第十一章 高级GUI设计

主要内容

1. 事件适配器
2. KeyEvent事件及其响应
3. MouseEvent事件及其响应
4. windowEvent事件及其响应
5. JScrollPane与JScrollBar组件
6. JTabbedPane容器
7. 菜单设计
8. 对话框设计

1 事件适配器

- Java采用委托事件模型来处理事件。当事件产生时，通过注册的监听器对象的Listener接口的事件处理方法处理。

- 然而当一个Listener接口有很多处理方法，则不管是否需要，必须实现所有的方法。浪费资源，使系统开销加大。

1 事件适配器

- Java语言为Listener接口提供适配器类(Adapter)，事件适配器(EventAdapter)提供了一种简单的处理方法。

处理方法：

当事件源注册了含有多个处理方法的监听器对象时，可通过继承事件所对应的Adapter类，重写需要的方法，不需要的方法不用重写。

2 键盘事件(KeyEvent)

- 当用户使用键盘进行操作时，则会产生KeyEvent事件。处理KeyEvent事件的监听者对象可以实现KeyListener接口，或者实现继承KeyAdapter的子类。

- KeyListener中有三个事件：

1. keyPressed(KeyEvent e); 键盘按键被按下的事件
2. keyReleased(KeyEvent e); 键盘按键被放开事件
3. keyTyped(KeyEvent e); 按键被敲击事件

- KeyEvent类中常用方法：

```
public char getKeyChar();
```

返回引发键盘事件的按键对应的Unicode字符。

如果按键没有Unicode字符与之对应，返回KeyEvent类的静态常量KeyEvent.CHAR-UNDEFINED。

```
public String getKeyText()
```

返回引发键盘事件的按键的文本内容。

- 键盘事件响应流程：

1. 将响应键盘事件所需业务逻辑封装在监听器类

```
Class KeyEventDemo extends KeyAdapter
{
  public void keyTyped(KeyEvent e)
  {}
  public void keyPressed(KeyEvent e)
  {}
  public void keyReleased(KeyEvent e)
  {}
}
```

- 键盘事件响应流程：

2. 创建事件源并注册完成所需业务逻辑的监听器

```
KeyAdapter key1 = new KeyEventDemo();
JTextField text = new JTextField(20);
text.addKeyListener(key1);
```

3. 引用监听器的KeyTyped(), KeyPressed(), KeyReleased()方法响应事件。

3 鼠标事件(MouseEvent)

- 在图形用户界面中，鼠标主要用来进行选择、切换或绘画。当用户用鼠标进行交互操作时，会产生鼠标事件MouseEvent。
- 处理MouseEvent事件的监听者对象是可以实现MouseListener接口和MouseMotionListener接口的类，或者实现继承MouseAdapter的子类。

- 与Mouse有关的事件可分为两类：

  1. MouseListener接口，共有5种方法，针对鼠标的按键与位置检测。

  2. MouseMotionListener接口，共有2种方法，针对鼠标的坐标与拖动操作。

MouseEvent事件及其监听者

事件监听者	成员方法	说明
MouseListener	moveClicked(MouseEvent e)	鼠标点击事件
	moveEntered(MouseEvent e)	鼠标进入事件
	moveExited(MouseEvent e)	鼠标离开事件
	movePressed(MouseEvent e)	鼠标按下事件
	moveReleased(MouseEvent e)	鼠标释放事件
MouseMotionListener	moveDragged(MouseEvent e)	鼠标拖动事件
	moveMoved(MouseEvent e)	鼠标移动事件

- 鼠标事件响应流程：

1. 将响应鼠标事件所需业务逻辑封装在监听器类

```
Class MouseEventDemo extends MouseAdapter
{
  public void moveEntered(MouseEvent e)
  {}
  public void moveExited(MouseEvent e)
  {}
  public void movePressed(MouseEvent e)
  {}
}
```

- 鼠标事件响应流程：

2. 创建事件源并注册完成所需业务逻辑的监听器

```
MouseAdapter mouse1 = new MouseEventDemo();

JLabel lb = new JLabel( );
addMouseListener(mouse1);
```

3. 引用监听器的moveEntered(), moveExited()方法响应事件。

4 窗口事件(WindowEvent)

