第9章

图形用户界面（GUI）

面向对象程序设计

9.1 Java图形用户界面概述

- 顾名思义，图形用户界面（Graphics User Interface，缩写GUI）是指以图形的显示方式与用户实现交互操作的应用程序界面。Java提供了十分完善的图形用户界面功能，使得软件开发人员可以轻而易举地开发出功能强大、界面友善、安全可靠的应用软件。

面向对象程序设计

在Java语言中，有两个包（java.awt和javax.swing）囊括了实现图形用户界面的所有基本元素，这些基本元素主要包括容器、组件、绘图工具和布局管理器等。组件是与用户实现交互操作的部件，容器是包容组件的部件，布局管理器是管理组件在容器中布局的部件，绘图工具是绘制图形的部件。

java.awt是java1.1用来建立GUI的图形包，这里的“awt”是抽象窗口工具包（Abstract Windowing Toolkit）的缩写，其中的组件常被称为AWT组件。

javax.swing是Java2提出的AWT的改进包，它主要改善了组件的显示外观，增强了组件的控制能力。

面向对象程序设计

在Java中，设计用户界面需要经历4个基本步骤：

（1）创建和设置组件

（2）将组件加入到容器中

（3）布局组件

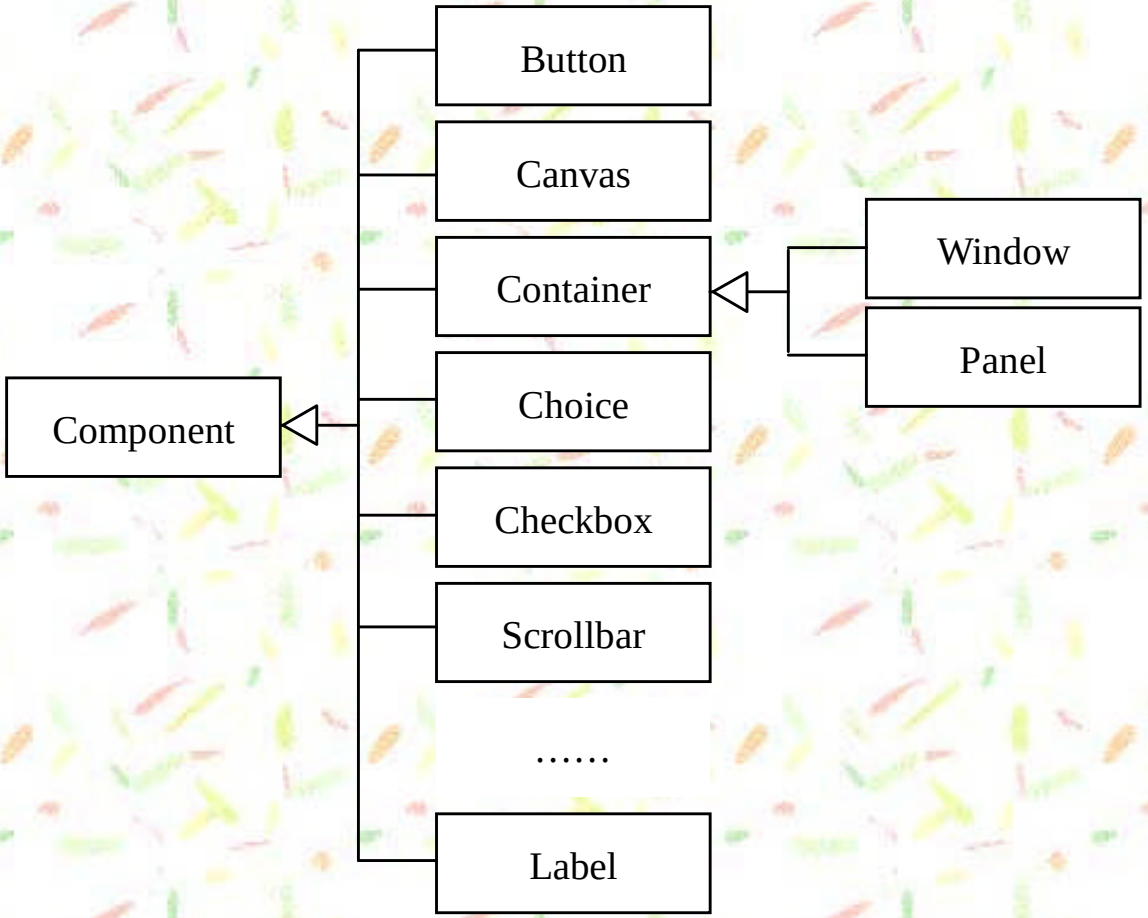
（4）处理由组件产生的事件

9.2 用AWT创建图形用户界面

- AWT是抽象窗口工具包，其中涵盖了Java API为开发Java应用程序提供的创建图形用户界面的工具集，它包含了用户界面的各种组件、事件处理模型、图形和图像处理工具、布局管理器、数据传输、剪切和粘贴操作等功能，使得在Java环境中可以比较轻松地设计出具有良好的用户交互界面的应用程序。

9.2.1 AWT概述

- AWT是Java基础类库JFC（Java Foundation Class）的一个重要组成部分，它位于java.awt包中，其中不仅包含了与显示界面有关的各种组件，还包含了一些子包，主要提供了色彩控制、数据传输、事件处理模型、拖放功能、字体设置、打印管理、图像处理等技术支持。
- AWT使用的是与运行环境相关的组件处理机制。也就是说，在应用程序中使用的各种组件需要在运行环境中有相应的本地组件与之配合，共同完成其功能。



AWT组件类结构层次图

面向对象程序设计

9.2.2 AWT容器

容器（Container）是用来放置其他组件的一种特殊部件，在Java中容器用Container 类描述，它是Component的一个子类，因此，容器也具有组件的全部特征，是一种具有特殊作用意义的组件。

Container类的部分成员方法

int getComponentCount()

Component[] getComponents()

Component add(Component comp)

void remove(Component comp)

LayoutManager getLayout()

