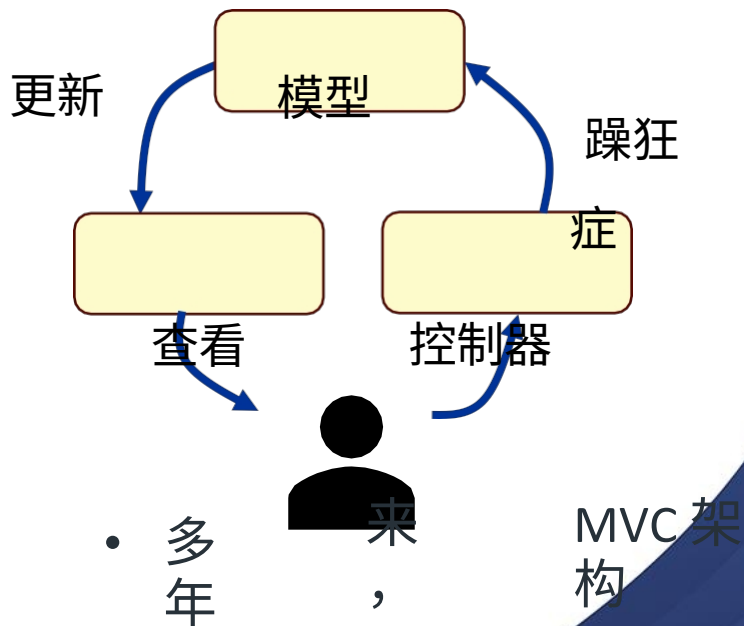
- 组件数量更少，与现代设备的集成度更高

Java Swing 库

- 在本课程中，我们主要使用 Swing 来制作图形用户界面
 - Swing 仍在广泛使用，重点是
 - Swing 在桌面应用程序中承担 "重任"
 - Swing 使用常见的模型-视图-控制器（MVC）设计模式
 - Swing 是跨平台的（如同 Java 一般），具有合适的外观和感觉
 - Swing 内容广泛，从 Swing 学到的知识将来很容易迁移到 JavaFX 中。

模型-视图-控制器 (MVC)

- 一般来说，视觉组件是指由三个不同方面组成：
 - 组件在屏幕上呈现的样子 (*视图*)
 - 组件对用户做出反应的方式 (*控制器*)
 - 与组件



被证明是异常有效的

看到

用途

用户

Java 基础类 (JFC) 和 Swing

- JFC 包含一组功能，用于构建图形用户界面和添加丰富的为 Java 应用程序提供图形功能和交互性
- JFC 包含这些功能：
 - Swing 图形用户界面组件
 - 可插拔外观和触感支持
 - 无障碍 API
 - Java 2D API
 - 国际化

摇摆组件

- 在 Swing 中，图形用户界面由图形部件
 - 图形组件是类，要使用它们，必须声明其对象
 - 通常，图形用户界面建立在 **JFrame** 组件上，该组件是图形用户界面的 *窗口或容器*
 - 其他常用组件包括 **JLabel**（可包含静态

文本或图像)
和
JButton

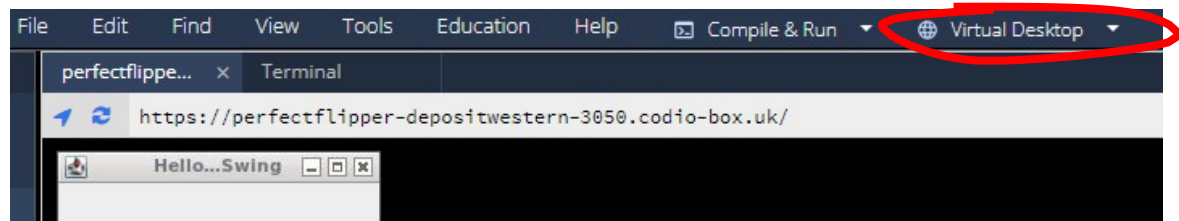
on（实现可按下的按钮）等。

JLabel



在 Codio 中运行 Swing

- 要使用图形用户界面运行 Java，底层平台必须支持图形用户界面
- 在 Codio 中，您可以使用*虚拟桌面*运行图形用户界面
 - 要安装，请按照以下说明进行：
<https://docs.codio.com/common/develop/ide/boxes/installsw/gui.html>
 - 请注意，在使用虚拟桌面之前，您需要重新启动 Codio
 - 然后您就可以在 Codio 中打开 "虚拟桌面" 选项卡了



使用 JFrame 的示例 (1)

```
1  import javax.swing.*;
2  公共类 FrameTest {
3      私人静态 void createAndShowGUI() {
4
5          JFrame frame = new JFrame("HelloWorldSwing");
6          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7
8          JLabel label = new JLabel("Hello World");
9          frame.getContentPane().add(label);
10
11         frame.pack();
12         frame.setVisible(true);
13     }
14     public static void main(String[] args) {
15         javax.swing.SwingUtilities.invokeLater(new Runnable() {
16             public void run() {
17                 createAndShowGUI();
18             }
19         });
20     }
21 }
```

使用 JFrame 的示例 (2)

```
1  import javax.swing.*;  
2  public class FrameTest {  
3      私人静态 void createAndShowGUI() {  
4  
5  
6      JFrame frame = new JFrame("HelloWorldSwing");  
7  
8      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
9  
10     JLabel label = new JLabel("Hello World");  
11  
12     frame.getContentPane().add(label);  
13  
14     frame.pack();  
15     frame.setVisible(true);  
16  
17 }  
18 public static void main(String[] args) {  
19     javax.swing.SwingUtilities.invokeLater(new Runnable() {  
20         public void run() {  
21             createAndShowGUI();  
22         }  
23     })  
24 }
```

导入 Swing 类

创建 JFrame 组件

创建 JLabel 组件

运行 Swing 图形用户界面

```
}  
  }  
};
```

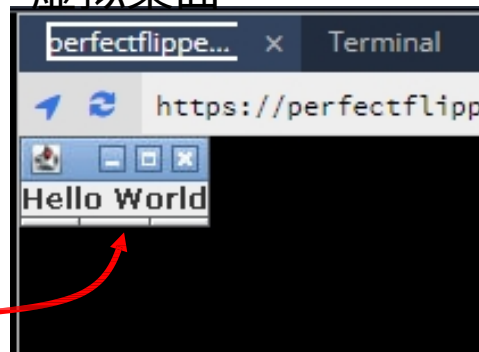
使用 JFrame 的示例 (3)

```
1  import javax.swing.*;
2  公共类 FrameTest {
3      私人静态 void createAndShowGUI() { 4
5      JFrame frame = new JFrame("HelloWorldSwing");
6      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7
8      JLabel label = new JLabel("Hello World");
9      frame.getContentPane().add(label);
10
11     frame.pack();
12     frame.setVisible(true);
13 }
14 public static void main(String[] args) {
15     javax.swing.SwingUtilities.invokeLater(new Runnable() {
16         public void run() {
17             createAndShowGUI();
18         }
19     });
20 }
```

控制台

```
$ javac FrameTest.java
$ java FrameTest
```

虚拟桌面



您可以使用鼠标调整应用程序的大小

```
}  
});  
}
```

设置框架尺寸

- 您需要确定框架的大小
 - 您可以使用 `frame.pack()` 将决定权委托给应用程序
- 包装方法确定框架的尺寸，使其所有内容物都达到或超过其首选尺寸
- `pack()` 的另一种方法是通过调用 `setSize()` 或 `setBounds()`（同时设置框架位置）
- 一般来说，使用 `pack()` 比调用 `setSize()` 更为可取，因为 `pack()` 布局管理器善于根据平台依赖性和其他影响组件大小

