Java程序设计

第10章 图形用户界面

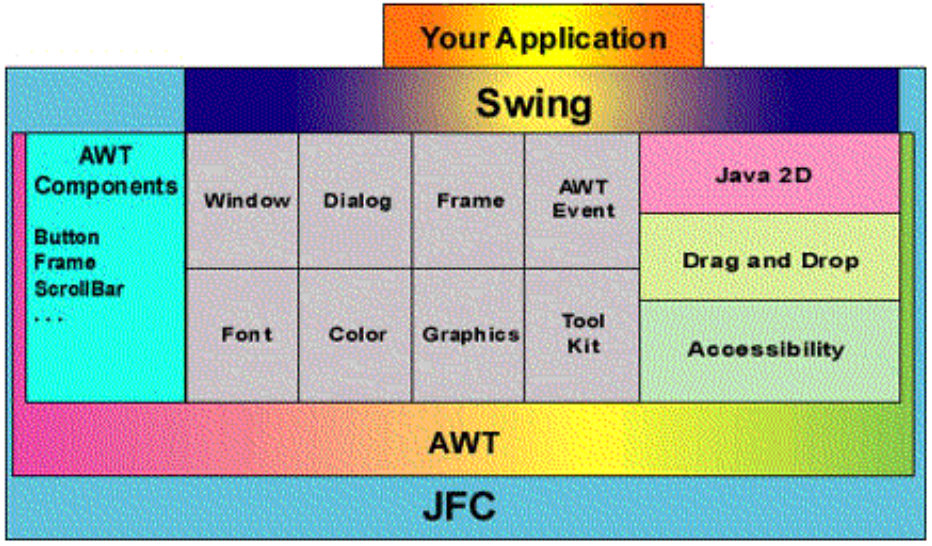
第10章 图形用户界面

10.1 组件

组件

图形用户界面

- 图形用户界面(graphical user interface，GUI)
- AWT和swing
  - AWT： abstract window toolkit(抽象窗口工具集)
  - Swing 是JDK1.2以后版本所引入的。功能更强，界面更丰富。
  - 各种平台上更统一，它是轻量级的 (lightweight，即all-Java language) 。

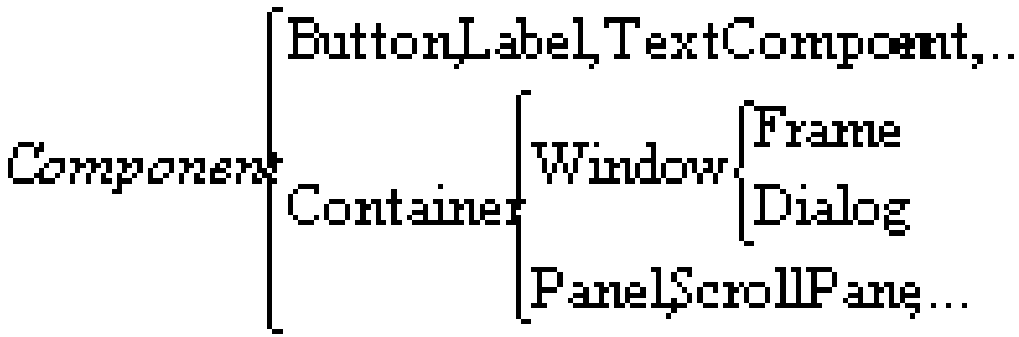


主要的包

- java.awt包
  - Frame Button Label TextField TextArea Panel
- javax.swing包
  - JFrame JButton JLabel JTextField JTextArea JPanel

AWT组件分类

- Java中构成图形用户界面的各种元素，称为组件（Component）。
- 组件分为容器（Container）类和非容器类组件两大类
- 容器又分为顶层容器和非顶层容器两大类



Component类

- Component类是所有组件和容器的抽象父类，其中定义了一些每个容器和组件都可能用到的方法
  - getBounds() , getSize(), getLocation(), getHeight(), getWidth()
  - setVisible(), setEnabled(),
  - setBackground(), setForeground()
  - getGraphics()
  - requestFocus()

Swing顶层容器

- 顶层（top-level）容器
  - JFrame, JDialog, JApplet
- 容器都有add(子组件)

```
java.lang.Object
    java.awt.Component
        java.awt.Container
            java.awt.Window
                java.awt.Frame
                    javax.swing.JFrame
```

```
java.lang.Object
    java.awt.Component
        java.awt.Container
            java.awt.Panel
                java.applet.Applet
                    javax.swing.JApplet
```

Swing组件

- Swing组件 JComponent
  - JComponent 是非顶层容器
  - 也都是容器(Container)，都有 add(子组件)
  - 有很多方法，如 setToolTipText

```
java.lang.Object
    java.awt.Component
        java.awt.Container
            javax.swing.JComponent
                javax.swing.AbstractButton
                    javax.swing.JButton
```