- Java提供窗口事件WindowEvent对窗口进行操作，总共包含7种事件。

窗口事件	说明
WINDOW_ACTIVATED	代表窗口被激活
WINDOW_DEACTIVATED	代表窗口失活
WINDOW_OPENED	代表窗口被打开
WINDOW_CLOSED	代表窗口已被关闭
WINDOW_CLOSING	代表窗口正在关闭
WINDOW_ICONIFIED	代表使窗口最小化成图标
WINDOW_DEICONIFIED	代表窗口从图标恢复

- WindowEvent类的主要方法：

  getWindow();

  返回引发当前WindowEvent事件的具体窗口，返回值是具体的Window对象。

- 窗口事件响应流程：

1. 将响应窗口事件所需业务逻辑封装在监听器类

```
Class WindowEventDemo extends WindowAdapter
{
  public void windowClosing(WindowEvent e)
  { System.exit(0); }
  public void windowIconified(WindowEvent e)
  {}
  public void windowDeiconified(WindowEvent e)
  {}
}
```

- 窗口事件响应流程：

2. 创建事件源并注册完成所需业务逻辑的监听器

```
WindowAdapter w = new WindowEventDemo();

JFrame f = new JFrame();

f.addWindowListener(w);
```

3. 引用监听器的方法响应事件。

5 JScrollPane

- 当窗口里的内容大于窗口时，可在窗口的右边和下边设置滚动条。
- JScrollPane就是具有这种功能的组件，称为滚动面板，用于滚动窗口。

5 JScrollPane

- JScrollPane类的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JScrollPane

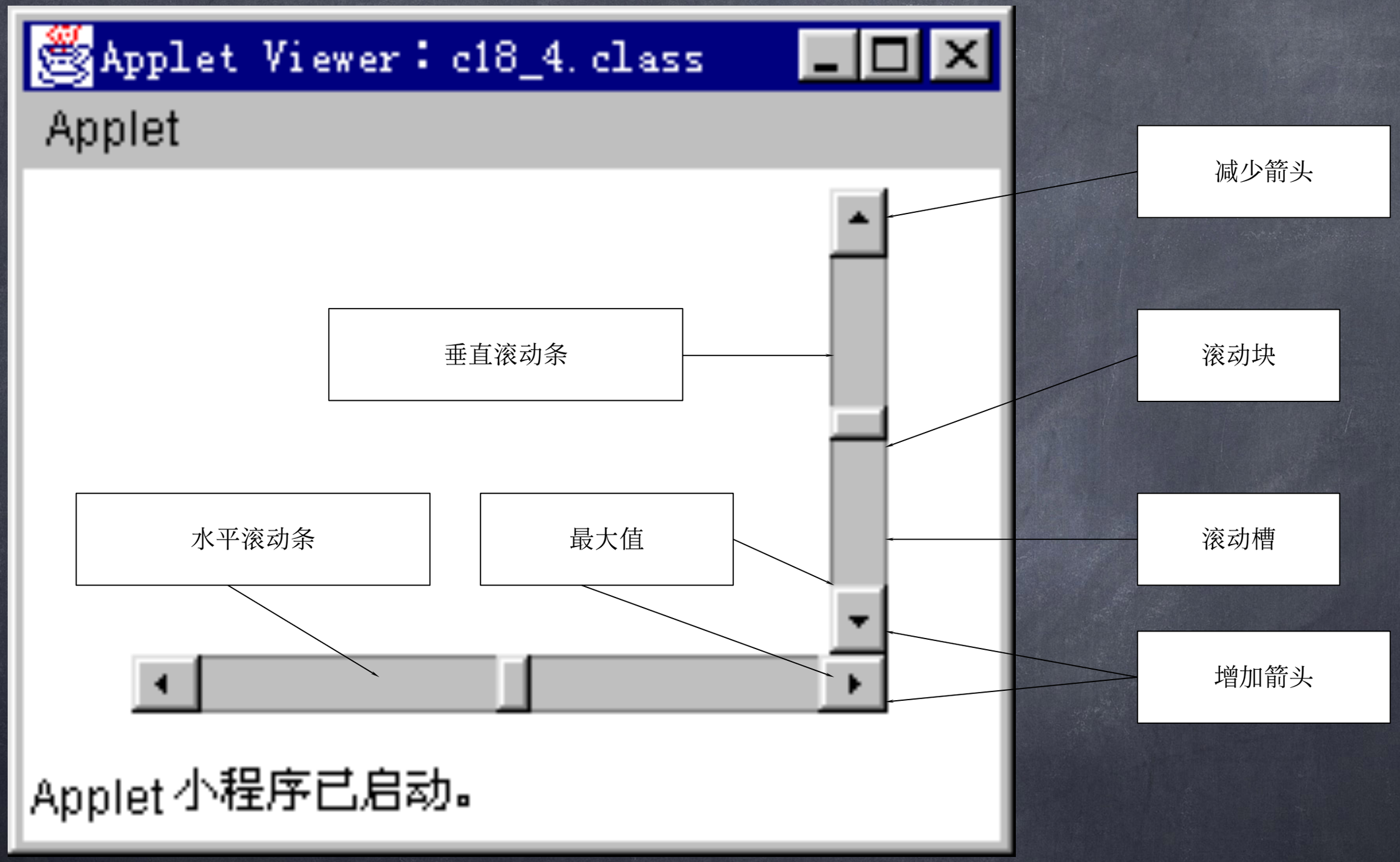
5 JScrollPane

- JScrollPane设置滚动条的参数：

1. HORIZONTAL_SCROLLBAR_ALWAYS
2. VERTICAL_SCROLLBAR_ALWAYS
3. HORIZONTAL_SCROLLBAR_NEVER
4. VERTICAL_SCROLLBAR_NEVER
5. HORIZONTAL_SCROLLBAR_AS_NEEDED
