

# JAVA编程高级

## —— GUI编程



# GUI编程

目标: **Java Swing**包, 什么是组件和容器的概念及相关类。布局管理器。**Swing** 中常用组件的使用。**Swing** 中的事件处理模型。

教学方法: 讲授ppt + 上机练习



# 本章要点

- **GUI** 概述
- **Swing** 容器和组件
- 布局管理器
- **GUI** 事件处理



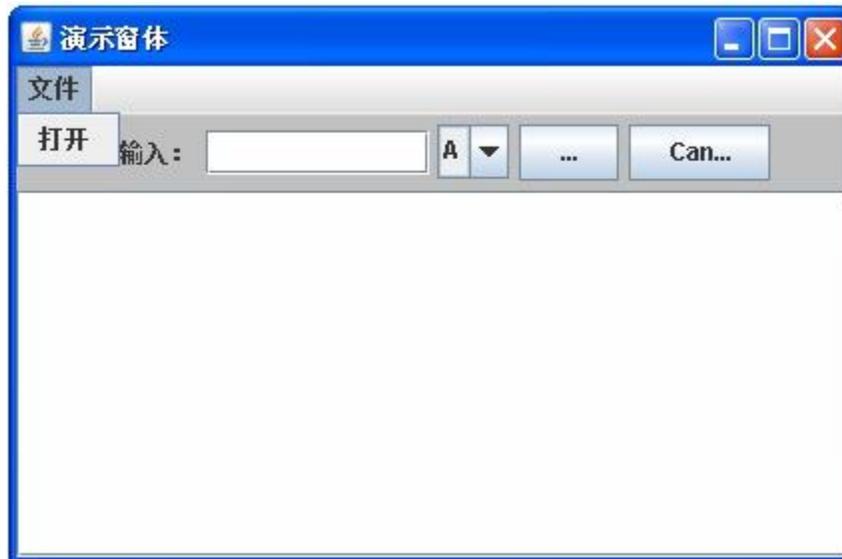
# Contents

- 1 GUI概述
- 2 Swing容器和组件
- 3 布局管理器
- 4 4 GUI事件处理



# GUI概述

- **GUI (Graphical User Interface)** 图形用户界面  
通过java应用程序提供给用户操作的图形界面，包括窗口、菜单栏、工具条、按钮等组件和其他各种屏幕元素



