图形用户界面

主要内容

图形用户界面概述

事件处理

容器和一般组件

布局管理器

图形界面的组成

有哪些图形界面呢 ？

早期计算机用户界面 ？

现在的图形用户界面 ？

建立图形用户界面的思路

面板

按钮

菜单

滚动条

……

建立图形用户界面的方法

创建一个顶级的容器组件

向容器添加组件

设计事件处理程序（难点和核心）

显示图形用户界面

图形用户界面概述

图形用户界面(Graphics User Interface)简称GUI，是用图形的方式，借助菜单、按钮等标准界面元素和鼠标操作，帮助用户方便地向计算机系统发出指令、启动操作，并将系统运行的结果同样以图形方式显示给用户的技术。

抽象窗口工具集-AWT

设计Java图形用户界面的基本元素由抽象窗口工具集-AWT提供，主要包括用户界面组件、事件处理模型、图形和图像工具、布局管理器等，它们都在java.awt包中。

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)。Java程序要显示的GUI组件都是抽象类java.awt.Component或java.awt.MenuComponent的子类。MenuComponent是与菜单有关的组件。

Swing组件分类

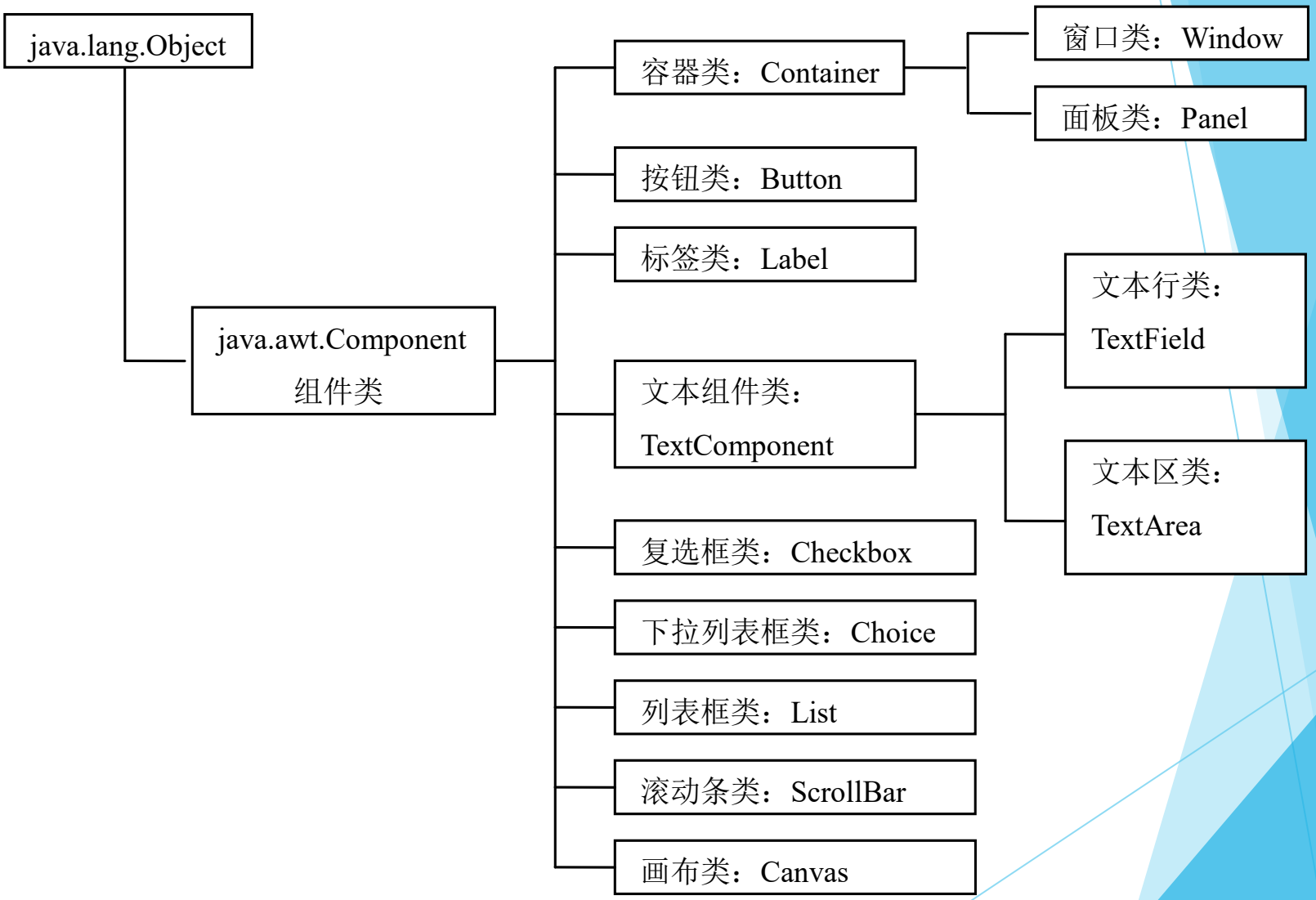
Swing是一个为Java设计的GUI工具包，属于Java基础类的一部分。Swing包括了图形用户界面（GUI）功能，其组件包含：文本框、文本域、按钮、表格、列表等。
Swing提供许多比AWT更好的屏幕显示元素。它们用纯Java写成，所以同Java本身一样可以跨平台运行，这一点不像AWT。它们是JFC的一部分。它们支持可更换的面板和主题（各种操作系统默认的特有主题），然而不是真的使用原生平台提供的设备，而是仅仅在表面上模仿它们。这意味着你可以在任意平台上使用Java支持的任意面板。轻量级组件的缺点则是执行速度较慢，优点就是可以在所有平台上采用统一的行