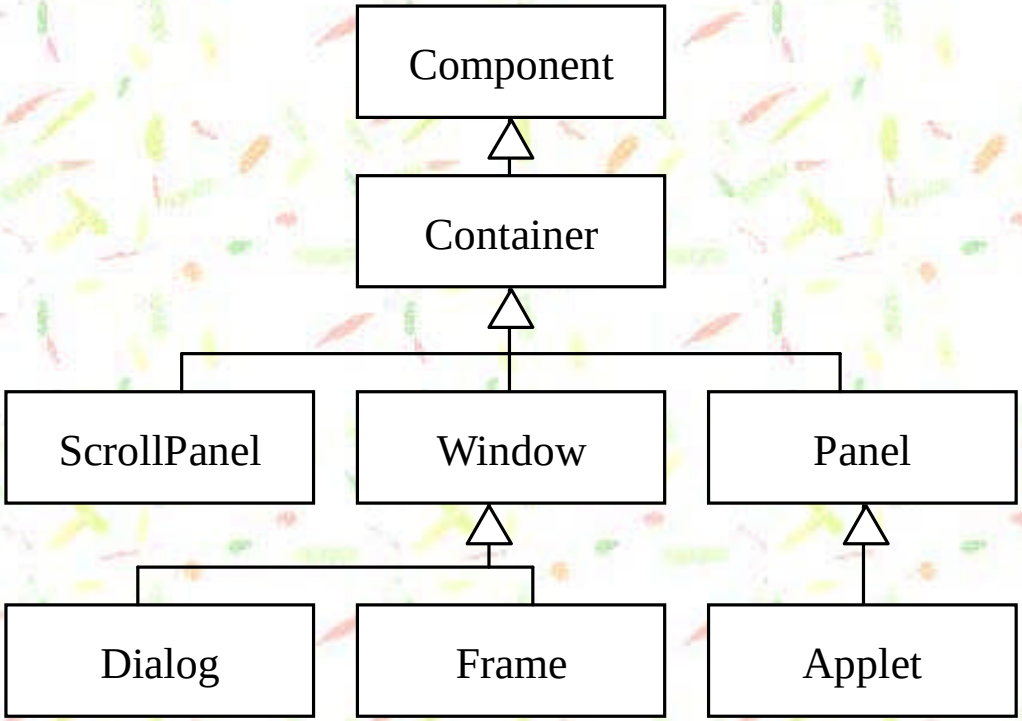
void setLayout(LayoutManager mgr)

Dimension getPreferredSize()

Dimension getMinimumSize()

Dimension getMaximumSize()

void paint(Graphics g)


Component
Container
ScrollPanel
Window
Panel
Dialog
Frame
Applet

容器类的层次结构图

1. Panel容器

Panel容器是一种最简单且无边框的容器，又称为面板容器。在这种容器中可以放置各种类型的组件，甚至是另外一个Panel容器，所以Panel容器可以嵌套地放置多层。Panel类提供了两种构造方法，其格式为：

Panel()

Panel(LayoutManager layout)

2．Frame容器

在java.awt包中提供了一个Window 类，这个类描述了无边框、无菜单栏的顶层窗口容器。由于一般的应用程序窗口都有边框、标题栏和菜单栏，所以通常使用它的子类Frame，这个类描述了一个包含边框和标题栏的顶层窗口。

Frame类提供了四种构造方法的格式：

Frame()

Frame(GraphicsConfiguration gc)

Frame(String title)

Frame(String title, GraphicsConfiguration gc)

Frame类的部分成员方法

String getTitle()

void setTitle(String title)

MenuBar getMenuBar()

void setMenuBar(MenuBar mb)

Rectangle getMaximizedBounds()

void setMaximizedBounds(Rectangle bounds)

9.2.3 AWT组件

尽管AWT容器也属于组件，但它的主要作用是用来包容其他组件的。下面将讨论一下非容器组件，即那些必须放置在容器中且相互之间不能嵌套的组件。为了避免混淆，在这里我们将前者称为容器，后者称为组件。

Java提供了大量的组件，它们都是Component的子类。尽管每种组件都有其特点和适用场合，但它们的使用过程都要经历创建组件对象、将组件添加到容器中、设置显示属性、设置事件监听，截获操作结果等几个步骤。

1．Label组件

Label被称为标签组件，它是一种用来显示说明性的静态文本的组件。用户不能直接地编辑它，但可以在应用程序中，通过调用Label提供的成员方法更换文本的内容。

Label类提供了三种构造方法的格式：

Label()

Label(String text)

Label(String text, int alignment)

alignment可以是Label类定义的整型常量，比如：LEFT（居左）、CENTER（居中）和RIGHT（居右）。

Label类的部分成员方法

String getText()

void setText(String text)

int getAlignment()

void setAlignment(int alignment)

2．TextField组件

TextField被称为单行文本组件，它是一种用来接收用户输入的组件。TextField是TextComponent的子类。在TextComponent类中描述了可编辑的文本串，并提供了一组成员方法，用来决定组件是否可编辑。如果可编辑，可以利用提供的相关成员方法对文本内容进行编辑操作。

TextComponent类的部分成员方法

void setText(String t)

String getText()

String getSelectedText()

boolean isEditable()

void setEditable(boolean b)

Color getBackground()

void setBackground(Color c)

面向对象程序设计

TextField类提供的构造方法和成员方法

TextField()

TextField(String text)

TextField(int columns)

TextField(String text, int columns)

char getEchoChar()

void setEchoChar(char c)

int getColumns()

void setColumns(int columns)

3．Button组件

Button组件被称为按钮组件，它是图形用户界面中非常重要的一种基本组件。主要用来激活用户编写的某项操作，即当用户按下某个按钮后，应用程序就会立即执行某个代码段。

在Button类中提供了两个构造方法。

Button()

Button(String label)

==========

在Button类中提供了两个与按钮标签有关的成员方法。

String getLabel()

void setLabel(String label)

4．Checkbox组件

Checkbox被称为复选组件，它是一种图形化的、可以设置“on”和“off”两种状态的组件。用鼠标点击一个复选按钮，可以使其在“on”和“off”之间进行切换。不仅如此，还可以将几个复选按钮利用CheckboxGroup组件绑定成一组，使得每一组复选按钮在每一时刻只有一个处于“on”状态。

在Checkbox类中，提供了四种构造方法的格式：

Checkbox()

Checkbox(String label)

Checkbox(String label, boolean state)

Checkbox(String label, boolean state,
CheckboxGroup group)

Checkbox类的部分成员方法

String getLabel()

void setLabel(String label)

boolean getState()