# GUI概述

- **Java GUI**

Java 提供了两个 GUI 的开发包：

java.awt

javax.swing

java.awt包

AWT是java GUI的早期版本，AWT中提供了基本的GUI设计工具，但组件种类有限，无法设计所需的所有功能。

java.awt 包中的抽象类 Component 是所有 Java GUI 组件的共同父类，它规定了所有 GUI 组件的基本特性。

javax.swing包

Swing是构筑在AWT上层的一组GUI组件集合，与AWT相比Swing提供了更完整的组件，引入了许多新的特性和能力。



# GUI概述

- GUI 组件

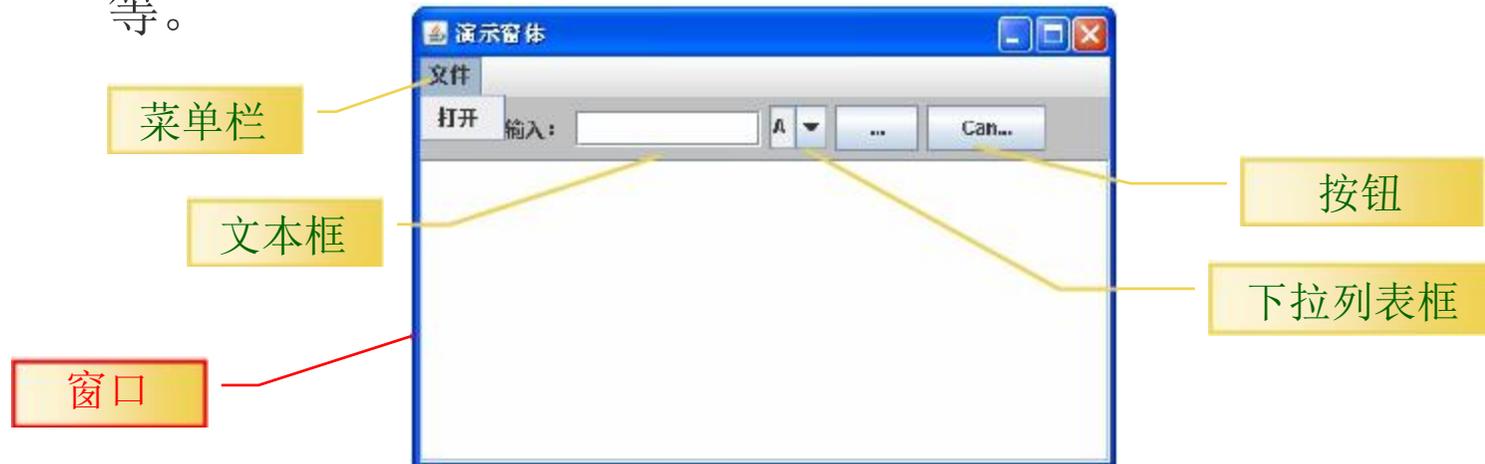
按其作用可分为两大类：

基本组件（简称组件、构件）

容器

GUI 组件：也称构件，其上不能容纳其他组件，如按钮、文本框等图形界面元素。

容器：是一种特殊的组件，用来容纳其他组件，如窗口、对话框等。



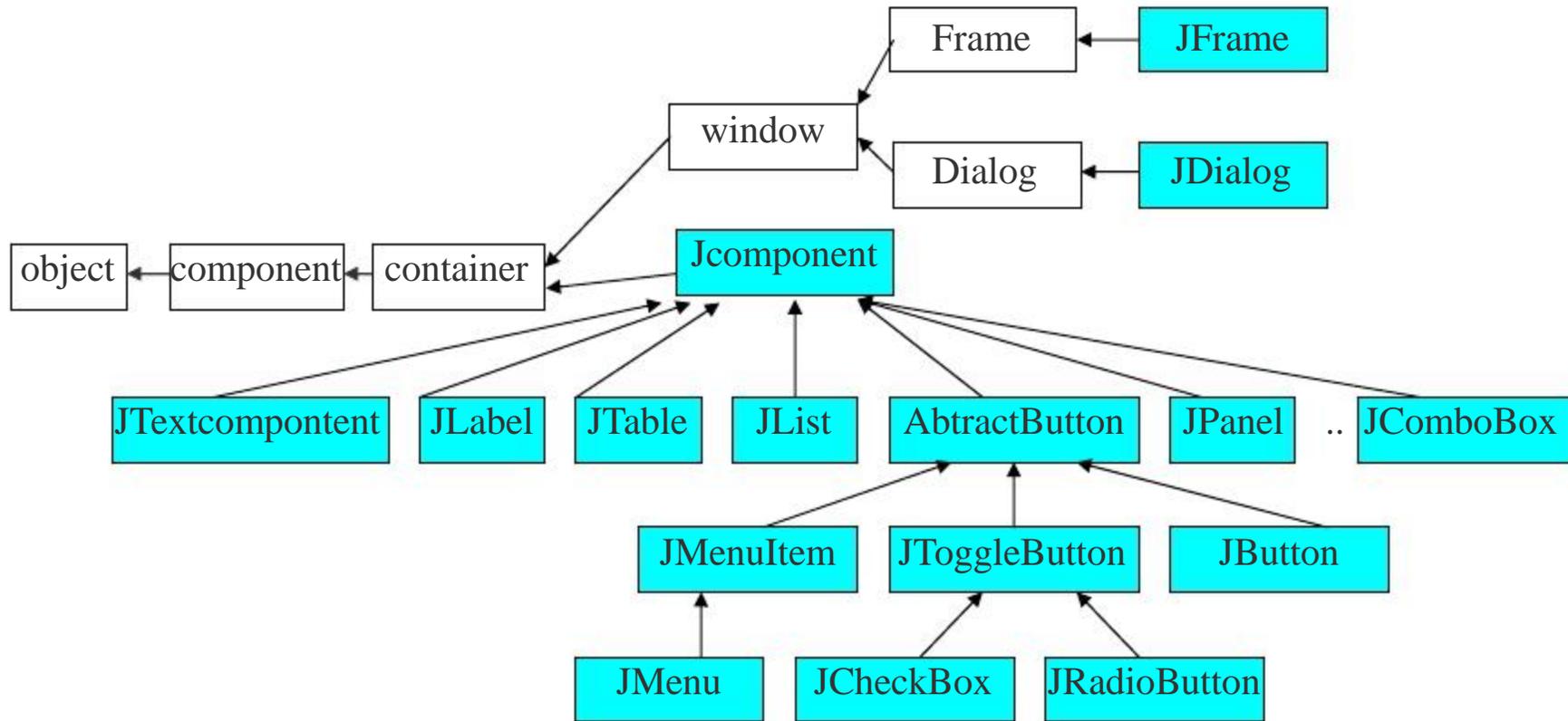
# Contents

- 1 GUI概述
- 2 **Swing**容器和组件
- 3 布局管理器
- 4 4 GUI事件处理



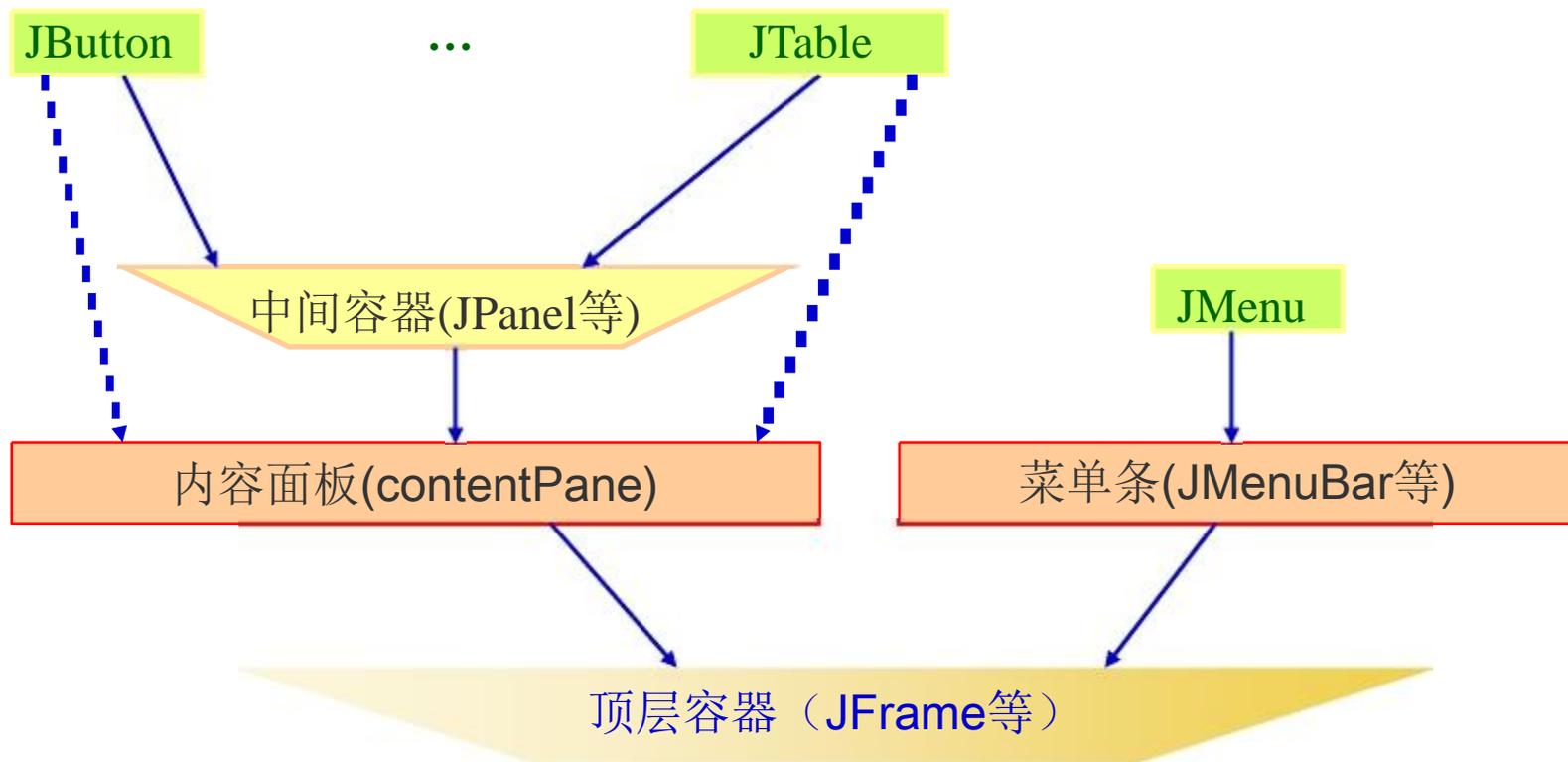