组件分为容器(Container)类组件和非容器类组件两大类。容器类组件本身也是组件，但容器中可以包含其他组件，也可以包含其他容器；非容器类组件是不能再包含其他组件的组件，其种类较多，如按钮(Button)、标签(Label)、文本类组件(TextComponent)等。

容器又分为两种：顶层容器和非顶层容器。

顶层容器是可以独立的窗口，顶层容器的类是Window，Window的重要子类是Frame和Dialog；

非顶层容器，不是独立的窗口，它们必须位于窗口之内，非顶层容器包括Panel及ScrollPane等。


java.lang.Object
java.awt.Component
组件类
容器类：Container
窗口类：Window
面板类：Panel
按钮类：Button
标签类：Label
文本组件类：
TextComponent
文本行类：
TextField
文本区类：
TextArea
复选框类：Checkbox
下拉列表框类：Choice
列表框类：List
滚动条类：ScrollBar
画布类：Canvas

一、容器类

容器类Container是组件类Component的子类，其特点是容器中可以容纳其他组件。使用add()方法可以将其他组件加入到容器中，加入到容器中后，

1. 窗口与面板

容器类Container有两个主要子类：窗口类Window和面板类Panel。

Window类是可以自由移动的、不依赖其他容器而存在的窗口。Window类有两个主要组件：框架Frame和对话框Dialog。Frame是一种带标题栏并且可以改变大小的窗口，而Dialog则是一种带有标题栏但不能改变其大小的窗口；