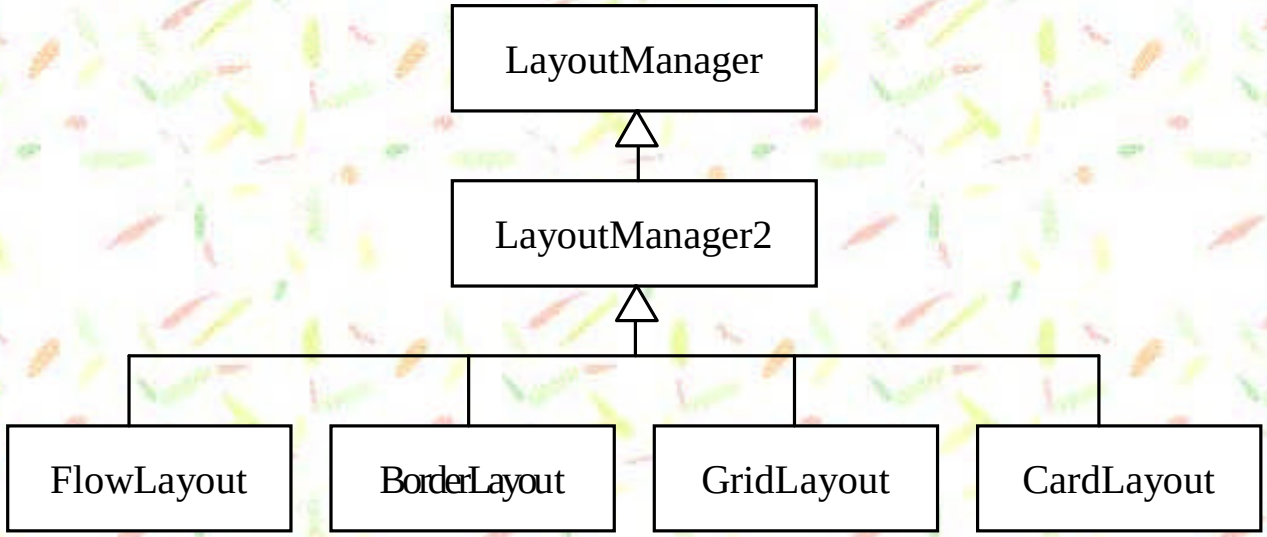
void setState(boolean state)

CheckboxGroup getCheckboxGroup()

void setCheckboxGroup(CheckboxGroup g)

9.2.4 布局管理器

所谓布局管理器是指按照指定的策略，安排并管理组件在容器中排列位置的一种特殊对象。



面向对象程序设计

LayoutManager接口中声明的有关布局管理器操作的部分抽象成员方法

void addLayoutComponent(String name,Component comp)

void removeLayoutComponent(Component comp)

Dimension preferredLayoutSize(Container parent)

Dimension minimumLayoutSize(Container parent)

1．FlowLayout布局管理器

FlowLayout是Panel容器的默认布局管理器。它按照从上到下，从左到右的规则，将添加到容器中的组件依次排列。如果一行中没有足够的空间放置下一个组件，FlowLayout换行后，将这个组件放置在新的一行上。另外，在创建FlowLayout对象时，可以指定一行中组件的对齐方式，默认为居中，还可以指定每个组件之间，在水平和垂直方向上的间隙大小，默认值为5个像素。这种布局管理器并不调整每个组件的大小以适应容器的大小，而是永远保持每个组件的最佳尺寸，剩余的空间用空格填补。

在FlowLayout类中，提供了构造方法和部分成员方法：

FlowLayout()

FlowLayout(int align)

FlowLayout(int align, int hgap, int vgap)

int getAlignment()

void setAlignment(int align)

int getHgap()

void setHgap(int hgap)

int getVgap()

void setVgap(int vgap)

2．BorderLayout布局管理器

BorderLayout是Frame和Dialog两种容器的默认布局管理器，它将容器分为5个部分，分别命名为North、South、West、East和Center。

BorderLayout类提供了两种格式的构造方法：

BorderLayout()

BorderLayout(int hgap, int vgap)

面向对象程序设计

3．GridLayout布局管理器

GridLayout是一种很容易理解的布局管理器，它将容器按照指定的行数、列数分成大小均匀的网格，且放入容器中的每个组件的大小都一样。将组件添加到容器中可以有两种基本方法：一是使用默认的布局顺序，即按照从上到下，从左到右的次序将组件放入容器的每个网格中；二是采用add(Component comp,int index)将组件放入到指定的网格中。

GridLayout类提供了三种格式的构造方法：

GridLayout()

GridLayout(int rows,int cols)

GridLayout(int rows,int cols,int hgap,int vgap)

4. CardLayout布局管理器

CardLayout是一种将每个组件看作一张卡片，且将所有卡片码放成一摞，每一时刻只有一张卡片被显示的布局管理器。有人将其形象地描述为一副落成一叠的扑克牌。第一个添加到容器中的组件位于最低层，最后一个添加到容器中的组件位于最上层。

面向对象程序设计

9.3 用Swing创建图形用户界面

Swing是在AWT基础上发展而来的，目前越来越多的人偏爱使用Swing组件，这是因为Swing是Java图形用户界面工具进步的象征，是AWT必然的替代品。

9.3.1 Swing概述

与AWT相比较，Swing具有以下几点优势：

（1）AWT是基于同位体（Peer）的体系结构，这种设计策略严重限制了用户界面中可以使用的组件种类及功能，成为一个致命的缺憾；而Swing不需要本地提供同位体，这样可以给设计者带来更大的灵活性，有利于增强组件的功能。

（2）在AWT中，有一部分代码是用C编写的；而Swing是100%的纯Java，增强了应用程序的与环境无关性。

（3）Swing具有控制外观（Pluggable look and feel）的能力，即允许用户自行定制桌面的显示风格，比如，更换配色方案，让窗口系统更加适应用户的习惯和需要，而AWT 组件完全依赖于本地平台。

（4）增加了裁剪板、鼠标提示和打印等功能。

所有Swing组件类都存在于javax.swing包中。为了避免混淆，Swing包中的所有类名均在AWT类名的前面冠于“J”字符，例如，JPanel、JFrame、JButton等。