的因素进行调整

使用 setSize 的示例

```
1  import javax.swing.*;
2  公共类 SetSizeTest {
3      私人静态 void createAnd 4
5      JFrame frame = new JFrame("H
6      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7
8      JLabel label = new JLabel("Hello World");
9      frame.getContentPane().add(label);
10
11     frame.setSize(100,100);
12     frame.setVisible(true);
13 }
14 public static void main(String[] args) {
15     javax.swing.SwingUtilities.invokeLater(new Runnable() {
16         public void run() {
17             createAndShowGUI();
18         }
19     });
20 }
21 }
```

使用 setSize() 设置
框架大小

控制台

虚拟桌面



100 个单位

```
$ javac SetSizeTest.java  
$ java SetSizeTest
```

有问题或意见？

JC2002 Java 程序设计

第 7 天，课程 2：在 Swing 中使用顶级容器

顶级容器类

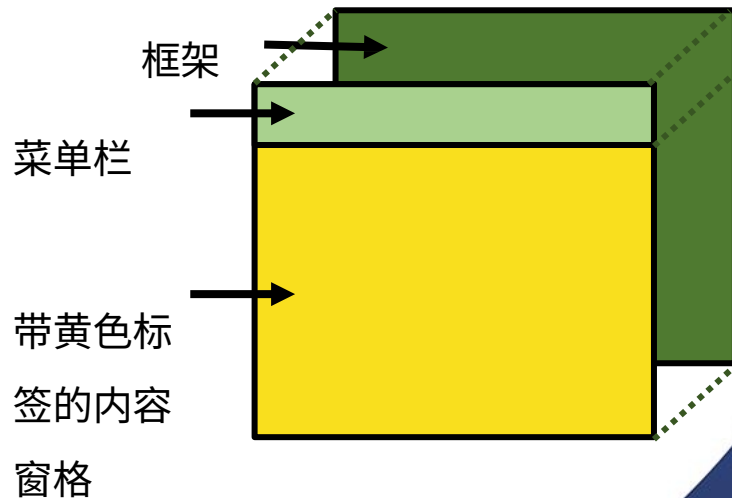
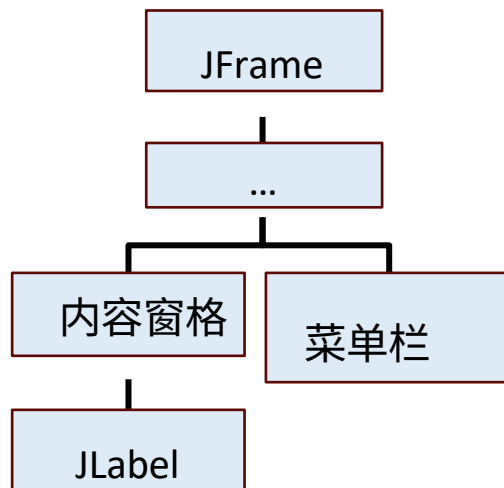
- Swing 提供了两个常用的顶级容器类：JFrame 和 JDialog
 - 每个使用 Swing 组件的程序都至少有一个顶层容器，它是 *容器层次结构* 的根节点
 - 要在屏幕上显示，每个图形用户界面组件都必须是一个包含层次
 - 每个图形用户界面组件只能被包含一次；如果一个组件已经在另一个容器中，而你试图将它添加到另一个容器中，该组件将从第一

个容器中移除，然后再添加到第二个容器中。

顶层容器类的内容窗格

- 每个顶层容器都有一个*内容窗格*，其中包含（直接或间接）该顶层容器图形用户界面中的可见组件
- 您可以选择在顶层容器中添加*菜单栏*
 - 按照惯例，菜单栏位于顶层容器内，但在内容窗格之外
 - 有些界面（如 Mac OS）允许你选择将菜单栏放在其他更适合该界面的位置，如屏幕顶部。

JFrame 顶级容器



容器

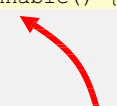
层次结构

屏幕上的窗口容器结构

顶级容器示例 (1)

```
1  import java.awt.*;
2  import
3  java.awt.event.*;
4  import javax.swing.*;
5
6
7  公共类 TopLevelDemo {
8
9      私人静态 void createAndShowGUI() {
10         JFrame frame = new JFrame("TopLevelDemo");
11         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12         JMenuBar greenMenuBar = new JMenuBar();
13         greenMenuBar.setOpaque(true);
14
15         greenMenuBar.setBackground(new Color(154, 165, 127));
16         greenMenuBar.setPreferredSize(new Dimension(200, 20));
17         JLabel yellowLabel = new JLabel();
18         yellowLabel.setOpaque(true);
19
20         yellowLabel.setBackground(new Color(248, 213, 131));
21         yellowLabel.setPreferredSize(new Dimension(200, 180));
22         frame.setJMenuBar(greenMenuBar);
23         frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
24     }
25 }
26
27 }
```

```
22     public static void main(String[] args) {
23         javax.swing.SwingUtilities.invokeLater(new Runnable() {
24             public void run() { createAndShowGUI(); }
25         });
26     }
27 }
```



这部分是标准内容，在
下面的幻灯片中将不再
展示

```
frame.pack();  
frame.setVisible(true);  
}
```

顶层容器示例 (2)

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  公共类 TopLevelDemo {
6      私人静态 void createAndShowGUI() {
7          JFrame frame = new JFrame("TopLevelDemo");
8          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9          JMenuBar greenMenuBar = new JMenuBar();
10         greenMenuBar.setOpaque(true);
11         greenMenuBar.setBackground(new Color(154, 165, 127));
12         greenMenuBar.setPreferredSize(new Dimension(200, 20));
13         JLabel yellowLabel = new JLabel();
14         yellowLabel.setOpaque(true);
```

必要的进口

创建并设置
窗口 (JFrame)

```
15     yellowLabel.setBackground(new Color(248, 213, 131));
16     yellowLabel.setPreferredSize(new Dimension(200, 180));
17     frame.setJMenuBar(greenMenuBar);
18     frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
19     frame.pack();
20     frame.setVisible(true);
21 }
```

顶级容器示例 (3)

```
1  import java.awt.*;
2  import
3  java.awt.event.*;
4  import javax.swing.*;
5
6
7  公共类 TopLevelDemo {
8
9      私人静态 void createAndShowGUI() {
10          JFrame frame = new JFrame("TopLevelDemo");
11          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12          JMenuBar greenMenuBar = new JMenuBar();
13          greenMenuBar.setOpaque(true);
14          greenMenuBar.setBackground(new Color(154, 165, 127));
15          greenMenuBar.setPreferredSize(new Dimension(200, 20));
16          JLabel yellowLabel = new JLabel();
17          yellowLabel.setOpaque(true);
18          yellowLabel.setBackground(new Color(248, 213, 131));
19          yellowLabel.setPreferredSize(new Dimension(200, 180));
20          frame.setJMenuBar(greenMenuBar);
21          frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
```

创建大小为 (200,20)
的绿色背景菜单栏

```
frame.pack();  
frame.setVisible(true);  
}
```

顶层容器示例 (4)

```
1  import java.awt.*;
2  import
3  java.awt.event.*;
4  import javax.swing.*;
5
6
7  公共类 TopLevelDemo {
8
9      私人静态 void createAndShowGUI() {
10         JFrame frame = new JFrame("TopLevelDemo");
11         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12         JMenuBar greenMenuBar = new JMenuBar();
13         greenMenuBar.setOpaque(true);
14         greenMenuBar.setBackground(new Color(154, 165, 127));
15         greenMenuBar.setPreferredSize(new Dimension(200, 20));
16         JLabel yellowLabel = new JLabel();
17         yellowLabel.setOpaque(true);
18         yellowLabel.setBackground(new Color(248, 213, 131));
19         yellowLabel.setPreferredSize(new Dimension(200, 180));
20         frame.setJMenuBar(greenMenuBar);
21         frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
```

创建标签
黄色背景