6. VERTICAL_SCROLLBAR_AS_NEEDED

5 JScrollBar组件

- JScrollPane是由JViewPort和JScrollBar组件组成的。
- JViewPort组件负责显示内容的区域大小；
- JScrollBar组件产生窗口滚动条，让用户看到整个内容。


Applet Viewer：c18_4.class
Applet
减少箭头
垂直滚动条
滚动块
水平滚动条
最大值
滚动槽
增加箭头
Applet小程序已启动。

- 常用的JScrollbar构造方法

JScrollbar(int orientation, int value, int extent,
int minimum, int maximum)

- 参数orientation可为JScrollbar.VERTICAL或JScrollbar.HORIZONTAL，用于指定滚动条的方向；
- 参数value用于指定滚动块的初始值，默认为0，表示开始时滚动块就设置在这个位置上；
- 参数extent用于指定滚动块的位移量，当用户在滚动块与滚动箭头之间点击时，将由这个值来确定滚动块应移动的位置；
- 参数minimum用于指定滚动槽的最小值(默认值为0)；
- 参数maximum用于指定滚动槽的最大值(默认值为100)。

- AdjustmentEvent事件响应流程：

  1. 将响应事件所需业务逻辑封装在监听器类

```
Class AdjustmentEventDemo implements AdjustmentListener
{
  public void adjustmentValueChanged(AdjustmentEvent e)
  { }
}
```

- AdjustmentEvent事件响应流程：

2. 创建事件源并注册完成所需业务逻辑的监听器

```
AdjustmentListener adj = new AdjustmentEventDemo();
JScrollBar js = new JScrollBar();
js.addAdjustmentListener(adj);
```

3. 引用监听器的方法响应事件。

6 JTabbedPane容器

- 当界面上需要放置的组件很多时，使用JTabbedPane。
- JTabbedPane容器与卡片盒类似，由多个标签框架的卡片和表明该框架的标签组成。每个标签框架和标签都自成一个系统(称为一张卡片)，可在标签框架中加入各式各样的组件及功能。
- 卡片上的标签在顶行或底部排成一行(也可以在左边或右边排成一列)，当用鼠标点击某一个标签时，这个标签所在的卡片(标签框架窗口)就会被翻到最上面，显示出此框架的内容。