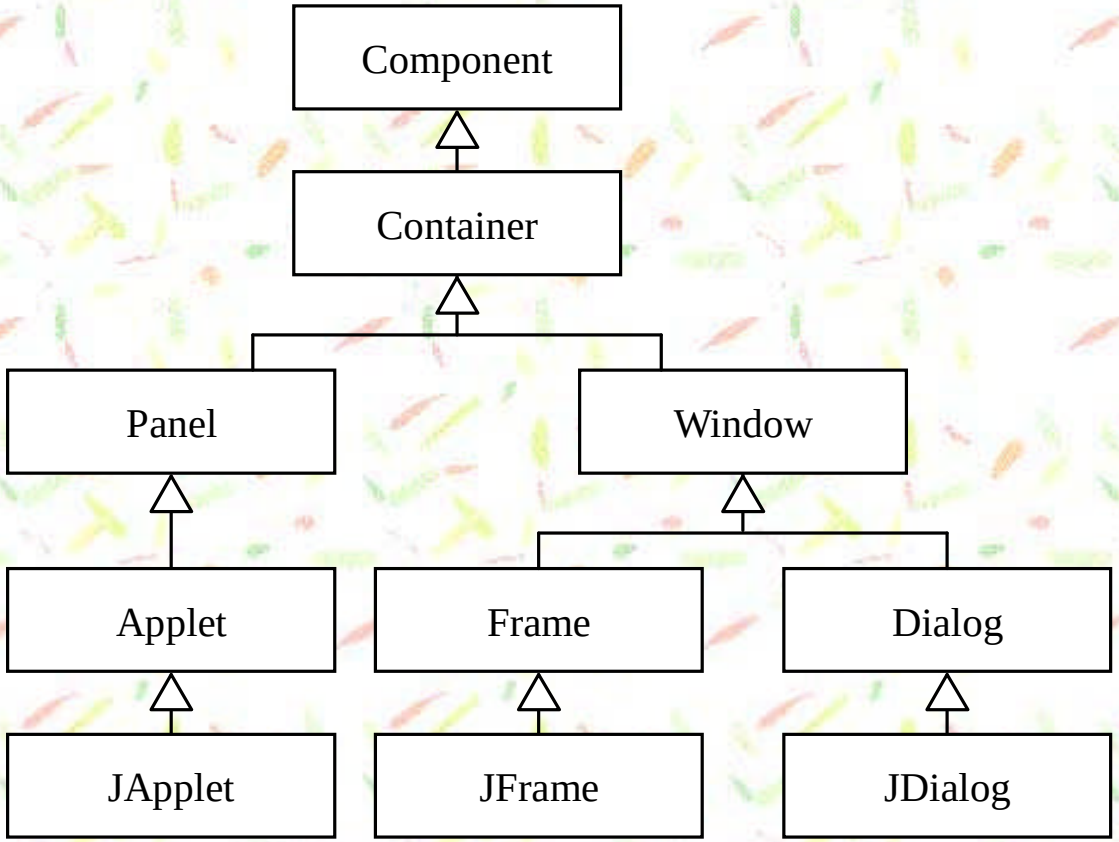
（5）Container是抽象容器类，所以派生于它的组件都应用具有容器的功能。

9.3.2 Swing容器

尽管所有的Swing都属于容器，但还是有几种专门用于作为容器的组件。它们被分成顶层容器、通用容器和专用容器三个类别，其中顶层容器和通用容器是常用的两类容器形式。

1. 顶层容器

每一个应用Swing组件的应用程序都至少要有一个顶层容器。大家都知道，一个容器可以包含其他的容器，即多个容器之间可以具有嵌套关系，这样就构成了一个层次结构。所谓顶层容器是指最外层的容器，即包含所有组件或容器的那层容器。如果我们将这个容器层次结构用树型结构描述的话，顶层容器就是这棵树的根。



Swing顶层容器类层次结构图

面向对象程序设计

在使用这些顶层容器时，需要注意以下几点：

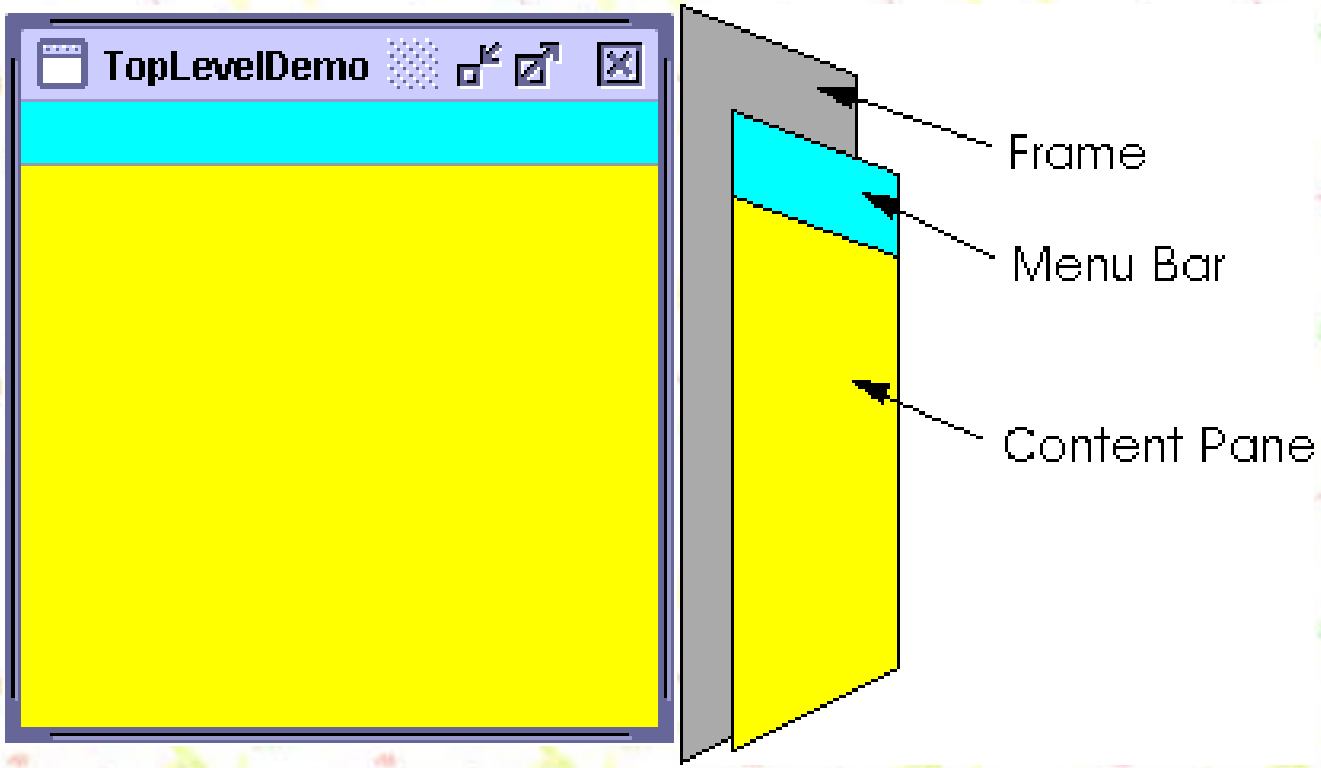
（1）为了能够在屏幕上显示，每个GUI组件都必须位于一个容器层级结构中。

（2）每个GUI组件只能被添加到一个容器中。如果一个组件已经被添加到一个容器中，又把它添加到另外一个容器中，则它将首先被从第一个容器中删除，然后再移入第二个容器。

（3）每个顶层容器都包含一个内容窗格（Content pane），所有的可视组件都必须放在内容窗格中显示。可以调用顶层容器中getContentPane()方法得到当前容器的内容窗格，并使用add()方法将组件添加到其中。

（4）可以在顶层容器中添加菜单栏，它将位于顶层容器的约定位置。例如，在Window环境下，菜单栏位于窗口标题栏的下面。

面向对象程序设计



窗口框架容器、内容窗格和菜单栏的位置关系

面向对象程序设计

在Swing中，用JFrame类实现窗口框架。正像前面讲述的那样，不能直接将可视组件放置在顶层容器中，而需要与内容窗格（ContentPane）配合使用。

在JFrame类中，提供了两种格式的构造方法：

JFrame()