```
frame.pack();  
frame.setVisible(true);  
}
```


顶层容器示例 (5)

```
1  import java.awt.*;
2  import
3  java.awt.event.*;
4  import javax.swing.*;
5
6
7  公共类 TopLevelDemo {
8
9      私人静态 void createAndShowGUI() {
10         JFrame frame = new JFrame("TopLevelDemo");
11         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12         JMenuBar greenMenuBar = new JMenuBar();
13         greenMenuBar.setOpaque(true);
14
15         greenMenuBar.setBackground(new Color(154, 165, 127));
16         greenMenuBar.setPreferredSize(new Dimension(200, 20));
17         JLabel yellowLabel = new JLabel();
18         yellowLabel.setOpaque(true);
19
20         yellowLabel.setBackground(new Color(248, 213, 131));
21         yellowLabel.setPreferredSize(new Dimension(200, 180));
22         frame.setJMenuBar(greenMenuBar);
23         frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
```

添加菜单栏和
标签到框架

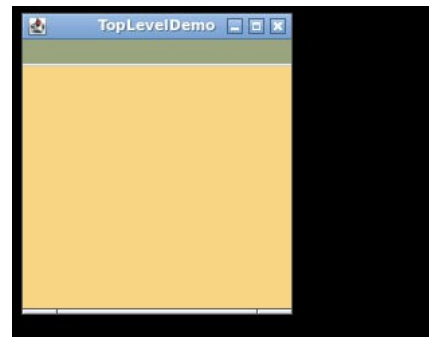
```
frame.pack();  
frame.setVisible(true);  
}
```

顶层容器示例 (6)

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 公共类 TopLevelDemo {
6     私人静态 void createAndShowGUI() {
7         JFrame frame = new JFrame("TopLevelDemo");
8         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9         JMenuBar greenMenuBar = new JMenuBar();
10        greenMenuBar.setOpaque(true);
11        greenMenuBar.setBackground(new Color(154, 165, 127));
12        greenMenuBar.setPreferredSize(new Dimension(200, 20));
13        JLabel yellowLabel = new JLabel();
14        yellowLabel.setOpaque(true);
15        yellowLabel.setBackground(new Color(248, 213, 131));
16        yellowLabel.setPreferredSize(new Dimension(200, 180));
17        frame.setJMenuBar(greenMenuBar);
18        frame.getContentPane().add(yellowLabel, BorderLayout.CENTER);
19        frame.pack();
20        frame.setVisible(true);
21    }
```

```
java TopLevelDemo
```

控制台

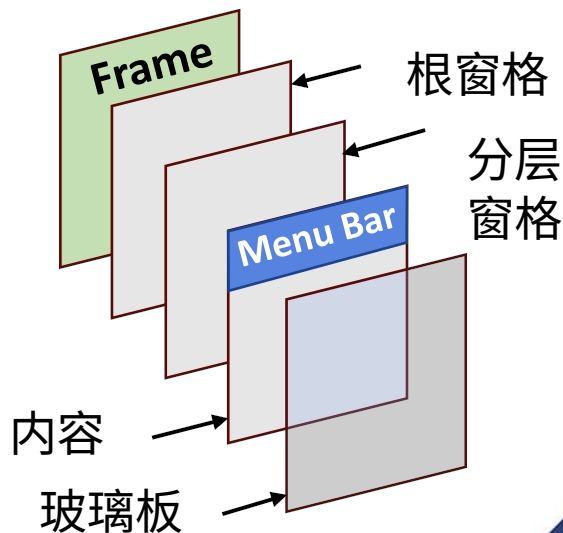


虚拟桌面

根窗格

- 每个顶层容器都依赖于一个隐蔽的称为 *根窗格* 的中间容器
- 根窗格管理内容窗格和菜单栏，以及其他几个容器，如内容和玻璃窗格
- 分层窗格包含菜单栏和内容窗格
- 玻璃窗格通常用于拦截顶层容器上的输入事件，也可用于涂抹其他组件

根窗格为框架提供的组件列表



玻璃板

对话框

- *对话框*是一个独立的子窗口（或*弹出窗口*）
除主应用程序窗口外，还可携带通知
 - 大多数对话框向用户显示错误信息或警告，但对话框也可以显示图像、目录树等。
 - 如果一个对话框阻止了用户对该对话框中所有其他窗口的输入，那么该对话框就是*模态对话框*。
直至关闭
- 在 Swing 中，类 **JDialog** 用于实例化对话框的顶级容器
 - **JOptionPane** 类提供了简单的标准模式对话框，但

要创建 *非模式* 对话框，必须直接使用 JDialog 类

通过 JOptionPane 实现的简单示例对话框

```
1  import javax.swing.*;  
2  公共类 OptionPaneExample {  
3      JFrame f;  
4      OptionPaneExample() {  
5          f=new JFrame();  
6          JOptionPane.showMessageDialog(f, "Hello, Welcome to JC2002.");  
7      }  
8      public static void main(String  
9          新的 OptionPaneExample();  
10     }  
11 }
```



通过 JDialog 的非模式示例对话 (1)

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  公共类 DialogExample {
5      私有静态 JDialog d;
6      DialogExample() {
7          JFrame f= new JFrame();
8          d = new JDialog(f, "Dialog Example", true);
9          d.setLayout(new FlowLayout());
10         JButton b = new JButton ("OK");
11         b.addActionListener (new ActionListener() {
12             public void actionPerformed(ActionEvent e) {
13                 DialogExample.d.setVisible(false);
14             }
15         });
16     }
17 }
```

```
16      d.add(new JLabel ("Click button to continue.));
17      d.add(b);
18      d.setSize(300,300);
19      d.setVisible(true)
20  } ;

21      public static void main(String args[]) {
22          新 DialogExample();
23      }
24  }
```

通过 JDialog 实现非模式示例对话 (2)

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  公共类 DialogExample {
5      私有静态 JDialog d;
6      DialogExample() {
7          JFrame f= new JFrame();
8          d = new JDialog(f, "Dialog Example", true);
9          d.setLayout(new FlowLayout());
10         JButton b = new JButton ("OK");
11         b.addActionListener (new ActionListener() {
12             public void actionPerformed(ActionEvent e) {
13                 DialogExample.d.setVisible(false);
14             }
15         });
```

创建包含对话框的
JFrame 对象

创建 JDialog 对象

创建 JButton 对象

```
16     d.add(new JLabel ("Click button to continue.));
17     d.add(b);
18     d.setSize(300,300);
19     d.setVisible(true)
20 } ;
```

```
21
22     public static void main(String args[]) {
23         新 DialogExample();
24     }
```

通过 JDialog 实现非模式示例对话 (3)

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  公共类 DialogExample {
5      私有静态 JDialog d;
6      DialogExample() {
7          JFrame f= new JFrame();
8          d = new JDialog(f, "Dialog Example", true);
9          d.setLayout(new FlowLayout());
10         JButton b = new JButton ("OK");
11         b.addActionListener (new ActionListener() {
12             public void actionPerformed(ActionEvent e) {
13                 DialogExample.d.setVisible(false);
14             }
15         });
16     }
17 }
```

我们稍后将讨论布局
和动作监听器。

```
16     d.add(new JLabel ("Click button to continue.));
17     d.add(b);
18     d.setSize(300,300);
19     d.setVisible(true)
20 } ;
```