JTabbedPane类的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JTabbedPane

- 可以使用下述两个构造方法创建JTabbedPane类的对象：

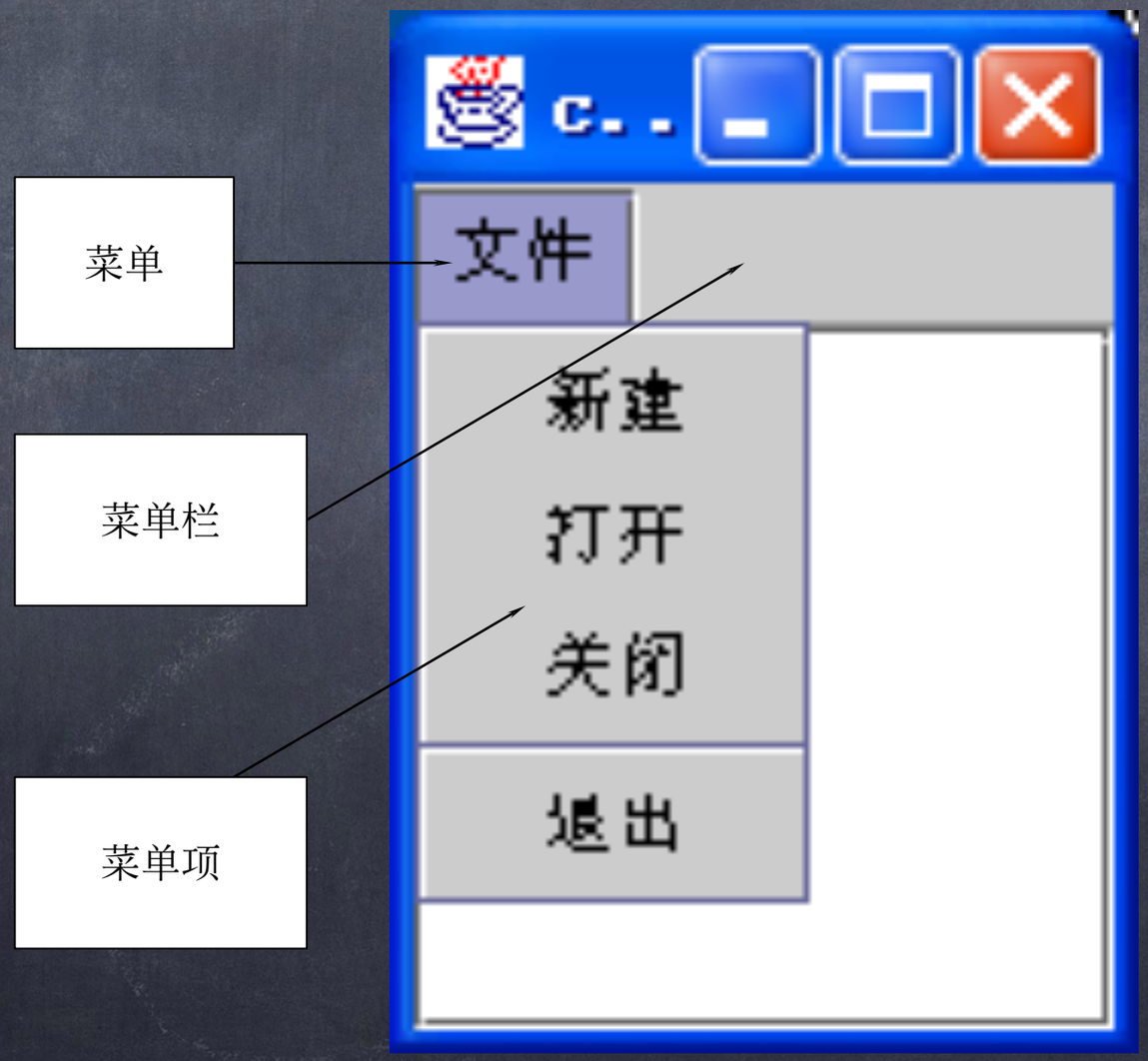
- JTabbedPane()：创建一个空的对象

- JTabbedPane(int tapposition)：创建一个空的JTabbedPane对象，并指定标签的位置，如TOP、BOTTOM、LEFT或RIGHT。