# Swing容器和组件

## •Swing组件继承关系图



# Swing容器和组件

- Swing容器及组件的使用



# Swing容器和组件

- 创建顶层容器（窗口） — 类**JFrame**的使用

可通过构造函数创建 `JFrame` 类对象，生成不可视的窗体组件。

构造一个初始时不可见的新窗体：

```
JFrame frame = new JFrame();
```

创建一个初始不可见的、具有指定标题 `title` 的新窗体：

```
JFrame frame = new JFrame(String title);
```

示例：**JFrameDemo.java**



# Swing容器和组件

- 创建顶层容器（窗口） — 类**JFrame**的使用

设置窗口体显示位置：

```
setLocation(int x,int y)
```

设置窗口体大小：

```
setSize(int width,int height)
```

设置窗口体是否可见：

```
setVisible(boolean b)
```

# Swing容器和组件

- 获得内容面板(**ContentPane**)      示例: **JFrameDemo.java**

可通过JFrame对象的getContentPane()方法获得内容面板。

获得窗口体上的内容面板:

```
JFrame frame = new JFrame(  
Container con = frame.getContentPane())
```

在内容面板上添加组件

```
add(Component comp);
```

```
add(String loc,Component comp);
```

# Swing容器和组件

- 创建基本组件

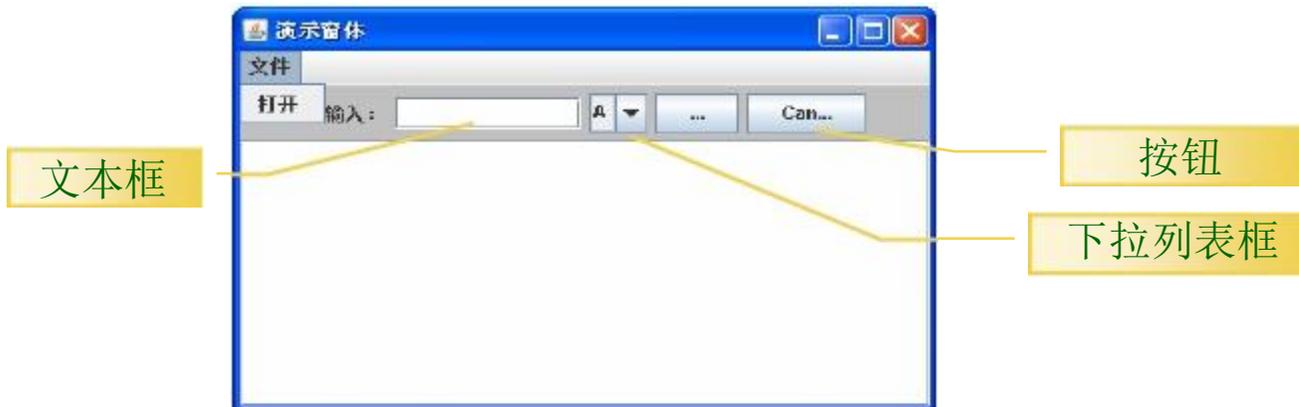
示例：**JFrameDemo.java**

基本组件分类：

文本输入组件：文本框(JTextField)、密码域(JPasswordField)、  
文本域(JTextArea)、标签(JLabel)

按钮组件：普通按钮(JButton)、复选框(JCheckBox)、单选按钮  
(JRadioButton)

下拉列表框(JComboBox)



# Swing容器和组件

- 创建基本组件 — 文本组件 示例: **JFrameDemo.java**

**标签(JLabel)**: 使用 JLabel 类可创建表示短文本字符串或图像或二者的显示标签组件对象。

创建文字或图像显示标签的构造方法:

**JLabel(String text)**

**JLabel(Icon image);**

Hi There!    Another Label

可指定文本水平对齐方式:

**JLabel(String text, int horizontalAlignment)**

其中的取值可以是SwingConstants中定义的LEFT、CENTER、RIGHT、LEADING 或 TRAILING 常量

# Swing容器和组件

- 创建基本组件 — 文本组件 示例: **JFrameDemo.java**

**文本框(JTextField)**: 使用 JTextField 类可创建表示单行文本框的组件对象。

构造一个具有指定列数的新的空 TextField:

```
JTextField(int columns)
```

构造一个用指定文本初始化的新 TextField:

```
JTextField(String text);
```



获取 / 设置单行文本:

```
public String getText(
```

```
public void setText(String
```

# Swing容器和组件

- 创建基本组件 — 文本组件      示例: **JFrameDemo.java**

**文本域(JTextArea)**: 使用 JTextArea 类可创建显示纯文本的多行区域的组件对象。

创建文字或图像显示标签的构造方法:

```
JTextArea(int rows, int columns)
```

获取、设置、追加多行文本:

```
public String getText()
```

```
public void setText(String str)
```

```
public void append(String str)
```

# Swing容器和组件

- 创建基本组件 — 按钮组件      示例: **JFrameDemo.java**

**普通按钮(JButton)**: 使用 JButton 类创建普通按钮对象, 生成按钮组件。

创建一个带文本 text 的按钮:

```
new JButton(String text)
```

创建一个带图标 icon 的按钮:

```
new JButton(Icon icon)
```



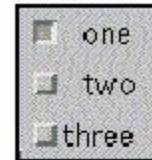
# Swing容器和组件

- 创建基本组件 — 复选框 示例: **JFrameDemo.java**

**复选框(JCheckBox)** : 使用 JCheckBox 类创建复选框对象, 可以让用户选择多个选项。

创建一个带文本 text的复选框:

```
new JCheckBox(String text)
```



创建一个带文本, 并有初始选定状态的复选框:

```
new JCheckBox (String text,boolean selected)
```

获取复选框文本的内容

```
getText()
```

# Swing容器和组件

- 创建基本组件 — 单选按钮 示例: **JFrameDemo.java**

**单选按钮(JRadioButton)** : 使用 JRadioButton 类创建单选按钮组件, 但用户每次只能选中一个单选按钮。

创建一个带文本 text的复选框:

```
new JRadioButton(String text)
```

创建一个带文本, 并有初始选定状态的复选框:

```
new JRadioButton(String text,boolean selected)
```



获取复选框文本的内容

```
getText()
```

# Swing容器和组件

- 创建基本组件 — 单选按钮 示例: **JFrameDemo.java**

## 单选按钮(JRadioButton)

将多个单选按钮放入一个单选按钮组:

```
ButtonGroup g = new ButtonGroup()
```

将单选按钮添加到同一单选按钮组:

```
add(AbstractButton b)
```



# Swing容器和组件

- 创建基本组件 — 下拉列表框 示例: **JFrameDemo.java**

**下拉列表框(JComboBox)** : 使用 JComboBox 类可以创建下拉列表对象。

创建一个没有选项的下拉列表:

```
new JComboBox()
```

创建含有指定数组元素的下拉列表:

```
new JComboBox(Object[ ] items)
```

向下拉列表中添加选项/获得所选项

```
void addItem(Object obj)
```

```
Object getSelectedItem()
```



# Swing容器和组件

- 中间容器类 — **JPanel**      示例: **JFrameDemo.java**

JPanel 类是最简单的容器类, 可容纳各类基本组件, 包括其它面板

JPanel 类作为面板组件可添加至窗体

```
jframe.getContentPane().add(JPanel p)
```

```
jframe.setContentPane(JPanel p)
```

创建一个流式布局的JPanel

```
new JPanel(
```

创建一个含有指定布局管理器的JPanel

```
new JPanel(LayoutManager layout)
```

# Swing容器和组件

- 中间容器类 — **JScrollPane**(滚动面板)