```
21
22     public static void main(String args[]) {
23         新 DialogExample();
24     }
```

通过 JDialog 实现非模式示例对话 (4)

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  公共类 DialogExample {
5      私有静态 JDialog d;
6      DialogExample() {
7          JFrame f= new JFrame();
8          d = new JDialog(f, "Dialog Example", true);
9          d.setLayout(new FlowLayout());
10         JButton b = new JButton ("OK");
11         b.addActionListener (new ActionListener() {
12             public void actionPerformed(ActionEvent e) {
13                 DialogExample.d.setVisible(false);
14             }
15         });
16     }
17 }
```

添加标签、按钮并使
对话框可见


```

16     d.add(new JLabel ("Click button to continue.));
17     d.add(b);
18     d.setSize(300,300);
19     d.setVisible(true)
20 }
    ;

21
22     public static void main(String args[]) {
23         新 DialogExample();
24     }
}

```

通过 Jdialog 实现非模态示例对话框 (5)

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  公共类 DialogExample {
5      私有静态 JDialog d;
6      DialogExample() {
7          JFrame f= new JFrame();
8          d = new JDialog(f, "Dialog Example", true);
9          d.setLayout(new FlowLayout());
10         JButton b = new JButton ("OK");
11         b.addActionListener (new ActionListener() {
12             public void actionPerformed(ActionEvent e) {
13                 DialogExample.d.setVisible(false);
14             }
15         });
```

```
16         d.add(new JLabel ("Click  
button to continue.));
```

```
17         d.add(b);
```



\$ java DialogExample

```
18     d.setSize(300,300);
19     d.setVisible(true)
20 } ;
```

■

```
21     public static void main(String args[]) {
22         新 DialogExample();
23     }
24 }
```

有问题或意见？

JC2002 Java 程序设计

第 7 天，课程 3：在 Swing 中使用布局和按钮

添加 JC 组件

- 不同的图形用户界面组件（按钮、图像、文本字段等）由 JComponent 的子类表示
- 使用 add() 方法将图形用户界面组件添加到容器中，并将添加的 JComponent

```
1  import javax.swing.*;  
2  公共类 ButtonExample {  
3      public static void main(String[] args) {  
4          JFrame f=new JFrame("Button Example");  
5          JButton b=new JButton("Click Here");  
6          b.setBounds(50,35,200,30);  
7          f.add(b);  
8          f.setSize(300,100);  
9          f.setLayout(null);  
          f.setVisible(true);
```

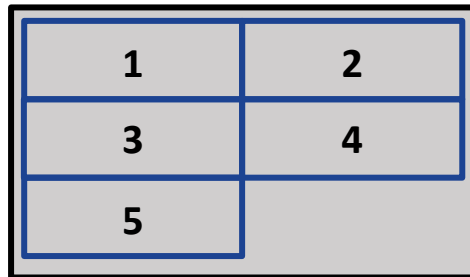
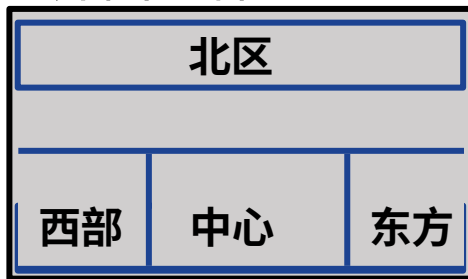
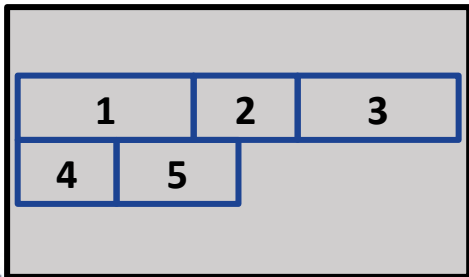
按钮位置和大
小

添加按钮

子类对象作为参数

布局管理器

- 为组件设置绝对位置和大小可能会显得不美观
如果不知道目标平台的屏幕分辨率
- 一些 Swing 和 AWT 类提供了布局管理器，用于根据不同的特定规则动态分配图形用户界面组件



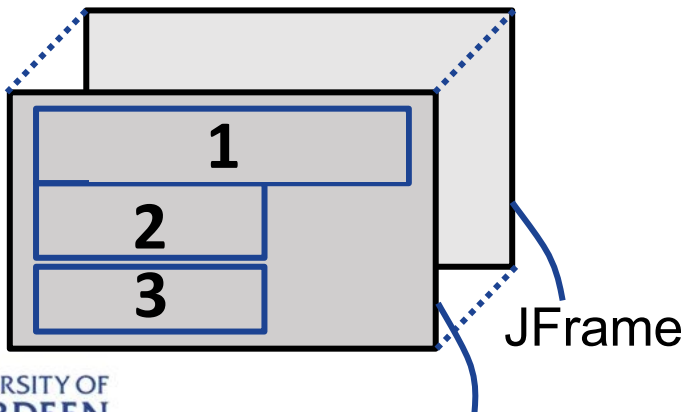
流式布局（默认）

边框布局

网格布局

JPanel 上的布局

- 某些布局（如 FlowLayout）可直接在 JFrame 上使用
- 某些布局，如 GridLayout 和 BoxLayout，需要创建 JFrame 和组件之间的 JPanel 对象

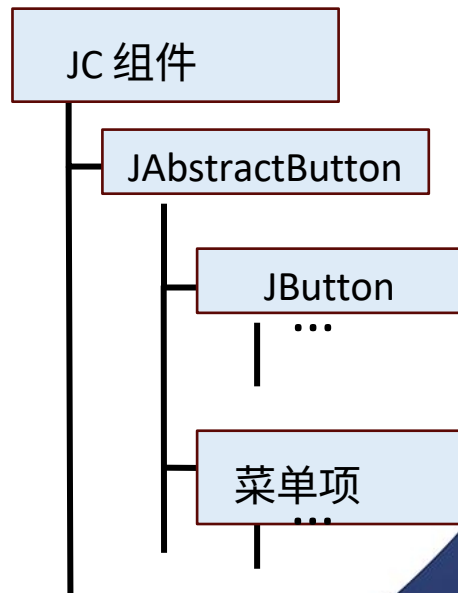


```
JFrame frame = new JFrame("Win");  
JPanel panel = new JPanel();  
BoxLayout boxlayout = new  
BoxLayout(panel, BoxLayout.Y_AXIS);  
panel.setLayout(boxlayout);  
...  
frame.add(panel);
```

方框布局 JPanel

按钮：类 JButton

- 按钮是使用最广泛的图形用户界面之一。
部件
 - 多个子类型：单选按钮、菜单项、复选框等。
- Swing 类 JButton 可同时显示文字和图像
 - 按钮文本中的下划线字母表示按钮的*助记符*（键盘上的替代选择
 - 通常，用户可以按 **Alt** 键点击按钮和记忆法
 - 可定义*工具提示*来解释按钮的含义



初始化 JButton 对象 (1)

- 带图标和文字的按钮示例（两者均可选）

```
1 ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
2 b1 = new JButton("Unhappy", unhappyButtonIcon);
3 b1.setSize(100,100);
4 b1.setVerticalTextPosition(AbstractButton.BOTTOM);
5 b1.setHorizontalTextPosition(AbstractButton.CENTER);
6 b1.setMnemonic(KeyEvent.VK_U);
7 b1.setToolTipText("如果您不满意，请单击此处");
```

初始化 JButton 对象 (2)

- 定义图标（图像文件）和文本