7 菜单设计

- 菜单和工具栏可提供简单明了的指示说明，让用户顺利地完成软件的操作。利用菜单可以将程序功能模块化。
- Java提供了多种样式的菜单，如一般式、复选框式、快捷键式及弹出式菜单等。这里我们仅介绍一般式菜单。

- 在Java中，一个一般式菜单由如图所示的菜单栏(JMenuBar)、菜单(JMenu)和菜单项(JMenuItem)三类对象组成。



java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JMenuBar

javax.swing.AbstractButton

javax.swing.JMenuItem

javax.swing.JMenu

7.1 菜单栏(JMenuBar)

- 菜单栏

  封装与菜单栏相关的各项操作，它只用来管理菜单，不参与交互式操作。Java应用程序中的菜单都包含在一个菜单栏对象之中。

- 创建菜单栏对象的构造方法只有一个。

  JMenuBar( )

  JMenuBar需要结合至少一个以上的JMenu组件才会在画面上显现出视觉效果。

7.2 菜单(JMenu)

- 菜单

  存放和整合菜单项(JMenuItem)的组件，构成菜单不可或缺的组件之一。

- 菜单的结构形式

  可是单一层次的结构，也可是多层次的结构，具体使用何种形式的结构则取决于界面设计上的需要。

- 创建菜单对象使用JMenu类的下述构造方法

构造方法	说明
JMenu( )	创建一个空标签对象
JMenu(String text)	使用指定的标签创建一个对象
JMenu(String text, Boolean b)	使用指定的标签创建一个对象，并给出此菜单是否具有下拉式的属性
JMenu(Action a)	创建一个支持Action的对象

7.3 菜单项(JMenuItem)

- 菜单项

  封装与菜单项相关的操作，它是菜单系统中最基本的组件。

- 菜单项继承AbstractButton类，因此具有许多AbstractButton的特性，JMenuItem是一种特殊的按钮。所以JMenuItem支持了许多在按钮上好用的功能。

- 创建菜单项对象的构造方法

构造方法	说明
JMenuItem()	使用空标签构造一个菜单项
JMenuItem(Action a)	创建一个支持Action的菜单项
JMenuItem(String text)	使用指定的标签创建一个菜单项
JMenuItem(Icon icon)	创建一个指定图标的菜单项
JMenuItem(String text, Icon icon)	创建一个指定标签和图标的菜单项
JMenuItem(String text, int mnimonic)	创建一个指定标签和键盘设置快捷的菜单项

- 制作菜单的一般步骤

1. 创建一个JMenuBar对象并将其放置在一个JFrame中；
2. 创建JMenu对象；
3. 创建JMenuItem对象并将其添加到JMenu对象中；
4. 把JMenu对象添加到JMenuBar中。