使用 JScrollPane 类可创建带滚动条的中间容器对象

创建一个空的JScrollPane

```
new JScrollPane()
```

创建一个显示指定组件内容的 JScrollPane

```
new JScrollPane(Component view)
```

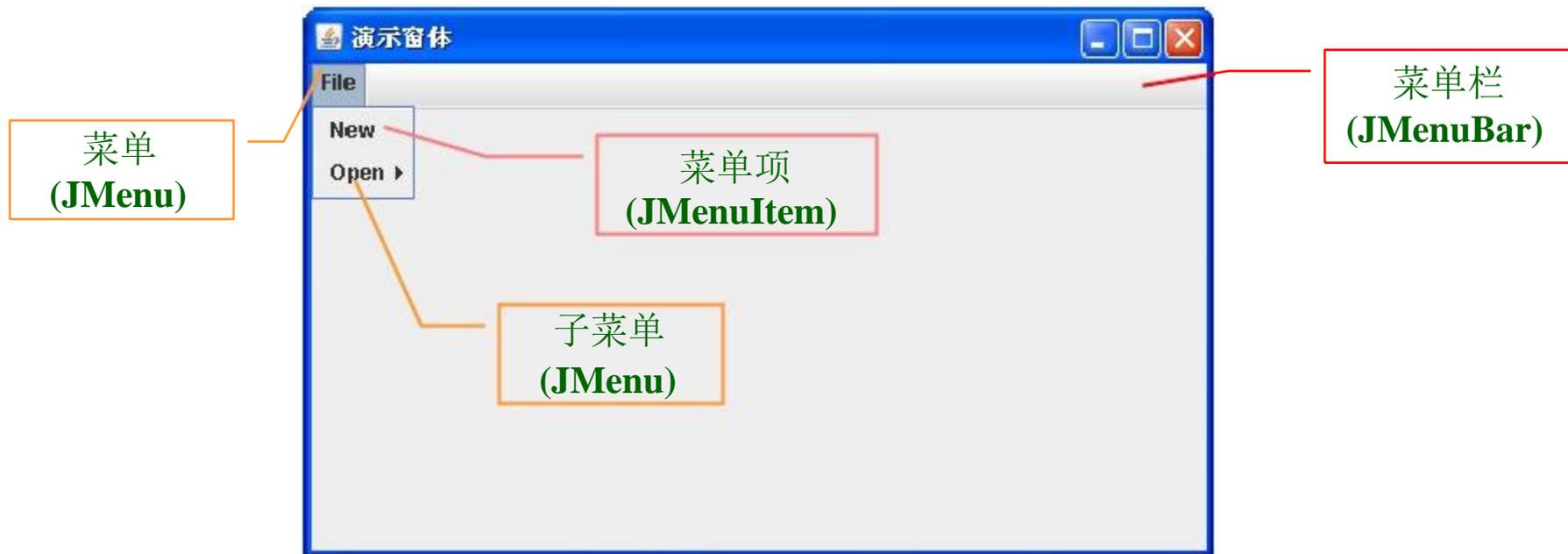
示例: **JFrameDemo.java**



# Swing容器和组件

- 常用组件 — 菜单(**JMenuBar**)      示例: **JFrameDemo.java**

在JFrame中, 有一个菜单栏(**JMenuBar**)用来建立菜单



# Swing容器和组件

- 常用组件 — 菜单(**JMenuBar**) 示例: **JFrameDemo.java**

在Swing中建立菜单一般需要3步:

建立一个菜单栏(**JMenuBar**), 并添加到窗口体(JFrame)中

```
new JMenuBar()
```

```
setJMenuBar(JMenuBar menubar)
```

建立菜单(**JMenu**), 加到菜单栏(JMenuBar)中

```
new JMenu(String name)
```

```
add(JMenu menu)
```

建立菜单项(**JMenuItem**)或子菜单(JMenu), 加到菜单中(JMenu)

```
new JMenuItem(String itemname)
```