```
1 ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png")
2 .
3 b1.setSize(100,100);
4 b1 = new JButton("Unhappy", unhappyButtonIcon);
5 b1.setVerticalTextPosition(AbstractButton.BOTTOM);
6 b1.setHorizontalTextPosition(AbstractButton.CENTER);
7 b1.setMnemonic(KeyEvent.VK_U);
8 b1.setToolTipText("Click this if you are unhappy.");
```

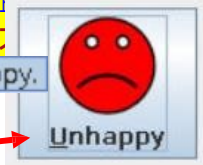


初始化 JButton 对象 (3)

- 定义助记符和工具提示文本

```
1 ImageIcon unhappyButtonIcon = new
2 ImageIcon("unhappy.png"); b1 = new JButton("Unhappy",
3 unhappyButtonIcon); b1.setSize(100,100);
4 b1.setVerticalTextPosition(AbstractButton.BOTTOM);
5 b1.setHorizontalTextPosition(AbstractButton.CENTER);
6 b1.setMnemonic(KeyEvent.VK_U);
7 b1.setToolTipText("如果您不高兴，请点击此处");
```

Click this if you are unhappy.

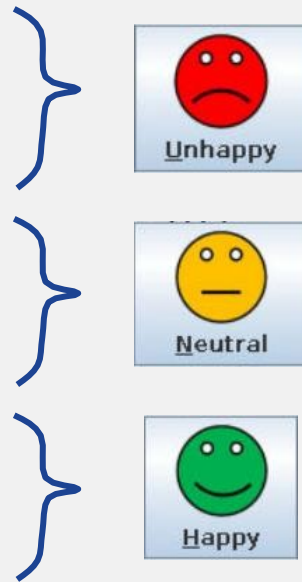


按钮示例：初始化

```
1  import javax.swing.*;
2  import javax.swing.border.*;
3  import java.awt.BorderLayout.BorderLayout.BorderLayout.BorderLayout
4  import java.awt.event.*;
5  公共类 ButtonExample1 {
6      public static void main(String[] args) {
7          JButton b1、 b2、 b3;
8          JLabel questionLabel, responseLabel;
9          JFrame frame = new JFrame();
10         questionLabel = new JLabel("告诉我您对 Java 有多满意! \n"、
11             SwingConstants.CENTER) ;
12         responseLabel = new JLabel("No answer given"、
13             SwingConstants.CENTER) ;
14
15         // 创建按钮图标
16         ImageIcon unhappyButtonIcon = new ImageIcon("unhappy.png");
17         ImageIcon neutralButtonIcon = new ImageIcon("neutral.png");
18         ImageIcon happyButtonIcon = new ImageIcon("happy.png");
```


按钮示例：定义按钮

```
19      // 定义不开心按钮
20      b1 = new JButton("Unhappy", unhappyButtonIcon);
21      b1.setVerticalTextPosition(AbstractButton.BOTTOM);
22      b1.setHorizontalTextPosition(AbstractButton.CENTER);
23      b1.setMnemonic(KeyEvent.VK_U);
24      b1.setToolTipText("如果您不满意，请单击此处");
25      // 定义中性按钮
26      b2 = 新 JButton("Neutral", neutralButtonIcon);
27      b2.setVerticalTextPosition(AbstractButton.BOTTOM);
28      b2.setHorizontalTextPosition(AbstractButton.CENTER);
29      b2.setMnemonic(KeyEvent.VK_N);
30      b2.setToolTipText("如果您觉得中立，请单击此处");
```

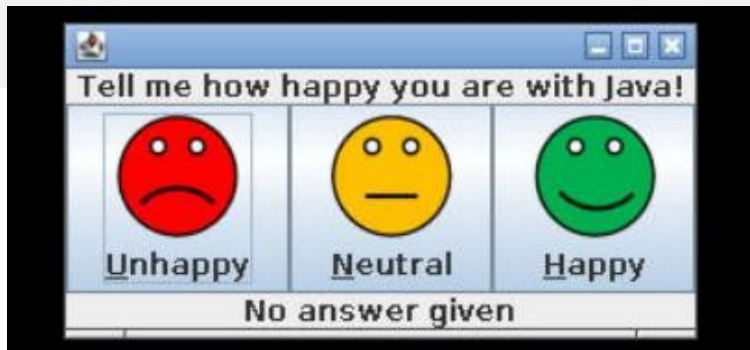


```
31      // 定义快乐按钮
32      b3 = new JButton("Happy", happyButtonIcon);
33      b3.setVerticalTextPosition(AbstractButton.BOTTOM);
34      b3.setHorizontalTextPosition(AbstractButton.CENTER);
35      b3.setMnemonic(KeyEvent.VK_H);
36      b3.setToolTipText("Click this if you are happy.");
```

按钮示例：布局

```
37
38
39 // 向框架添加组件
40 frame.add(questionLabel, BorderLayout.PAGE_START);
41 frame.add(b1, BorderLayout.LINE_START);
42 frame.add(b2, BorderLayout.CENTER);
43 frame.add(b3, BorderLayout.LINE_END);
44 frame.add(responseLabel, BorderLayout.PAGE_END);
45 frame.pack();
46 frame.setVisible(true);
47 }
48 }
```

使用
边框布局



复选框：类 JCheckBox

- JCheckBox 类提供支持复选框按钮
- 对于菜单中的复选框，使用 JCheckBoxMenuItem 类
- JCheckbox 继承了 JAbstractButton；因此，它具有按钮的常规特性和可用方法



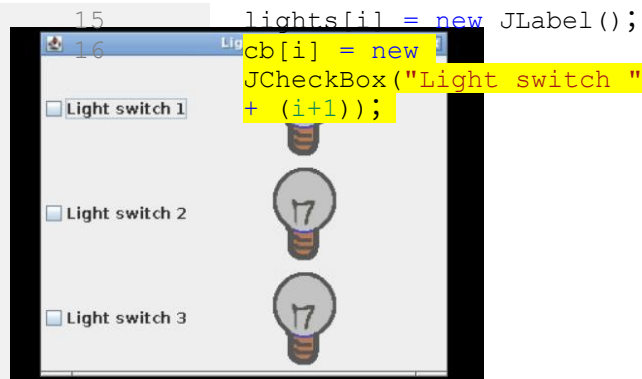
复选框示例：使用 GridLayout

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  公共类 CheckBoxExample1 {
5      public static void main(String[] args) {
6          JFrame frame = new JFrame("Lights");
7          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8          ImageIcon lightOffIcon = new ImageIcon("light_off.png");
9          ImageIcon lightOnIcon = new ImageIcon("light_on.png");
10
11         JLabel lights[] = new JLabel[3];
12         JCheckBox cb[] = new JCheckBox[3];
13         JPanel panel = new JPanel();
14         GridLayout gridlayout = new GridLayout(3,2);
15         panel.setLayout(gridlayout);
16         for(int i=0; i<3; i++) {
17             lights[i] = new JLabel();
18             cb[i] = new JCheckBox("Light switch " + (i+1));
19             lights[i].setIcon(lightOffIcon);
20             panel.add(cb[i]);
21             panel.add(lights[i]);
22         }
23         frame.add(panel);
24         frame.setSize(500,500);
25         frame.setVisible(true);
26     }
```

请注意，标签可以包含图像（图标）而不是文本！

复选框示例：定义复选框

```
1  import javax.swing.*;
2  import java.awt.*;
3  公共类 CheckBoxExample1 {
4      public static void main(String[] args) {
5          JFrame frame = new JFrame("Lights");
6          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7          ImageIcon lightOffIcon = new ImageIcon("light_off.png");
8          ImageIcon lightOnIcon = new ImageIcon("light_on.png");
9          JLabel lights[] = new JLabel[3];
10         JCheckBox cb[] = new JCheckBox[3];
11         JPanel panel = new JPanel();
12         GridLayout gridlayout = new GridLayout(3,2);
13         panel.setLayout(gridlayout);
14         for(int i=0; i<3; i++) {
```



```
17      lights[i].setIcon(lightOffIcon);
```

```
        panel.add(cb[i]);  
        panel.add(lights[i]);  
    }  
21    frame.add(panel);  
22    frame.setSize(500,500);  
23    frame.setVisible(true)  
24    }  
25    ;
```