10.2 实现图形用户界面的三步曲

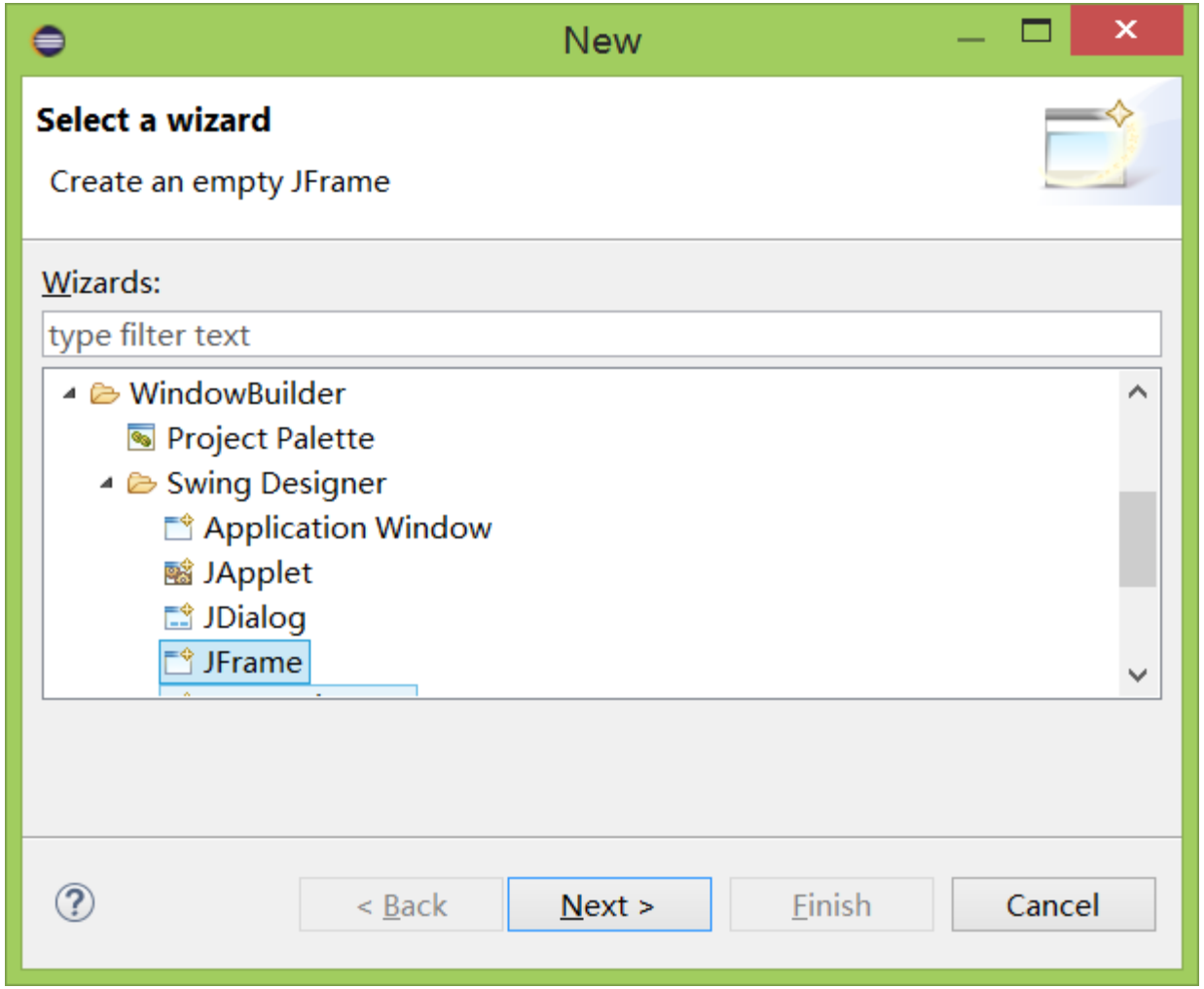
实现图形用户界面的三步曲

实现界面的三步曲

- 设计和实现图形用户界面的工作主要有以下几点。
- （1）创建组件（Component）
  - 创建组成界面的各种元素，如按钮、文本框等。
- （2）指定布局（Layout）
  - 根据具体需要排列它们的位置关系。
- （3）响应事件（Event）
  - 定义图形用户界面的事件和各界面元素对不同事件的响应， 从而实现图形用户界面与用户的交互功能。