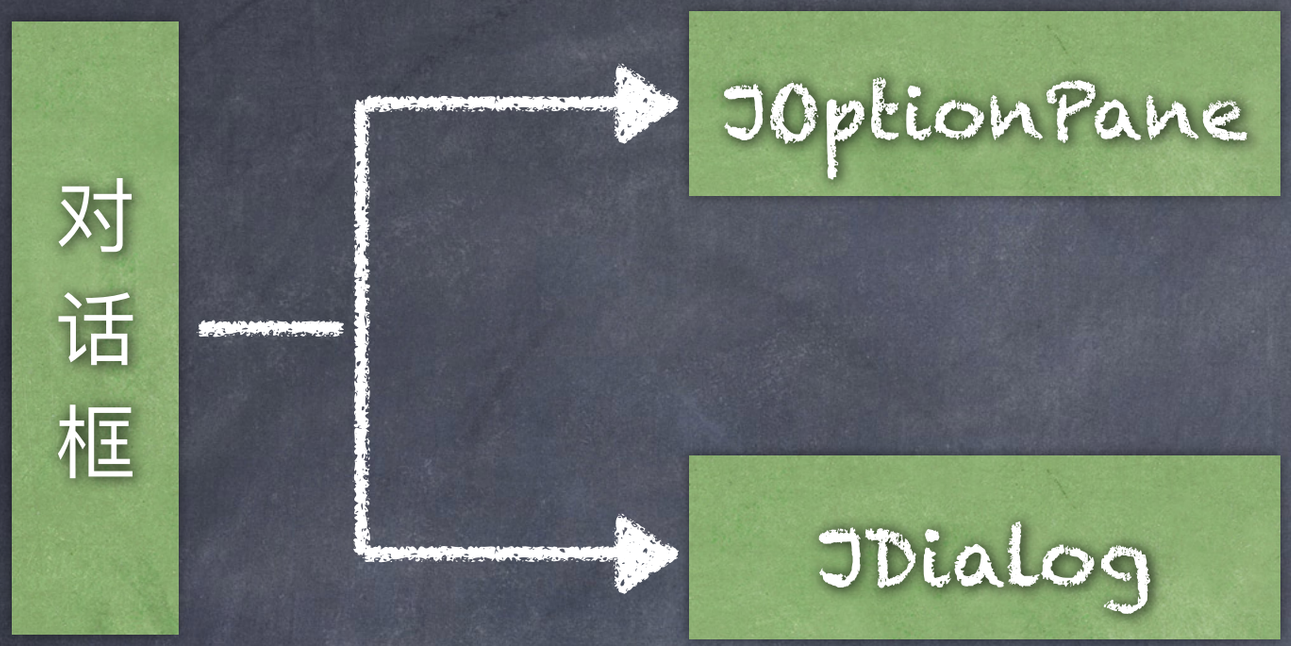
对话框设计

- 对话框

  向用户显示信息并获取程序继续运行所需数据的窗口，可以起到与用户交互的作用。

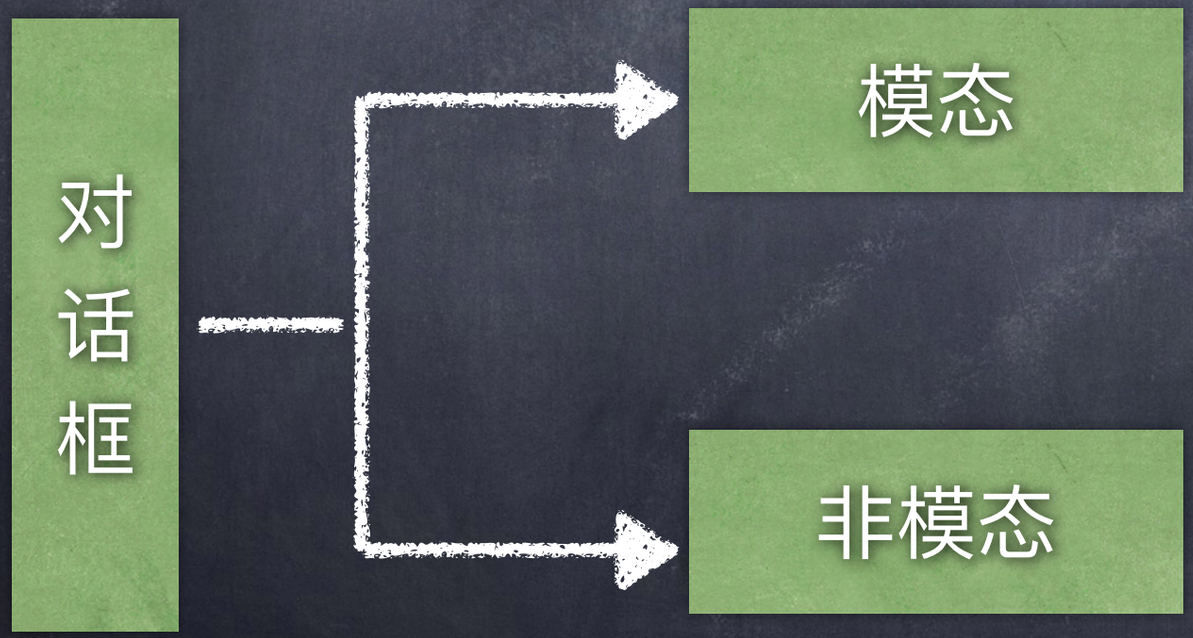
- 注意事项

  1. 对话框有边框、有标题且独立存在的容器，不能被其他容器所包容。
  2. 对话框不能作为程序的最外层容器，也不能包含菜单。
  3. Java的对话框上没有最大化和最小化按钮。