Panel类与窗口类似，但它是一种没有标题的容器，且不能独立存在，必须包含在另外一个容器之中。

java.awt

Class Frame

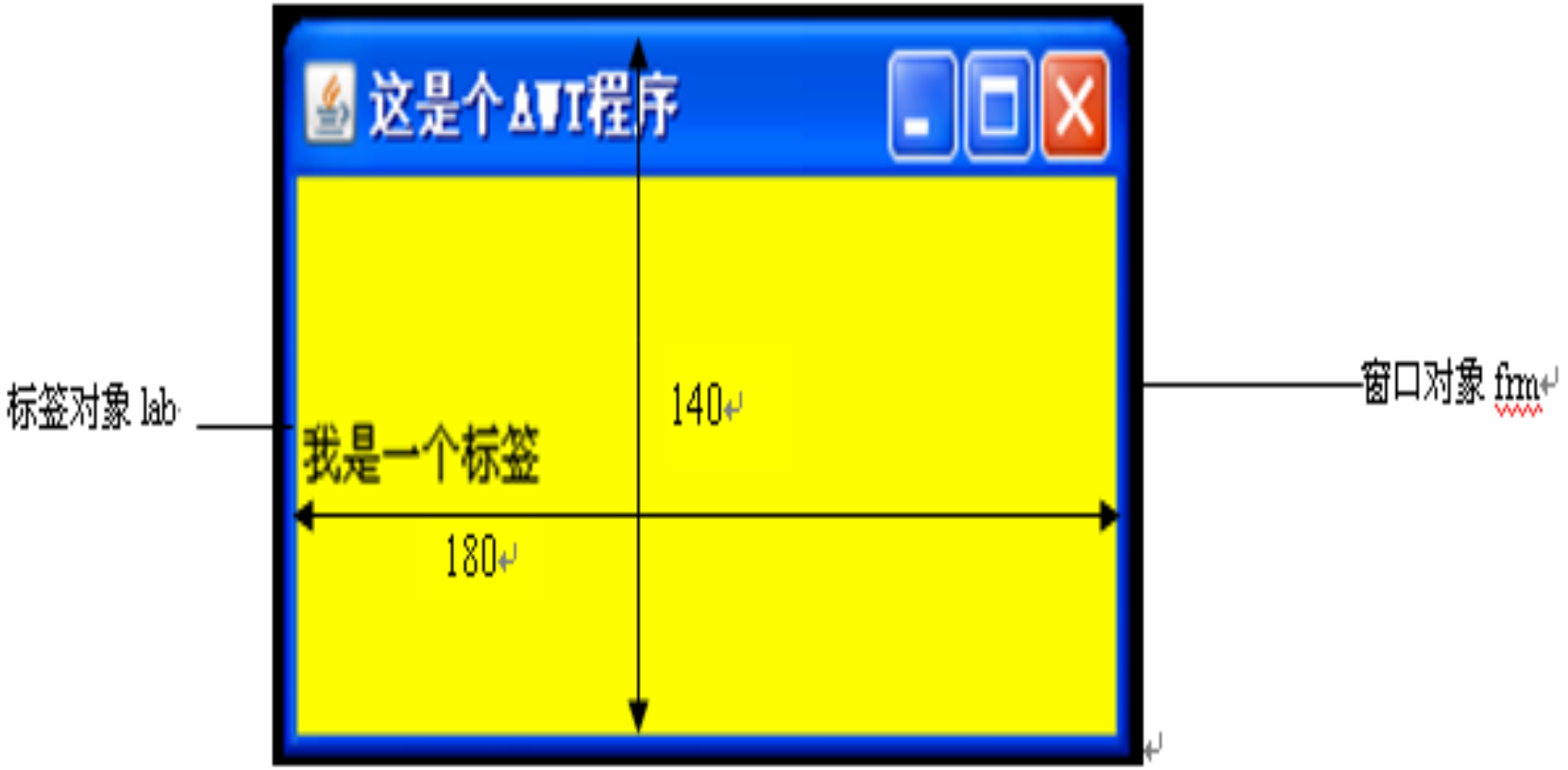
java.lang.Object
- java.awt.Component
  - java.awt.Container
    - java.awt.Window
      - java.awt.Frame