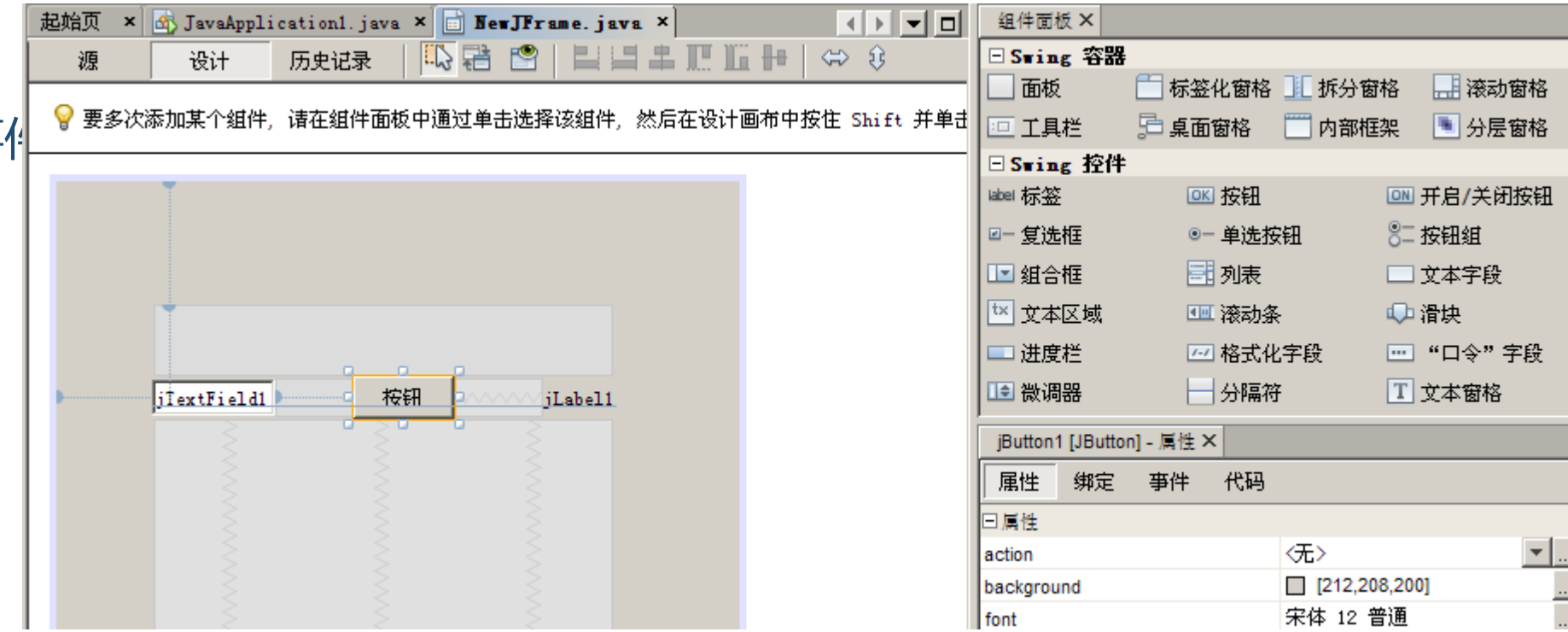
IDE中的窗体设计

- Eclipse中
  - 项目上点右键
    - New—Other—Windows Builder
    - —Swing Designer—Jframe
    - 其顶部可切换source/design模式
  - 在窗体上右键，Layout
    - (absolute表示绝对布局)
  - 加上按钮等组件
    - 设置其属性
  - 添加事件
    - 组件上右键，Add New Event Handler



可视化界面设计

- Netbeans类似
  - 新建—Other—Swing GUI窗体
  - 布局可设为null
- 直接生成组件、布局、事件



示例代码

- 示例 使用JFrame TestJFrame.java
  - 要点：
    - 设置为可关闭
    - setDefaultCloseOperation(EXIT_ON_CLOSE);
  - 注： 对于JFrame、JDialog、JApplet
    - 其中有一个ContentPane的容器
    - getContentPane().add(lbl); 可直接写为 add(lbl);
    - getContentPane().setLayout(…); 可直接写为 setLayout (…);

示例

- 示例 使用按钮 JButtonDemo.java
  - 要点：
    - getContentPane().setLayout(new FlowLayout());
    - b1.addActionListener(al);

窗体设计器生成的代码

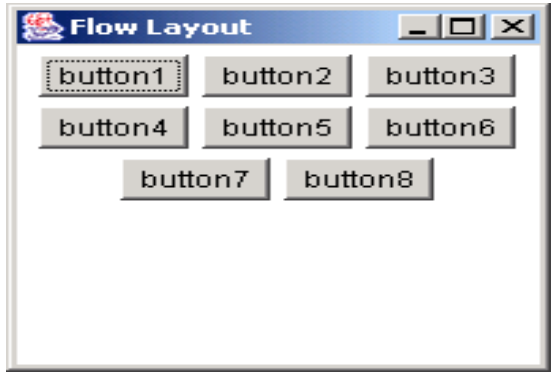
- 示例 Eclipse自动生成的代码
- 示例 Netbeans自动生成的代码