单选按钮：类 JRadioButton

- JRadioButton 类提供了支持复选框按钮
- 对于菜单中的复选框，使用 JRadioButtonMenuItem 类
- JRadioButton 也继承自 JAbstractButton，因此它具有按钮的常规特性和可用方法
- 使用 ButtonGroup 可确保只有每次检查一个按钮



单选按钮示例：使用 BoxLayout

```
1  import javax.swing.*;
2  公共类 RadioButtonExample1 {
3      public static void main(String[] args) {
4          JFrame frame = new JFrame("Quiz");
5          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6          JPanel panel = new JPanel();
7          BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
8          panel.setLayout(boxlayout);
9          JLabel question = new JLabel("What is the capital of China?");
10         JButton submit = new JButton("Submit your answer");
11         ButtonGroup group = new ButtonGroup();
12         JRadioButton rb[] = new JRadioButton[4]
13             ;
14         rb[0] = new JRadioButton("Shanghai");
15         rb[1] = new JRadioButton("Beijing");
16         rb[2] = new JRadioButton("Guangzhou");
17         rb[3] = new JRadioButton("Hong Kong");
```

```
JRadioButton("Chongqing");
```

```
17     submit.setEnabled(false);
18     panel.add(question);
19     for(int i=0; i<4; i++) {
20         group.add(rb[i]);
21         panel.add(rb[i]);
22     }
23     panel.add(submit);
24     frame.add(panel);
25     frame.pack();
26     frame.setVisible(true);
27 } }
```

单选按钮示例

```
1 import javax.swing.*;
2
3 公共类 RadioButtonExample1 {
4     public static void main(String[] args) {
5
6         JFrame frame = new JFrame("Quiz");
7         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8
9         JPanel panel = new JPanel();
10        BoxLayout boxlayout = new BoxLayout(panel, BoxLayout.Y_AXIS);
11        panel.setLayout(boxlayout);
12        JLabel question = new JLabel("What is the capital of China?");
13        JButton submit = new JButton("Submit your answer");
14        ButtonGroup group = new ButtonGroup();
15        JRadioButton rb[] = new JRadioButton[4];
16        rb[0] = new JRadioButton("Shanghai");
17        rb[1] = new JRadioButton("Beijing");
18        rb[2] = new JRadioButton("Guangzhou");
19        rb[3] = new JRadioButton("Chongqing");
```

```
20        submit.setEnabled(false);
21        panel.add(question);
22        for(int i=0; i<4; i++) {
23            group.add(rb[i]);
24            panel.add(rb[i]);
25        }
26        panel.add(submit);
27        frame.add(panel);
```



按钮组中只有一个单选按钮
可同时按下

}

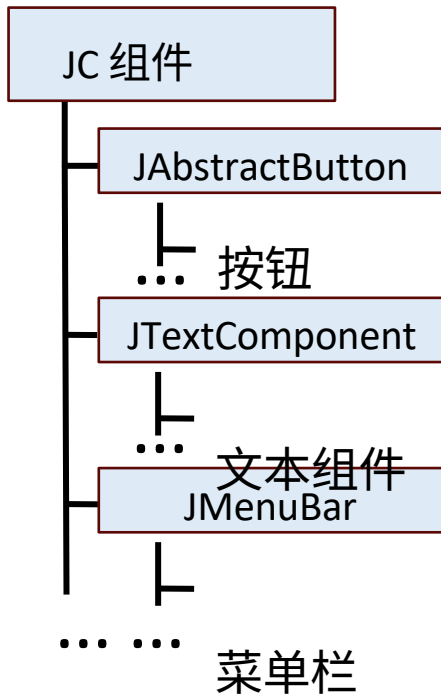
有问题或意见？

JC2002 Java 程序设计

第 7 天，第 4 课：更多 JC 组件

选择 JC 组件

- 从 JComponent 类继承的 GUI 组件种类繁多，用途各异
 - 在本课中，我们将只介绍最重要的课程
 - 不同的组件有不同的方法来定制外观、调整绝对大小和位置、设置和获取内部状态等。



使用 JLabel 显示图像

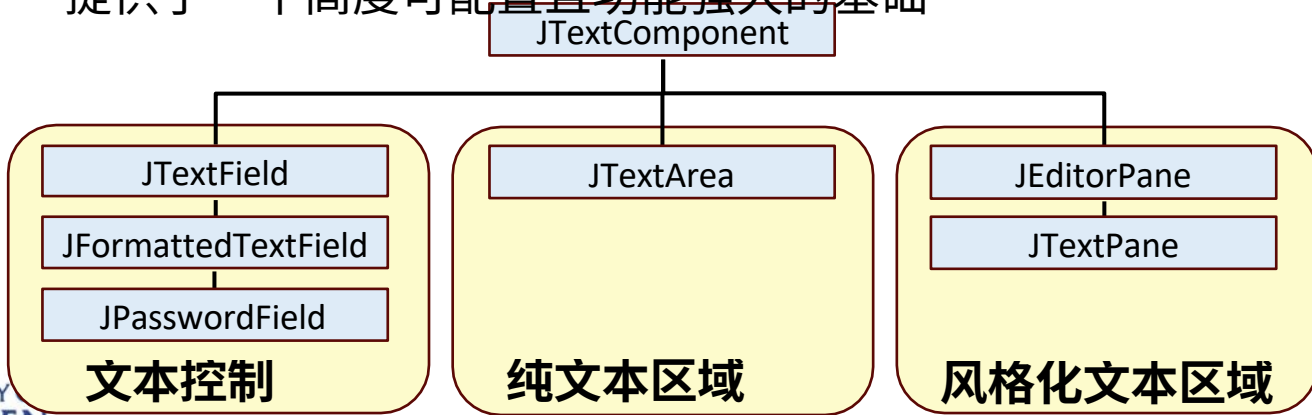
- JLabel 直接从 JComponent 继承而来，它不会对任何用户输入
 - 我们已经使用 JLabel 显示了文本，但使用其 setIcon() 方法还可以显示图像

```
1  import javax.swing.*;
2
3  类 LabelExample {
4      public static void main(String args[]){
5          JFrame f = new JFrame("Label Example");
6          ImageIcon icon = new ImageIcon("image.jpg");
7          JLabel label = new JLabel();
8          label.setIcon(icon);
9          f.add(label);
10         f.setSize(200,150);
11         f.setVisible(true);
12     }
13 }
```



文本组件

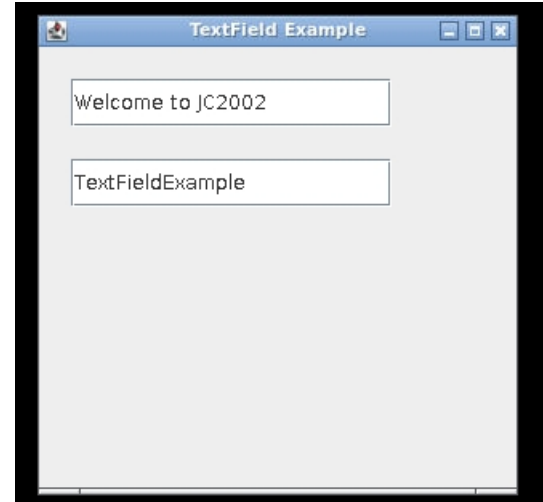
- Swing 提供了六个文本组件以及辅助类和界面，用于显示和编辑文本
 - 所有 Swing 文本组件都继承自 JTextComponent，它为文本操作提供了一个高度可配置且功能强大的基础