提供现成的对话框样式供用户选择

自定义方式设计对话框



要求用户必须对该对话框作出响应
关闭对话框后才能回到原程序继续

没有要求

8.1 JOptionPane

- 用户只需使用该类提供的静态方法，指定方法所需要的参数，JOptionPane对话框就能轻易的显示出来。

- JOptionPane类的继承关系如下：

java.lang. Object

  java.awt.Component

    java.awt.Container

      javax.swing. JComponent

        javax.swing.JOptionPane

构造方法	说明
JOptionPane()	创建一个显示测试信息的JOptionPane组件
JOptionPane(Object message)	创建一个显示特定信息的JOptionPane组件
JOptionPane(Object message, int messageType)	创建一个显示测试信息的JOptionPane组件，并设置信息类型
JOptionPane(Object message, int messageType, int optionType)	创建一个显示测试信息的JOptionPane组件，设置信息与选项类型
JOptionPane(Object message, int messageType, int optionType, Icon icon)	创建一个显示测试信息的JOptionPane组件，设置信息与选项类型，且可显图案
JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options)	创建一个显示测试信息的JOptionPane组件，设置信息与选项类型，且可显图案，选项值可用作更改按钮上文字
JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options, Object initialValue)	创建一个显示测试信息的JOptionPane组件，设置信息与选项类型，且可显图案，选项值可用作更改按钮上文字，并设置默认按钮

根据对话框用途

- **Message** — 提示信息
- **Confirm** — 用户确认
- **Input** — 输入数据
- **Option** — 用户自定义类型

8.1.1 Message Dialog

- Message Dialog是提示信息对话框。这种对话框中通常只含有一个"确定"按钮。

静态方法	说明
void showMessageDialog(Component parentComponent, Object message)	parentComponent：是指产生对话框的组件类型，通常是指Frame或Dialog组件； message：是指要显示的组件，通常是String或Label类型； title：对话框标题上显示的文字； messageType：指定信息类型(图标及字符串)； icon：自定义的图标
void showMessageDialog(Component parentComponent, Object message, String title, int messageType)	
void showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)	
void showInternalMessageDialog(Component parentComponent, Object message)	
void showInternalMessageDialog(Component parentComponent, Object message, String title, int messageType)	
void showInternalMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)	

8.1.2 Confirm Dialog

- 确认对话框，这类对话框通常会询问用户一个问题，要求用户作YES/NO的回答。

静态方法
void showConfirmDialog(Component parentComponent, Object message)
void showConfirmDialog(Component parentComponent, Object message, String title, int optionType)
void showConfirmDialog(Component parentComponent, Object message, String title, int messageType, int optionType, Icon icon)
void showInternalConfirmMessageDialog(Component parentComponent, Object message)
void showInternalConfirmMessageDialog(Component parentComponent, Object message, String title, int optionType, int messageType)
void showInternalConfirmMessageDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)

8.1.3 Input Dialog

- 输入对话框，这类对话框可以让用户输入相关的信息，当用户完成输入并按下确定按钮后，系统会得到用户所输入的信息。

- 输入对话框不仅可以让用户自行输入数据，也可以提供ComboBox组件让用户选择相关信息，避免用户输入错误。

静态方法
void showInputDialog(Object message)
void showInputDialog(Component parentComponent, Object message)
void showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialselectionValue)
void showInternalInputMessageDialog(Object message)
void showInternalInputMessageDialog(Component parentComponent, Object message, String title, int messageType)
void showInternalInputMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialselectionValue)

8.1.4 Option Dialog

- 选择对话框，可以让用户自已定义对话框的类型。它的最大好处是可以改变按钮上的文字。

静态方法
int showOptionDialog(Component parentComponent,Object message,String title,int optionType,int messageType,Icon icon,Object[ ] options,Object initialValue)
int showInternalOptionDialog(Component parentComponent,Object message,String title,int optionType,int messageType,Icon icon,Object[ ] options,Object initialValue)

8.2 JDialog

- 如果JOptionPane提供的样式无法满足我们的需求，就需要使用JDialog来自行设计对话框。
- JDialog是java.awt.Dialog子类，继承关系如下

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Dialog
          javax.swing.JDialog