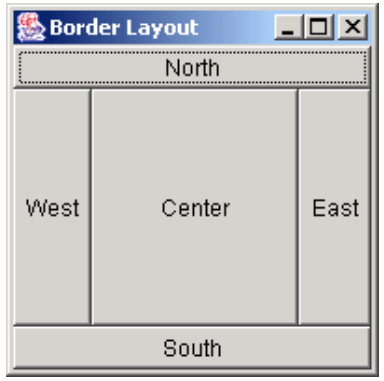
10.3 布局管理

布局管理

五种布局管理器

- Java.awt包中共定义了多种布局管理器
  - 常用的三种：FlowLayout　　BorderLayout　　GridLayout

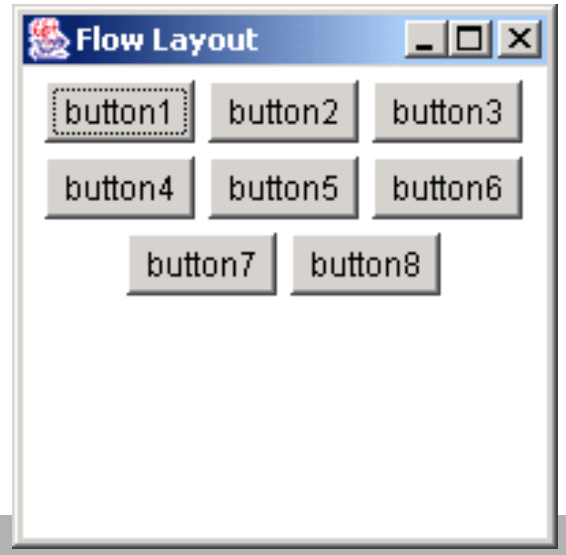


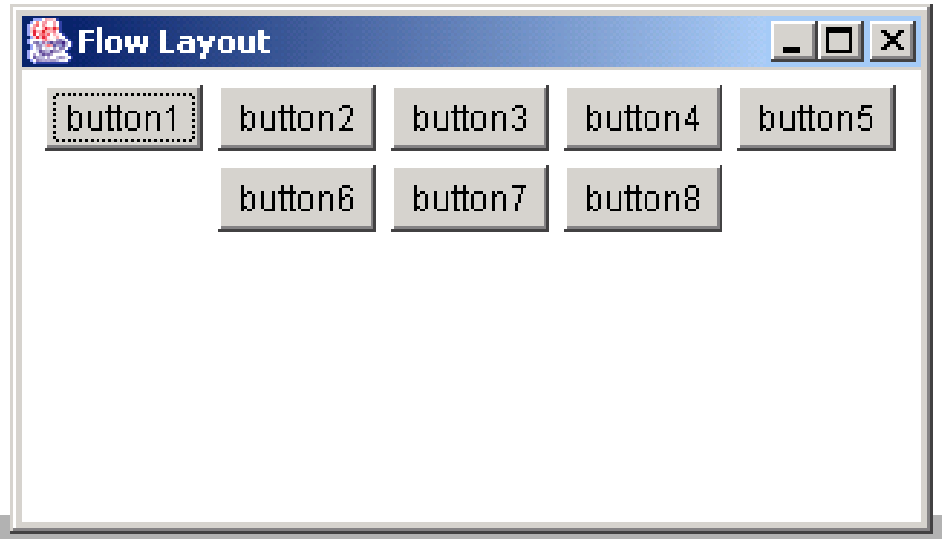


  - 还有CardLayout, GridBagLayout等

FlowLayout布局管理器

- FlowLayout布局
  - 对组件逐行定位，行内从左到右，一行排满后换行
  - 默认对齐方式为居中对齐
  - 不改变组件的大小，按组件原有尺寸显示组件
  - FlowLayout是Panel类的默认布局管理器
  - TestFlowLayout.java



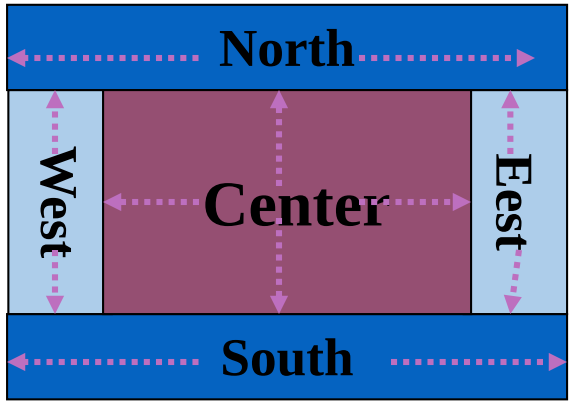


FlowLayout 的构造方法

- 可在构造方法中设置不同的组件间距、行距及对齐方式
  - new FlowLayout(FlowLayout.RIGHT,20,40);
  - 右对齐，组件之间水平间距20个像素，竖直间距40个像素；
  - new FlowLayout(FlowLayout.LEFT);
  - 左对齐，水平和竖直间距为缺省值：5；
  - new FlowLayout();
  - 使用缺省的居中对齐方式，水平和竖直间距为缺省值：5；

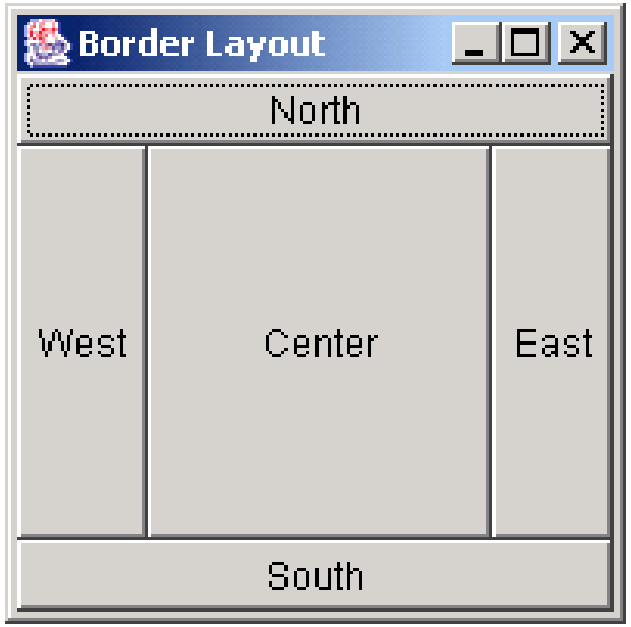
BorderLayout 布局管理器

- BorderLayout将整个容器的布局划分成东、西、南、北、中五个区域，组件只能被添加到指定的区域
- 如不指定组件的加入部位，则默认加入到Center区域
- 每个区域只能加入一个组件，如加入多个，则先前加入的组件会被遗弃
- BorderLayout是Frame类的默认布局管理器