JTextField 示例

- 使用 JTextField 类创建一个组件，允许编辑一个单行

```
1 import javax.swing.*;
2
3 类 TextFieldExample{
4     public static void main(String args[]){
5         JFrame f= new JFrame("TextField Example");
6         JTextField t1,t2;
7
8         t1 = 新 JTextField("Welcome to JC2002");
9
10        t1.setBounds(20,20, 200,30);
11        t2=new
12        JTextField("TextFieldExample");
13        t2.setBounds(20,70, 200,30);
14        f.add(t1); f.add(t2);
15        f.setSize(300,300);
16        f . setLayout(null);
17        f . setVisible(true);
18    }
19 }
```



JTextArea 示例

- 使用 JTextArea 类创建一个可编辑多行文本

```
1  import javax.swing.*;
2
3  类 TextAreaExample{
4      public static void main(String args[]){
5          JFrame f= new JFrame("TextField Example");
6          JTextArea t1;
7          t1 = new JTextField("Welcome to \nJC2002!");
8          t1.setBounds(20,20, 200,100);
9          f.add(t1);
10         f.setSize(300,300);
11         f.setLayout(null);
12         f.setVisible(true);
13     }
```



JTextPane 示例 (1)

- 使用 JTextPane 类编辑或显示样式文本（甚至 HTML）

```
...
11 公共类 TextPaneExample {
12      public static void main(String args[]) throws BadLocationException {
13          JFrame frame = new JFrame("TextPane Example");
14          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15          Container cp = frame.getContentPane();
16          JTextPane pane = new JTextPane();
17          SimpleAttributeSet attributeSet = new SimpleAttributeSet();
18          StyleConstants.setBold(attributeSet, true);
19          pane.setCharacterAttributes(attributeSet, true);
20          pane.setText("Here is ");
21          attributeSet = new SimpleAttributeSet();
22          StyleConstants.setItalic(attributeSet, true);
23          StyleConstants.setForeground(attributeSet, Color.red);
24          Document doc = pane.getStyledDocument();
25          doc.insertString(doc.getLength(), "styled text!", attributeSet);
26          JScrollPane scrollPane = new JScrollPane(pane);
27          cp.add(scrollPane, BorderLayout.CENTER);
28      }
29  }
30  }
31  }
```

此处未显示进口

可为文本定义不同的样式
(颜色、斜体、粗体等)

```
        frame.setSize(400, 300);  
        frame.setVisible(true);  
    }  
}
```


JTextPane 示例 (2)

```
...
11  ...
12  公共类 TextPaneExample {
13      public static void main(String args[]) throws BadLocationException {
14          JFrame frame = new JFrame("TextPane Example");
15          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16          Container cp = frame.getContentPane();
17          JTextPane pane = new JTextPane();
18          SimpleAttributeSet attributeSet = new SimpleAttributeSet();
19          StyleConstants.setBold(attributeSet, true);
20          pane.setCharacterAttributes(attributeSet, true);
21
22          pane.setText("Here is ");
23          attributeSet = new SimpleAttributeSet();
24          StyleConstants.setItalic(attributeSet, true);
25          StyleConstants.setForeground(attributeSet, Color.red);
26
27          Document doc = pane.getStyledDocument();
28          doc.insertString(doc.getLength(), "styled text!", attributeSet);
29          JScrollPane scrollPane = new JScrollPane(pane);
30          cp.add(scrollPane, BorderLayout.CENTER);
31          frame.setSize(200, 100);
          frame.setVisible(true);
      }
  }
```

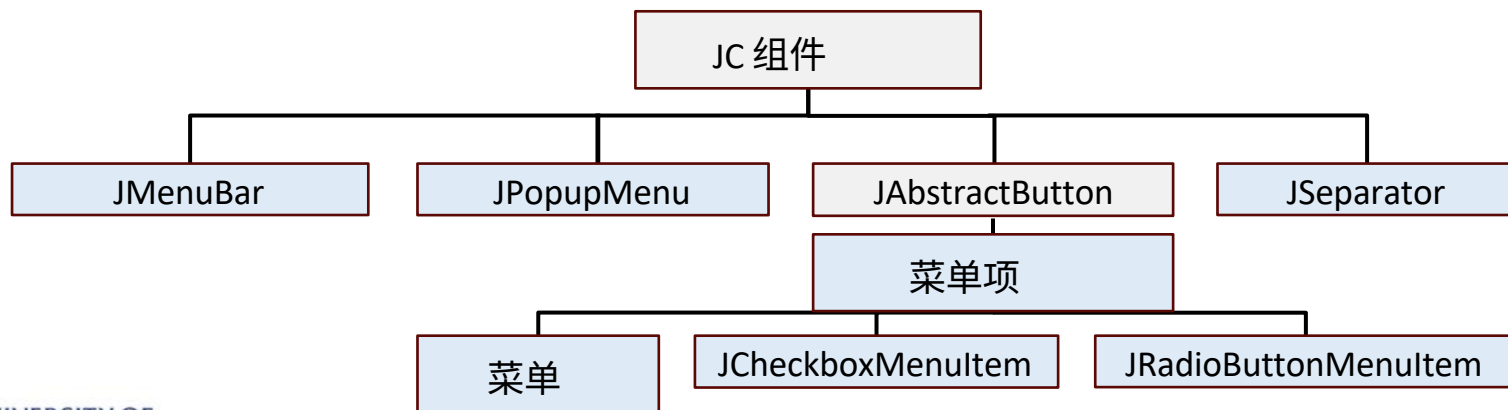


- 更多信息，请参见：

<https://docs.oracle.com/javase/tutorial/uiswing/components/editorpane.html>

菜单组件

- 有几个菜单组件为以下方面提供了许多选项
在狭小空间内执行菜单
- 菜单项 (JMenuItem) 是显示标签的各种特殊按钮



使用菜单组件示例 (1)

- 创建带子菜单的菜单栏示例

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("菜单和菜单项示例");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu"); 18      15      menu.add(i1); 25
9          submenu=new JMenu("Sub Menu"); 19      16      menu.add(i2); 26
10         i1=new JMenuItem("Item 1"); 20
11         i2=new JMenuItem("Item 2"); 21
12         i3=new JMenuItem("Item 3"); 22
13         i4=new JMenuItem("Item 4"); 23
14         i5=new JMenuItem("Item 5"); 24
```

```
submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);

17 menu.add(i3)
;
```

```
mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }
```

使用菜单组件示例 (2)

- 实例化菜单栏、菜单和子菜单

```
1  import javax.swing.*;
2  class MenuExample {
3      JMenu menu, submenu;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = 新 JFrame ("菜单和      菜单 项目示例") ;
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu ("Menu");
9          submenu=new JMenu ("Sub Menu");
10         i1=new JMenuItem("Item 1");
11         i2=new JMenuItem("Item 2");
12         i3=new JMenuItem("Item 3");
13         i4=new JMenuItem("Item 4");
14         i5=new JMenuItem("Item 5");
15         menu.add(i1);
16         menu.add(i2);
17         menu.add(i3);
18         submenu.add(i4);
19         submenu.add(i5);
20         menu.add(submenu);
21         mb.add(menu);
22         f.setJMenuBar(mb);
23         f.setSize(400,400);
24         f.setLayout(null);
25         f.setVisible(true);
26     }
27     public static void main(String args[]){
```

```
28         new MenuExample();  
29     }  
30 }
```

使用菜单组件示例 (3)

- 创建五个 JMenuItem 类菜单项对象