JFrame(String title)

JFrame类的部分成员方法

int getDefaultCloseOperation()

void setDefaultCloseOperation()

void pack()

Dimension getSize()

void setSize(int width,int height)

void setSize(Dimension size)

Rectangle getBounds()

void setBounds(int xleft,int yleft,int width,int height)

void setBounds(Rectangle size)

Container getContentPane()

JMenuBar getJMnuBar()

2．通用容器

通用容器包含了一些可以被使用在许多不同环境下的中间层容器。主要包括面板容器（Panel）、带滚动条的视口容器（ScrollPane）、工具栏（ToolBar）等。Swing中，分别用JPanel、JScrollPane和JToolBar类实现，它们都是JComponent的子类，且通常被放置在其他容器中。

（1）面板容器（Panel）

面板容器是一种常用的容器种类。在默认情况下，除了背景外，它不会自行绘制任何东西。当然，可以通过相应的方法很方便地为它添加边框，或定制希望绘制的内容。

在默认情况下，面板容器是不透明的，这使得它具有类似内容窗格的特点，且有助于提高绘图效率。当然，也可以调用setOpaque()方法，将其设置为透明的。如果面板容器是透明的，将没有背景，这样可以使位于该容器覆盖区域下面的组件显示出来。

（2）带滚动视口容器（ScrollPane）

在Swing中，用JScrollPane类实现了具有滚动功能的视口容器。由于屏幕大小的限制，有些组件不能在一屏中全部显示出来，或欲显示内容的大小动态地发生变化，因此可以使用带滚动功能的视口容器，利用它提供的滚动条移动窗口在组件上的位置，将组件的全部内容分区域地显示出来。

9.3.3 Swing组件

在Swing中，所有的组件都是JComponent类的子类，这个类为它的所有子类提供了下列功能：

工具提示、绘画和设置边框、可控制显示外观、 定制属性、支持布局管理器、支持拖拽功能、双缓冲、可访问性、击键绑定。

与AWT组件相比较， Swing组件增加了以下几个新功能：

（1）按钮和标签组件不仅可以显示文本串，还可以显示图片。

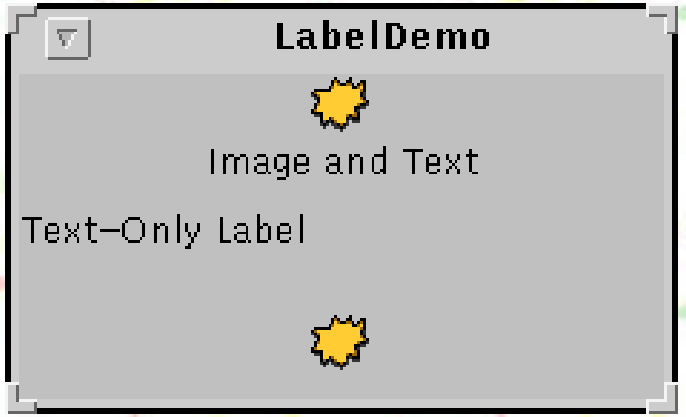
（2）可以轻松地为大多数组件添加或更改边框。例如，可以轻而易举地在一个容器或标签的周围添加一个边框。

（3）可以通过调用内部的成员方法或创建一个子类，改变Swing组件的外观和行为。

（4）Swing组件不必要求一定是矩形，例如，可以创建一个圆形按钮。

一．标签（Label）

在AWT中，我们已经讲过，标签是一种不对任何事件响应的组件，它主要被用来实现一些说明性的描述。在Swing中，用JLabel类实现标签组件，它的显示形式得到了扩展，它不仅可以显示文字，还可以显示图片。



JLabel提供的主要构造方法

JLabel()

JLabel(Icon icon)

JLabel(Icon icon,int horizontaAlignment)

JLabel(String text)

JLabel(String text, int horizontaAlignment)

JLabel(String text,Icon icon,
int horizontaAlignment)

面向对象程序设计

JLabel类的部分成员方法

String getText()

void setText(String text)

Icon getIcon()

void setIcon(Icon icon)

int getHorizontaAlignment()

void setHorizontalAlignment(int alignment)

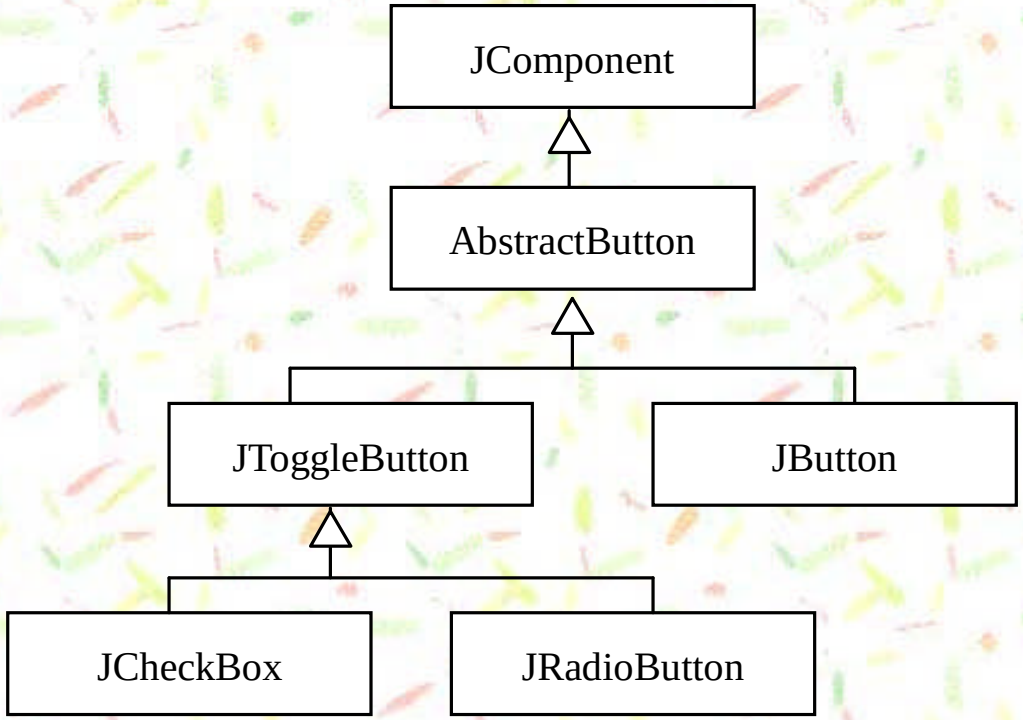
int getVerticalAlignment()

void setVerticalAlignment(int alignment)