BorderLayout举例

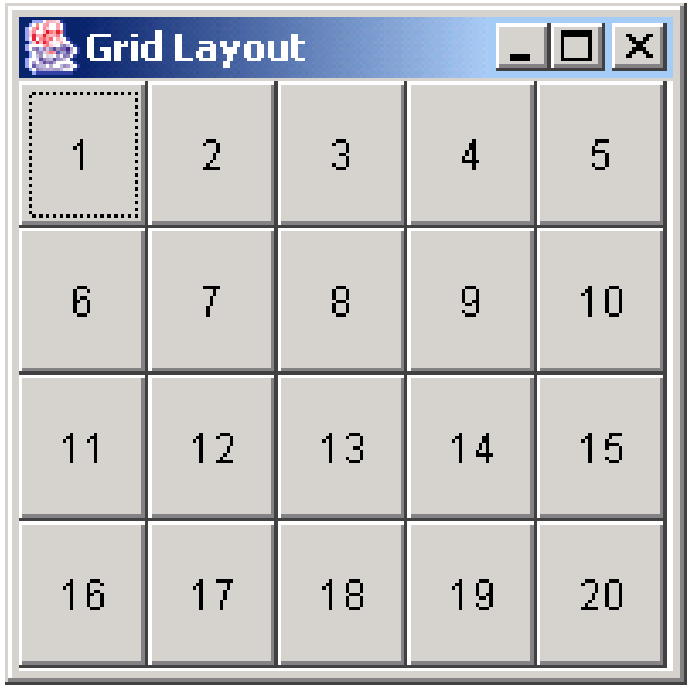
范例：**TestBorderLayout.java**



GridLayout 布局管理器

- GridLayout型布局管理器将布局划分成规则的矩形网格，每个单元格区域大小相等.
- 组件被添加到每个单元格中，先从左到右添满一行后换行，再从上到下.
- 在GridLayout构造方法中指定分割的行数和列数.
- new GridLayout(4,5);

- 范例：TestGridLayout.java



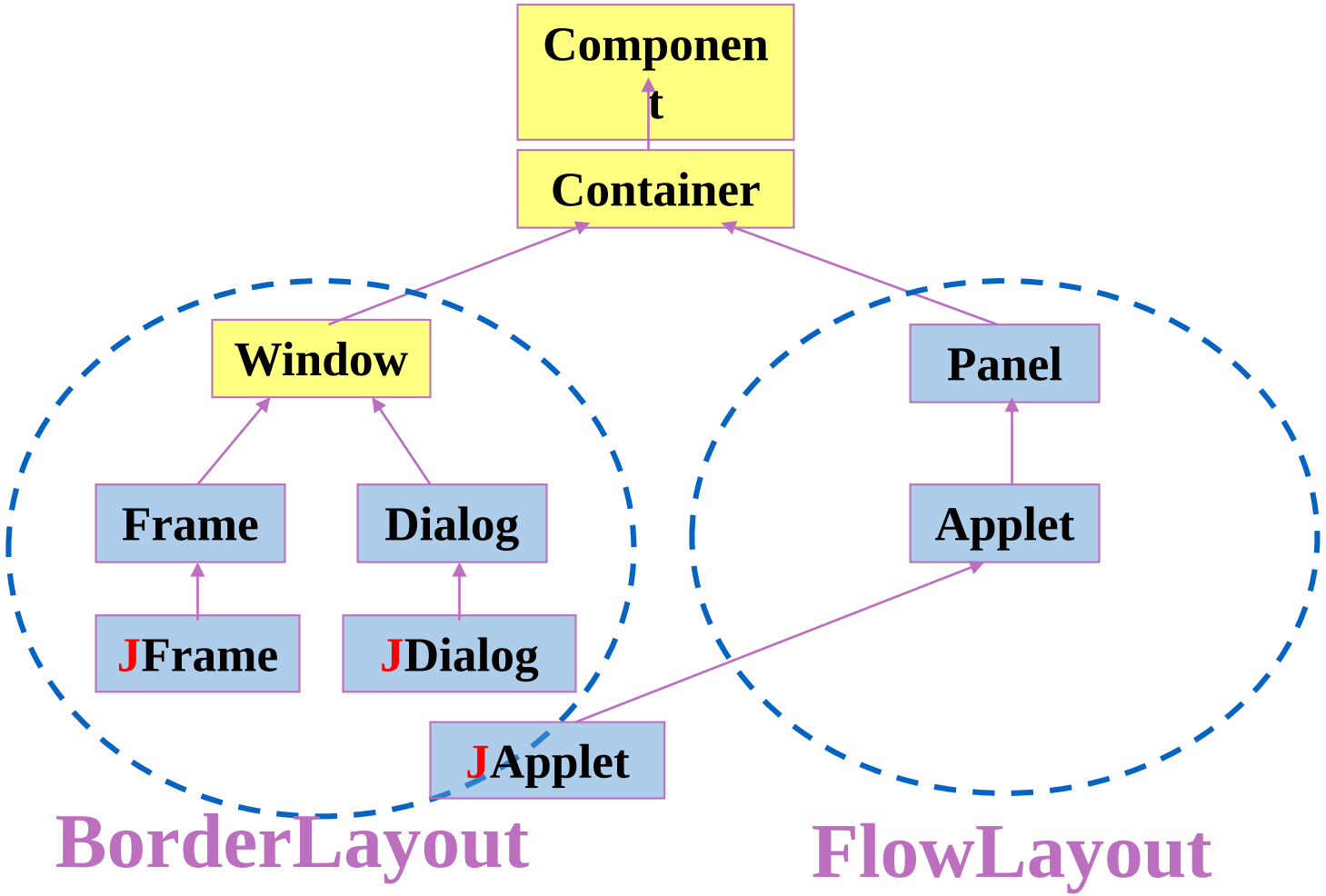
* CardLayout 布局管理器

- CardLayout布局管理器能够帮助用户处理两个以至更多的成员共享同一显示空间，就好象一叠卡片摞在一起。
- 注意：在一张卡片中只能显示一个组件，因此可以使用容器嵌套方法显示多个组件。
- addLayoutComponent, first, next, last方法