```
1  import javax.swing.*;
2  class MenuExample {
3      JMenu menu, submenu;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("菜单和 菜单 项目示例");
7
8          JMenuBar mb=new JMenuBar();
9          menu=new JMenu("Menu");
10         submenu=new JMenu("Sub Menu");
11         i1=new JMenuItem("Item 1");
12         i2=new JMenuItem("Item 2");
13         i3=new JMenuItem("Item 3");
14         i4=new JMenuItem("Item 4");
15         i5=new JMenuItem("Item 5");
16         menu.add(i1);
17         menu.add(i2);
18
19         submenu.add(i4);
20         submenu.add(i5);
21         menu.add(submenu);
22         mb.add(menu);
23         f.setJMenuBar(mb);
24         f.setSize(400,400);
25         f.setLayout(null);
26         f.setVisible(true);
27     }
```



```
17 menu.add(i3);
```

```
27 public static void main(String args[]){  
28     new MenuExample();  
29 }  
30 }
```

使用菜单组件示例 (4)

- 在主菜单中添加前三个菜单项

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("菜单和菜单项示例");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu");      18      15      menu.add(i1);      25
9          submenu=new JMenu("Sub Menu"); 19      16      menu.add(i2);      26
10         i1=new JMenuItem("Item 1");    20
11         i2=new JMenuItem("Item 2");    21
12         i3=new JMenuItem("Item 3");    22
13         i4=new JMenuItem("Item 4");    23
14         i5=new JMenuItem("Item 5");    24
```

```
submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);
```

```
17 menu.add(i3)
;
```

```
mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }
```

使用菜单组件示例 (5)

- 在子菜单中添加最后两个菜单项

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("菜单和菜单项示例");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu");          18      15      menu.add(i1);          25
9          submenu=new JMenu("Sub Menu");    19      16      menu.add(i2);          26
10         i1=new JMenuItem("Item 1");        20
11         i2=new JMenuItem("Item 2");        21
12         i3=new JMenuItem("Item 3");        22
13         i4=new JMenuItem("Item 4");        23
14         i5=new JMenuItem("Item 5");        24
```

```

submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);
17     menu.add(i3)
        ;

```

```

mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }

```

使用菜单组件示例 (6)

- 在主菜单上添加子菜单，在菜单栏上添加主菜单

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("菜单和菜单项示例");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu");      18      15      menu.add(i1);      25
9          submenu=new JMenu("Sub Menu"); 19      16      menu.add(i2);      26
10         i1=new JMenuItem("Item 1");    20
11         i2=new JMenuItem("Item 2");    21
12         i3=new JMenuItem("Item 3");    22
13         i4=new JMenuItem("Item 4");    23
14         i5=new JMenuItem("Item 5");    24
```

```
submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);

17     menu.add(i3)
      ;
```

```
mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }
```

使用菜单组件示例 (7)

- 使用 setJMenuBar() 方法为框架添加菜单栏

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("Menu and MenuItem Example");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu");
9          submenu=new JMenu("Sub Menu");
10         i1=new JMenuItem("Item 1");
11         i2=new JMenuItem("Item 2");
12         i3=new JMenuItem("Item 3");
13         i4=new JMenuItem("Item 4");
14         i5=new JMenuItem("Item 5");
15         menu.add(i1);
16         menu.add(i2);
17         submenu.add(i3);
18         submenu.add(i4);
19         submenu.add(i5);
20         f.add(mb);
21         f.add(submenu);
22         f.setVisible(true);
23     }
24 }
25
26
```



```
submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);

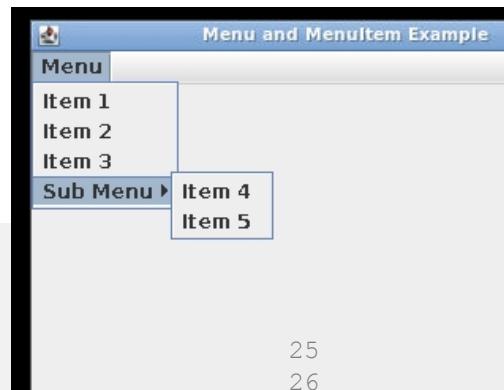
17 menu.add(i3)
;
```

```
mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }
```

使用菜单组件示例 (8)

- 运行程序

```
1  import javax.swing.*;
2  类 MenuExample {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      MenuExample() {
6          JFrame f = new JFrame("Menu and MenuItem Example");
7          JMenuBar mb=new JMenuBar();
8          menu=new JMenu("Menu");
9          submenu=new JMenu("Sub Menu");
10         i1=new JMenuItem("Item 1");
11         i2=new JMenuItem("Item 2");
12         i3=new JMenuItem("Item 3");
13         i4=new JMenuItem("Item 4");
14         i5=new JMenuItem("Item 5");
15         menu.add(i1);
16         menu.add(i2);
17         submenu.add(i3);
18         submenu.add(i4);
19         submenu.add(i5);
20         mb.add(menu);
21         mb.add(submenu);
22         f.setJMenuBar(mb);
23         f.setVisible(true);
24     }
25 }
26
```



```
submenu.add(i4)
;
submenu.add(i5)
;
menu.add(submen
u);

17 menu.add(i3)
;
```

```
mb.add(menu);
f.setJMenuBar(mb);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
27 public static void main(String args[]){
28     new MenuExample();
29 }
30 }
```

使用 JSeparator 的示例

- 使用 JSeparator 实例分隔菜单项组

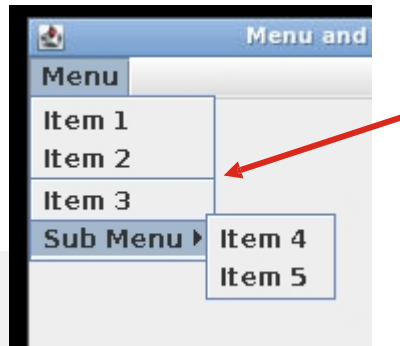
```
1  import javax.swing.*;
2  类 MenuExample2 {
3      JMenu 菜单, 子菜单;
4      JMenuItem i1, i2, i3, i4, i5;
5      JSeparator s;
6      MenuExample() {
7          JFrame f = new JFrame("Menu and MenuItem Example");
8          JMenuBar mb=new JMenuBar();
9          menu=new JMenu("Menu");
10         submenu=new JMenu("Sub Menu");
11         i1=new JMenuItem("Item 1");
12         i2=new JMenuItem("Item 2");
```

13

i
3
=
n
e
w

14
15
16

```
JMenuItem("Item 3");
i4=new JMenuItem("Item 4");
i5=new JMenuItem("Item 5");
s=new JSeparator();
```



```
17     menu.add(i1);
```

```
18     menu.add(i2)  
19     ;
```

```
19     menu.add(s);
```

```
20     menu.add(i3)  
21     ;
```

```
21     submenu.add(i4);  
22     submenu.add(i5);  
23     menu.add(submenu);  
24     mb.add(menu);  
25     f.setJMenuBar(mb);  
26     f.setSize(400,400);  
27     f.setLayout(null);  
28     f.setVisible(true)  
29 } ;
```

```
30 public static void main(String args[]){  
31     new MenuExample();  
32 } }
```

摘要

- 图形用户界面提供了一个用户友好的机制，以便与简单易学的应用程序互动
 - 图形用户界面通常基于相似的逻辑，但在不同平台上会有不同的外观和感觉
- 在 Java 中，Swing 是实现图形用户界面的常用库。
 - 在 Swing 中，不同的图形用户界面组件都是通过继承自 JComponent 类的类来实现的
- Swing 允许使用不同的布局来组织图形用户界面组件

- 介绍了一些最典型的图形用户界面组件，包括按钮、文本字段和菜单项

有问题或意见？