二．按钮（Button）

Swing按钮既可以显示文字，也可以显示图像，并且每个按钮中的文字可以相对于图像显示在不同的位置，每个按钮中带下划线的字母是快捷键，例如，点击ALt-M等价于用鼠标点击中间的按钮。当按钮被禁用时，自动地变成禁用的外观，而且还可以提供一幅专门用于按钮禁用状态的图像。

在Swing中，包含多种形式的按钮，它们都以AbstractButton作为父类。



面向对象程序设计

抽象类AbstractButton类的部分成员方法

boolean isSelected()

void setSelected(boolean b)

String getText()

void setText(String text)

Icon getIcon()

void setIconb(Icon icon)

Icon getDisabledIcon()

void setDisabledIcon(Icon icon)

Icon getPressedIcon()

void setPressedIcon(Icon icon)

面向对象程序设计

使用按钮组件需要经过下列基本步骤：

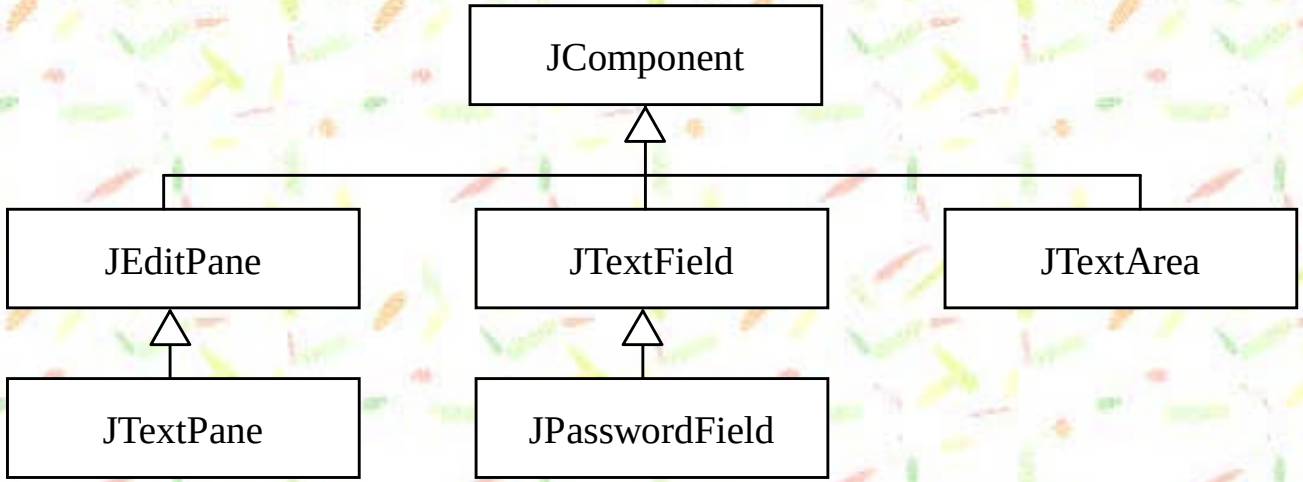
（1）创建按钮对象；

（2）将按钮对象添加到容器中；

（3）设置响应点击按钮的操作。

三．文本框（Text field）

文本框是接收用户输入的一种组件，在Swing中提供了多种文本框组件，它们都是由JTextComponent类派生的子类实现。



JTextField类提供构造方法

JTextField()

JTextField(String text)

JTextField(String text,int col)

JTextField(int col)

面向对象程序设计

JTextField类的部分成员方法

String getText()

void setText(String text)

boolean isEditable()

void setEditable(boolean editable)

int getColumns()

void setColumns(int col)

四．组合框（Combo box）

组合框允许用户从若干个选项中选择一项。在Swing中用JComboBox类实现，它提供了两种不同形式的组合框。一种是不可编辑的组合框，它由一个按钮和下拉列表组成，这是默认形式；另一种是可编辑的组合框，它由一个可接收用户输入的文本框、按钮和下拉列表组成，用户既可以在文本框中输入文本串，也可以点击按钮，打开下拉列表。

JComboBox类提供的构造方法和成员方法

JComboBox()

JComboBox(Object [] item)

JComboBox(Vector item)

void addItem(Object item)

void insertItemAt(Object item,int index)

Object getItemAt(int index)

Object getSelectedItem()

void removeItem(Object item)

int getItemCount()

面向对象程序设计

9.4 事件处理机制

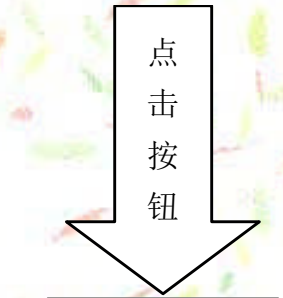
Java采用的是事件处理机制，即程序的运行过程是不断地响应各种事件的过程，事件的产生顺序决定了程序的执行顺序，这是图形用户界面应用程序最重要的部分，是实现各种操作功能的重要途径。

9.4.1 Java事件处理机制

程序的执行过程由用户对GUI的操作行为控制。例如，点击鼠标、敲击键盘、移动窗口等，所有这些用户行为都会引发特定的操作行为。在任意给定的时刻，应用程序下一步执行哪条代码是未知的，它将取决于未来发生什么事件。

可以这样说，产生事件是Java程序执行各种操作的前提。用户敲击一下键盘或点击一下鼠标都会产生事件，这些事件产生后，首先由操作系统鉴别。对于每个由于用户的操作行为产生的事件，操作系统都要决定这个事件将由哪个应用程序处理，并把这个事件的相关信息传递给相应的处理程序。

9.4.2 事件的处理过程



传递给

```
public class MyButtonHandler
   implements ActionListener
{
        ....
   Button.addActionListener(thsi);
      ...
    void actionPerformed(ActionEvent e)
    {
          ...
    }
}
```

面向对象程序设计

9.4.3 事件类

一个Java应用程序可以响应各种类型的事件，比如，点击按钮、拖动滚动条、极小化窗口等。为了便于处理，Java将这些事件划分成了低级事件和语义事件两个类别。

低级事件是指来自键盘、鼠标或与窗口操作有关的事件。比如，窗口极小化、关闭窗口、移动鼠标或敲击键盘。

语义事件是指与组件有关的事件，比如，点击按钮、拖动滚动条等。这些事件源于图形用户界面，其含义由程序设计员赋予，例如，一个“确定”按钮将确认刚才的操作，一个“取消”按钮将撤消刚才的操作。