* GridBagLayout 布局管理器

- AWT中最灵活、最复杂的布局管理器，各组件所占空间可以不相同且灵活规定，参见参考书及API文档；

默认的布局管理器

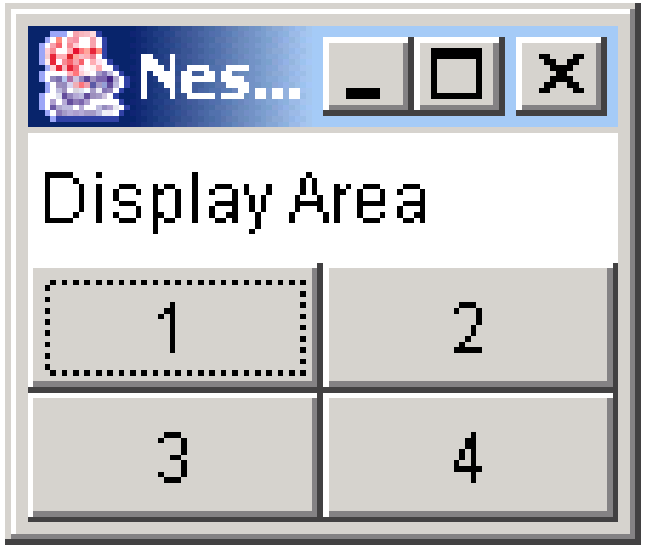

Component
Container
Window
Panel
Frame
Dialog
Applet
JFrame
JDialog
JApplet
BorderLayout
FlowLayout

绝对布局

- 绝对布局的一般流程：
  - 对象.setLocation(100,20);
  - 对象.setSize(80,20);（或用setBounds同时设定四个参数）
  - 容器.setLayout(null);
  - 容器.add(对象);
  - 示例： TestLayoutNull.java
- 绝对布局的蔽端

容器的嵌套使用举例

范例：**NestedContainer.java**



10.4 事件处理

事件处理

事件

- 事件及事件监听器
  - 事件（Event）
    - 鼠标、键盘、布局改变等等操作等
  - 事件监听器（Event Listener)
    - 对这些事件作出响应的程序

简单示例

- 示例　TestActionEvent.java

```
ActionListener al = new MyListener();
btn.addActionListener(al);

class MyListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
                System.out.println("a button has been pressed");
        }
}
```

事件监听器

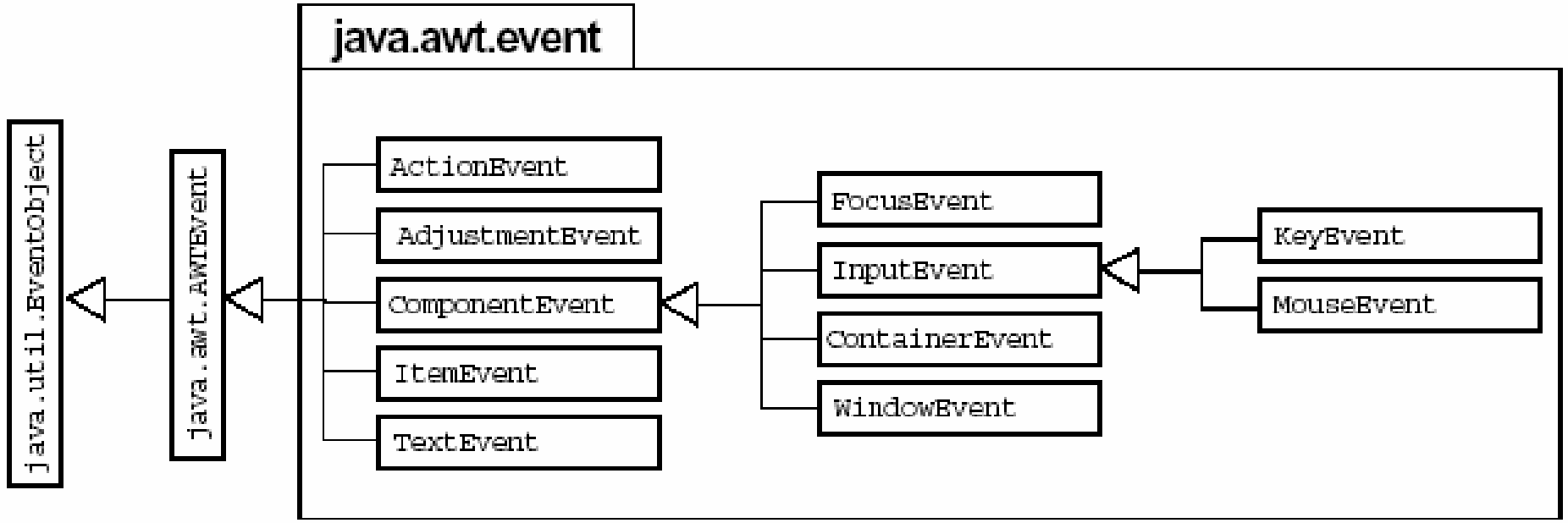
- 事件监听器是一些事件的接口
  - 是 AWTEventListener的子接口
  - 接口中含有相关的方法
    - 如：MouseMotionListener 是对鼠标移动事件的处理的接口，它含有两个重要的方法：
    - void mouseDragged (MouseEvent e)；// 处理鼠标拖动的方法
    - void mouseMoved (MouseEvent e)；// 处理鼠标移动的方法
- 这些方法带一个事件对象作为参数
  - 事件参数是AWTEvent的子类
    - 如MouseMotionListener的两个方法都带MouseEvent参数。
    - 程序中可以根据这个参数可以得到有关事件的详细信息。