# Contents

- 1 GUI概述
- 2 Swing容器和组件
- 3 布局管理器
- 4 4 GUI事件处理



# 布局管理器

- 布局管理器简介

Java 提供了布局管理器来管理组件在容器中的布局，java.awt 包中常用的布局管理器类有：

**BorderLayout**—边界布局

**FlowLayout**—流式布局

**CardLayout**—卡片布局

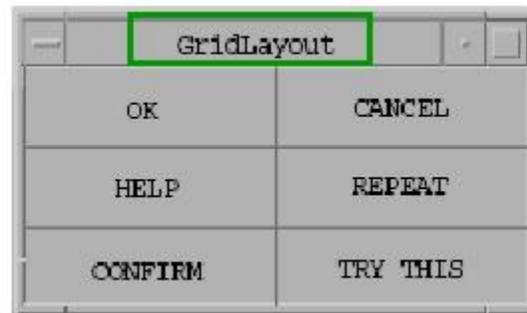
**GridLayout**—网格布局

**GridBagLayout**—网袋布局



# 布局管理器

- 布局管理器简介



# 布局管理器

- BorderLayout (边界布局管理器)

BorderLayout 将容器划分为东、南、西、北、中五个区域，可以将组件分放在这五个区域中。

原则上，BorderLayout 允许最多放置五个组件，而可借助面板类 Panel 添加更多的组件。

BorderLayout 是窗口体 (JFrame) 的默认布局



# 布局管理器

- **FlowLayout**（流式布局管理器）

FlowLayout 把容器中的组件从左到右，从上到下依次排列。

如果某容器采用 FlowLayout 布局，那么其add方法中指定的位置参数将被忽略。

当容器窗口大小改变时，组件的位置可能会发生变化，但组件尺寸不变。

FlowLayout 是面板（**JPanel**）的默认布局



# 布局管理器

- 设置容器的布局管理器

修改窗口体（顶层容器，JFrame）的布局管理器

```
void setLayout(LayoutManager manager)
```

修改中间容器（JPanel）的布局管理

```
new JPanel(LayoutManager manager)
```

```
void setLayout(LayoutManager manager)
```

取消布局管理器/设置各组件的大小和位置

```
setLayout(null)
```

```
setBounds(int x,int y,int width,int height)
```