一．低级事件

焦点事件、鼠标事件、键盘事件和窗口事件都属于低级事件，下面是低级事件的事件类名。

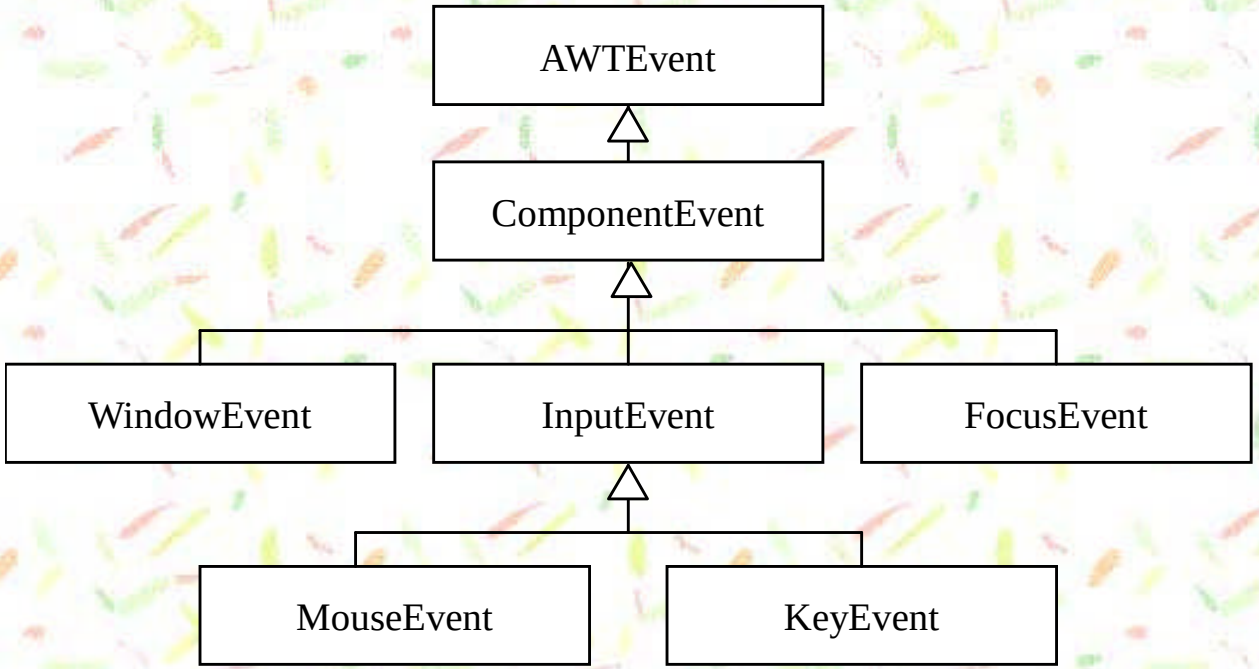
FocusEvent

MouseEvent

KeyEvent

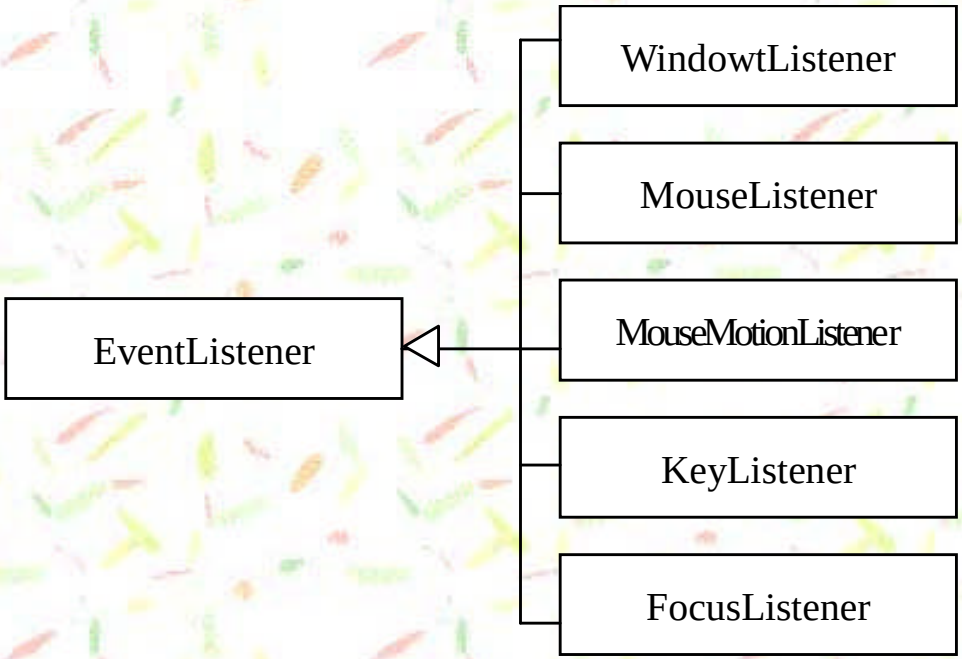
WindowEvent

事件类结构图



面向对象程序设计

低级事件的监听器接口



面向对象程序设计

二. 语义事件

下面是语义事件类和事件类名

ActionEvent

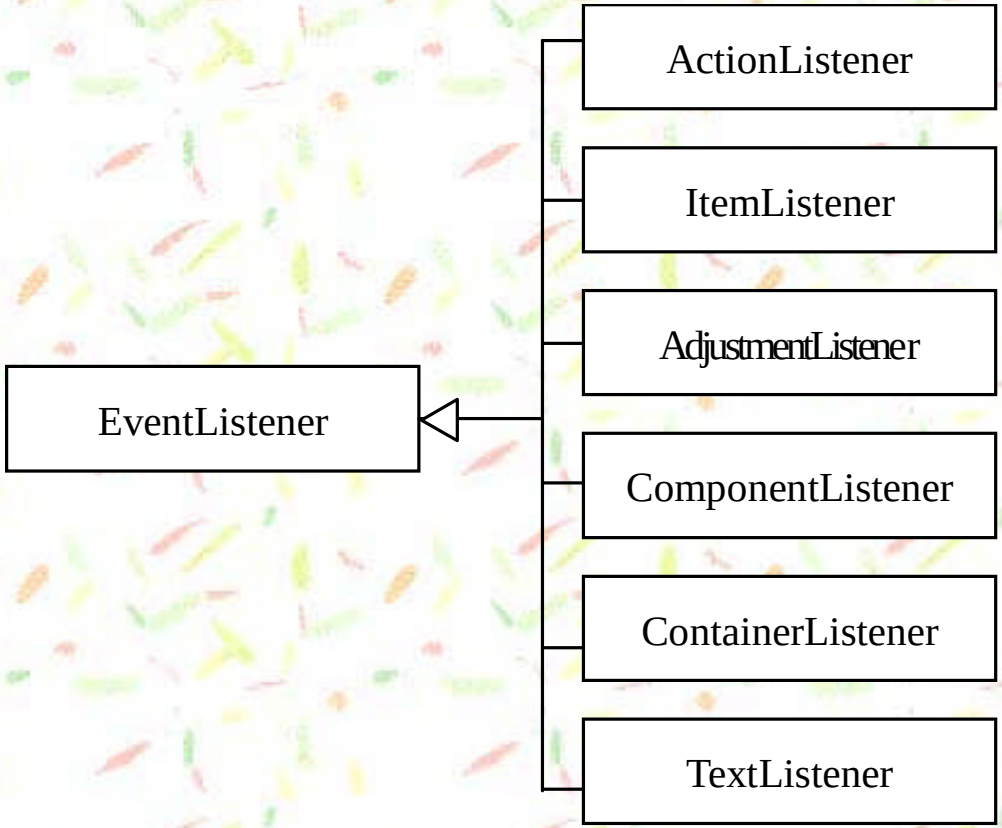
ItemEvent

ComponentEvent

ContainerEvent

TextEvent

语义事件的监听器接口



面向对象程序设计

9.4.4 窗口事件的处理

窗口事件的ID定义：

WINDOW_OPENED

WINDOW_CLOSEING

WINDOW_CLOSED

WINDOW_ACTIVATED

WINDOW_DEACTIVATED

WINDOW_ICONFIED

WINDOW_DEICONIFIED

对应于窗口事件的监听器接口为**WindowListener**，在这个接口中，包含了处理每一种具体窗口事件的方法。当某个事件发生时，将自动地调用相应的方法。

windowOpened(WindowEvent e)

windowClosing(WindowEvent e)

windowClosed(WindowEvent e)

windowActivated(WindowEvent e)

windowDeactivated(WindowEvent e)

windowIconfied(WindowEvent e)

windowDeiconfied(WindowEvent e)

面向对象程序设计

按照Java语法的规定，监视器类需要实现接口中的全部方法，这样就需要我们将没有特别操作要求的那些事件对应的成员方法设计为空。很显然，这会增加程序的复杂性，降低程序的清晰度。为了解决这个问题，Java提出了适配器的概念。所谓适配器是指API提供的一种实现了监听器接口的类。

WindowListener是窗口事件的监听器类，其中包含7个成员方法，因此**API**提供了一个适配器**WindowAdapter**，该适配器不仅实现了**WindowListener**接口，还实现了**WindowFocusListener**接口及**WindowStateListener**接口，这是我们没有提到的一种窗口状态发生变化时产生的事件的监听器类，其中只有一个方法，因此，该适配器具有处理这三种事件的功能。

面向对象程序设计

WindowAdapter类的源代码。

```
public abstract class WindowAdapter
    implements WindowListener,
       WindowStateListener, WindowFocusListener
{
   public void windowOpened(WindowEvent e) {}
   public void windowClosing(WindowEvent e) {}
   public void windowClosed(WindowEvent e) {}
   public void windowIconified(WindowEvent e) {}
   public void windowDeiconified(WindowEvent e) {}
   public void windowActivated(WindowEvent e) {}
   public void windowDeactivated(WindowEvent e) {}
   public void windowStateChanged(WindowEvent e) {}
   public void windowGainedFocus(WindowEvent e) {}
   public void windowLostFocus(WindowEvent e) {}
}
```

9.4.5 键盘事件的处理