Event包含的信息

- 事件类中包含有事件相关的信息，最重要的有：
- （1）事件源（即产生事件的组件）
  - getSource()
  - 得到的Object可以强制类型转换成相应的类型
- （2）事件的具体情况
  - 如MouseEvent 的 getX(), getY() 方法得到鼠标的坐标
  - KeyEvent 的 getKeyChar() 得到当前的字符等。

事件分类

- 包：java.awt.event 及 javax.swing.event



常用的事件类

- 对所有组件能用的事件
  - component listener
  - focus listener
  - key listener
  - mouse listener
  - mouse-motion listener
  - mouse-wheel listener
  - 。。。
- 对特定组件能用的事件
  - Action Listener
  - Caret Listener
  - Change Listener
  - Item Listener
  - List Selection Listener
  - Window Listener
  - 。。。

事件适配器

- 事件适配器类（Adapter）----简化实现Listener
  - 如WindowListener有7个方法，即使一些方法不做任何事情，也得书写全。
  - 在适配器类中，实现了相应监听器接口中所有的方法，但不做任何事情。
  - 在extends事件适配器时，只需要Override自己所需要的方法即可

常见的事件适配器：

- （1）ComponentAdapter (组件适配器)；
- （2）ContainerAdapter (容器适配器)；
- （3）FocusAdapter (焦点适配器)；
- （4）KeyAdapter (键盘适配器)；
- （5）MouseAdapter (鼠标适配器)；
- （6）MouseMotionAdapter (鼠标运动适配器)；
- （7）WindowAdapter (窗口适配器)。

注册事件监听器

- 事件的注册 addxxxxListener
- 监听器的实现有两种方法
  - implements xxxListener
    - 具体写其中的每个方法
  - extends xxxAdapter
    - 其中Adapter是Listener的默认实现，每个方法的方法体为空
    - Adapter可以只Override其中重要的方法

实现Listener的方法

- 实现时，可以
  - 单独写个类 (实现接口或继承Adapter)
  - 或者写在相前的类中（ implements ）
  - 或者匿名类
  - 或者用Lambda表达式（ Java8以上 ）

使用事件的例子

```
b.addActionListener( new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e){
                ((JButton)e.getSource()).setText(""+new java.util.Date());
        }
});
```

使用匿名类的例子

```
f.addMouseMotionListener(new MouseMotionListener(){
        public void mouseDragged(MouseEvent e) {
                String s = "位置 ( " + e.getX() + "," + e.getY() + ")";
                tf.setText(s);
        }
        public void mouseMoved(MouseEvent e) { }
});
f.addWindowListener( new WindowAdapter() {
        public void windowClosing( WindowEvent e ){
                System.exit(0);
        }
});
```

- 其中用匿名类实现了MouseMotionListener及继承了WindowAdapter，同时实例化了这个匿名类的对象
- 示例：TestAnonymous.java

使用Lambda表达式的例子

- 针对只有一个方法的接口，可以用Lambda表达式

```
b.addActionListener( e->{
        ((JButton)e.getSource()).setText(""+new java.util.Date());
} );
```

多个监听器

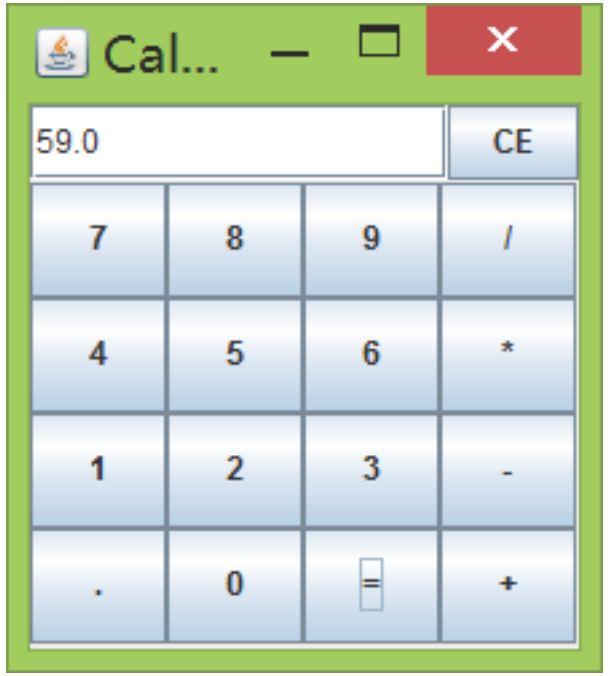
- 一个对象上可注册多个监听器
  - 示例 TestMultiListener.java
- 多个对象可注册同一个监听器
  - 用事件参数的 getSource() 可以得到是哪一个对象