# Contents

- 1 GUI概述
- 2 Swing容器和组件
- 3 布局管理器
- 4 4 GUI事件处理



# GUI事件处理

- 事件处理机制

在Java的GUI程序中，事件用于描述程序、系统或者使用者的活动，是图形系统的最基本的功能之一。

在一个图形界面的程序中，用户点击一个按钮，这时系统便会创建一个事件，并且把该事件交给相应的程序去处理。



# GUI事件处理

- 事件处理模型

**事件源**：发生事件的组件。

**事件**：用户对组件的操作。

**事件处理程序**：负责处理事件的方法

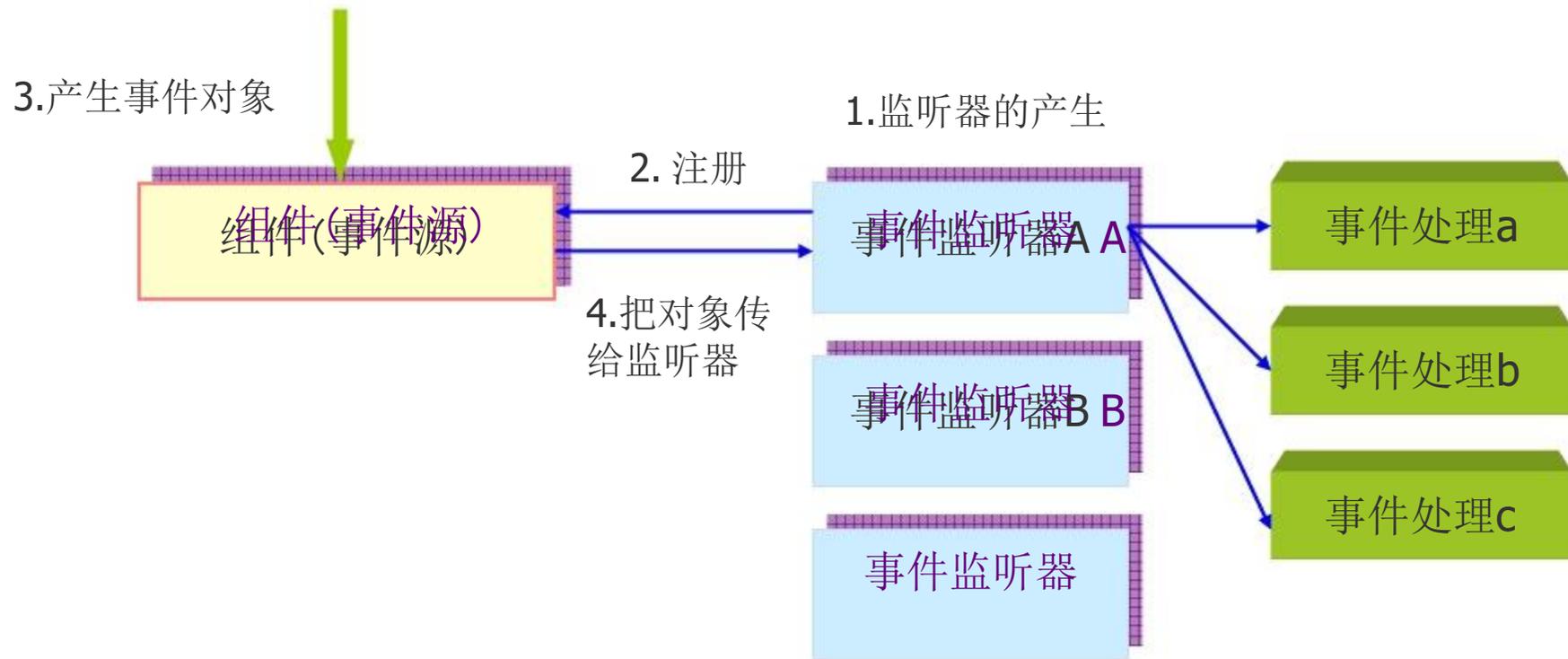
**事件监听器**：存放事件处理器方法的类对象。

**事件监听器注册**：将一个事件监听器对象同某个事件源的某种事件进行关联的过程。



# GUI事件处理

- 事件处理模型



# GUI事件处理

- 事件处理5步骤

示例：**JFrameDemo.java**

- 1、监听器的产生。实现每个监听接口的类就可以作为监听器来使用。
- 2、把监听器注册到组件上。采用addXXXListener的方式。如：`button.addActionListener(监听器)`。
- 3、事件的产生。如按下按钮，就产生该事件的一个对象。
- 4、系统把产生出来的事件对象自动返回给已经注册过的监听器。
- 5、由该监听器来指派相应的方法来处理事件。这个一般由我们开发者来编写处理代码。



# GUI事件处理

- 常见的事件及其事件监听器

事件类	事件监听器	事件源(组件)	产生事件的时机
ActionEvent	ActionListener	Button	按钮按下时
		List	双击List中的项目时
		MenuItem	选取菜单中的某项时
		TextField	按下Enter时
AdjustmentEvent	AdjustmentListener	Scrollbar	滚动滚动条时
ItemEvent	ItemListener	Checkbox	选中某个选项时
		CheckboxMenuItem	勾选菜单中某个选项时
		Choice	选取下拉菜单中某个选项时
		List	选取List某个选项时
TextEvent	TextListener	TextArea	文本内容改变时
		TextField	文本内容改变时



# GUI事件处理

- 事件监听器及其接口方法

事件分类	接口名	方法
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
TextEvent	TextListener	textValueChanged(TextEvent)



# GUI事件处理

- 事件适配器

为了简化编程，JDK针对大多数事件监听接口定义了相应的实现类，我们称之为事件适配器(XXXAdapter)类。

在事件适配器中实现了相应监听器接口中的所有方法，但不做任何操作。

一个类只要继承适配器类，如果要对某类事件的某种情况进行处理，只要覆盖相应的方法就可以了，其他方法就再也不用“简单实现”了。

示例：**JFrameDemo.java**



# GUI事件处理

- 事件监听器的匿名内部类实现方式

如果一个事件监听器类只用于在一个组件上注册监听器事件对象，为了让程序代码更为紧凑，可以用匿名内部类的语法来产生这个事件监听器对象

示例 **TestEvent.java**



# 本章小结

- **GUI**简介
- **Swing**容器和组件
- 布局管理器
- **GUI**事件处理



谢谢