键盘事件用java.awt.event.KeyEvent类描述，其中提供了getKeyCode()成员方法可以获得按下或释放的那个键所对应的键码；getKeyChar()成员方法可以返回按下的那个键所对应的字符；getKeyText(int)成员方法可以返回由参数带入的键码的描述信息。键盘操作可以被分成三个类别，它们分别用不同的ID标识：

KEY_PRESSED当按下键盘中的某个键时发生该事件。

KEY_RELEASED当释放按键时发生该事件。

KEY_TYPED当按下键盘中的字符键（非系统键）时发生该事件。

面向对象程序设计

处理键盘事件的监听器接口是KeyListener接口，在这个接口中，声明了对应上述三种事件的三个方法。

KeyPressed(KeyEvent) 处理KEY_PRESSED事件

KeyReleased(KeyEvent) 处理KEY_RELEASED事件

KeyTyped(KeyEvent) 处理KEY_TYPED事件

KeyListener接口对应的适配器为KeyAdapter。

面向对象程序设计

9.4.6 鼠标事件的处理

鼠标事件由MouseEvent类描述。在这个类中，提供了下面几个成员方法可以获得与鼠标操作有关的信息。

int getX()和int getY() 将返回发生鼠标事件时光标所处的坐标位置。

Point getPoint() 将以Point类型的形式返回发生鼠标事件时光标所处的位置。

int getClickCount() 将返回点击鼠标的次数。

面向对象程序设计

鼠标事件被划分成两个大类别，

一类被称为鼠标事件，用

MOUSE_EVENT_MASK标识；

另一类被称为鼠标移动事件，用

MOUSE_MOTION_EVENT_MASK标识。

它们分别对应MouseListener接口和

MouseMotionListener接口。

鼠标事件的ID

MOUSE_CLICKED

MOUSE_PRESSED

MOUSE_ENTERED

MOUSE_EXITED

MOUSE_RELEASED

MOUSE_MOVE

MOUSE_DRAGGED

面向对象程序设计

在MouseListener接口中声明了5个成员方法，对应处理属于MOUSET_EVENT_MASK事件类别的5个不同的事件。它们是：

mouseClicked(MouseEvent) 处理MOUSE_CLICKED事件

mousePressed(MouseEvent) 处理MOUSE_PRESSED事件

mouseReleased(MouseEvent) 处理MOUSE_RELEASED事件

mouseEntered(MouseEvent) 处理MOUSE_ENTERED事件

mouseExited(MouseEvent) 处理MOUSE_EXITED事件

面向对象程序设计

在MouseMotionListener接口中声明了处理MOUSE_MOTION_EVENT_MASK事件类别的2个不同事件的成员方法。

mouseDragged(MouseEvent e)

处理MOUSE_DRAGGED事件

mouseMoved(MouseEvent e)

处理MOUSE_MOVE事件。

MouseListener接口和MouseMotionListener接口对应的适配器为MouseAdapter和MouseMotionAdapter。

9.4.7 语义事件的处理

语义事件是与组件有关的一些事件。例如，点击某个按钮组件，就会产生ActionEvent事件；当复选按钮或单选按钮被选或取消选择时就会发生ItemEvent事件；当拖动滚动条时就发生AdiustmentEvent事件。这三个类别的事件分别用ActionEvent、ItemEvent和AdiustmentEvent类描述。

上面三个类别的监听器接口分别ActionListener接口、ItemListener接口和AdiustmentListener接口。这三个接口有一个共同的特点，就是每个接口只声明了一个成员方法。

ActionEvent　void actionPerformed(ActionEvent e)

ItemEven　void itemStateChanged(ItemEvent e)

AdiustmentEvent

void adjustmentValueChanged(AdjustmentEvent e)