事件与线程

- 线程中，如果要更新界面，要放到the event dispatching thread
- 也就是要调用 SwingUtilities.invokeLater()方法
- 示例：InvokeLaterDemo.java

应用示例

- 小小计算器　JCalculator.java



应用示例

- 排块游戏 BlockMoveGame.java

- 要点
  - 按钮的数组
  - 按钮的生成、加入、事件
  - 函数的书写
  - 注释的书写

10.5 常用组件的使用

常用组件的使用

常用组件

- 标签、按钮 与动作事件
  - JLabel Jbutton ActionListener
- 文本框、文本区域 与文本事件
  - JTextField JTextArea JPasswordField JFormattedTextField Text:istener
- 单、复选按钮，列表 与选择事件
  - JRadioButton JCheckbox JList JComboBox ItemListener
- 滚动条与调整事件
  - JScrollBar JScrollPane AdjustmentListener
- 画布与鼠标、键盘事件
  - Canvas JComponent KeyListener MouseListener
- Panel与容器事件
  - JPanel ComponentListener

强功能的组件

- JTextPane
  - 可以编辑文本、网页、RTF
- JScrollPane 能自动滚动
  - new JScrollPane(组件)
- 画图
  - 可以Override 其void paint(Graphics g)

窗口、对话框

- JFrame　JDialog
  - 其中的RootPane 及 ContentPane
- WindowListener
  - 窗口能关闭
    - setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

标准对话框

- JOptionPane的4种方法
  - showConfirmDialog --- 确认对话框，提出问题，然后由用户自己来确认（按"Yes"或"No"按钮）
  - showInputDialog --- 提示输入文本
  - showMessageDialog --- 显示信息
  - showOptionDialog - - 组合其它三个对话框类型
- JFileChooser
  - 文件选择器
- JColorChooser
  - 颜色选择器

菜单及工具栏

- 菜单
  - JMenuBar JMenu JMenuItem JPopupMenu
  - frame.setMenuBar(...)
- 工具栏
  - JToolbar
  - 其中可加入按钮

小技巧

- 焦点
  - 对象.requestFocus();
- 处理按键

```
textMaxScores.addKeyListener(new KeyAdapter (){
   public void keyPressed(  KeyEvent e){
            if(e.getKeyCode()==KeyEvent.VK_LEFT){.....}
    }
});
```

- 线程中操作界面
  - SwingUtilites.invokeLater(...)

通过官网学习

- 多查看API文档
- 官网上的教程及示例
  - http://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html#TextSamplerDemo

应用示例

- TextEditorApp2.java
- 要点：
  - 组件、布局、事件
  - 窗体、对话框
  - 菜单、工具栏
  - 文件
  - 分层
  - 日志

10.6 Applet

Applet

Applet

- Applet(小程序)
  - 在Internet的浏览器中运行的Java程序
- java.Applet.Applet 类
  - 是 java.awt.Panel 的子类
  - 默认布局是 FlowLayout
- javax.swing.Japplet 类
  - 是java.Applet.Applet的子类
  - 默认布局是 BorderLayout

Applet程序

- HelloWorldApplet.java
  - import表示导入
  - extends JApplet表示继承
    - Applet或JApplet都可以
  - 有paint()方法，表示如何绘制
  - 没有main()方法
- HelloWorldApplet.html

```
import java.awt.*;
import java.applet.*;
import javax.swing.*;
public class HelloWorldApplet extends JApplet {
    public void paint(Graphics g){
        g.drawString ("Hello World!",20,20);
    }
}
```

```
<HTML>
<HEAD><TITLE> An Applet </TITLE></HEAD>
<BODY>
<applet code="HelloWorldApplet.class"
    width=200 height=40 background=white>
</applet>
</BODY>
</HTML>
```

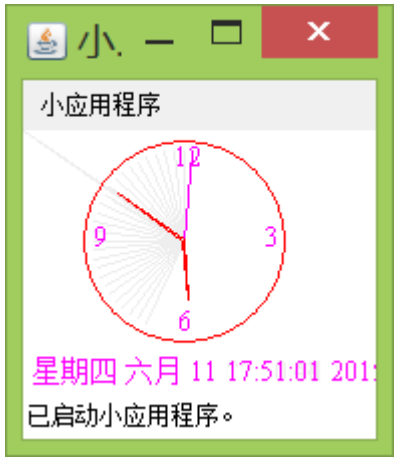
编写Applet类

- Applet类的主要方法
  - init()
  - start()
  - paint()
  - stop()
  - destroy()
- 一般需要Override其中几个
- 另外，可以用getParameter()方法得到环境参数

示例

- 示例
  - Clock2.java Clock2.htm

- 可以用appletViewer运行Applet
  - appletViewer xxxxxx.html



浏览器运行Applet

- 浏览applet
  - 将applet编译后的class及html放到服务器上
  - 然后用浏览器浏览 html

- 在浏览之前，先要
  - 下载java运行环境
    - http://www.javac.com
    - (注意这是jre，不是jdk)
  - 配置java环境
    - 控制面板—Java—安全—级别“中”
    - Java8不支持本地文件打开html中的applet参考 http://blog.csdn.net/fhx007/article/details/3389585

Applet用得较少

- Applet需要下载、配置运行环境
- Applet的安全性处理比较麻烦
- 现在有很多替代方法
  - Flash
  - Html5 + javascript
  - 其他