2、框架类Frame的应用

Frame类的许多方法是从它的父类Window或更上层的类Container和Component继承过来的。

【例10.1】框架窗口的创建。

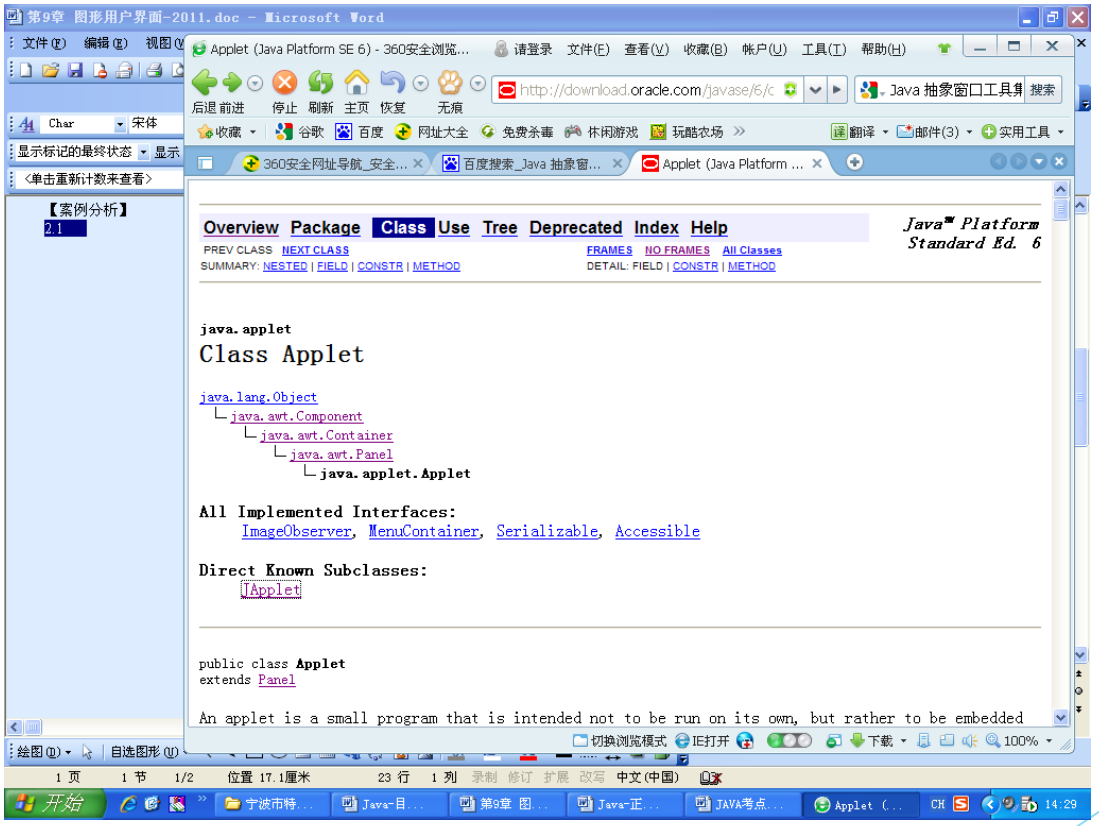
```
import java.awt.*;       //加载java.awt类库里的所有类
public class  FrameDemo
{
  static Frame frm=new Frame(“这是个AWT程序”);
  public static void main(String args[])
  {
    Label lab=new Label("我是一个标签");     //创建一个标签对象lab
    frm.setSize(180,140);              //设置框架大小
    frm.setBackground(Color.yellow);   //设置框架背景颜色
    frm.setLocation(250,150);          //设置窗口的位置
    frm.add(lab);                      //将标签对象lab加入窗口中
    frm.setVisible(true);              //将窗口显示出来
  }
}
```


这是个AWT程序
标签对象 lab
我是一个标签
140
180
窗口对象 frm

3、面板类Panel的应用

面板类Panel是容器类的直接子类，是一种没有标题的容器，并且实例化后必须使用Container类的add()方法装入到窗口对象中。

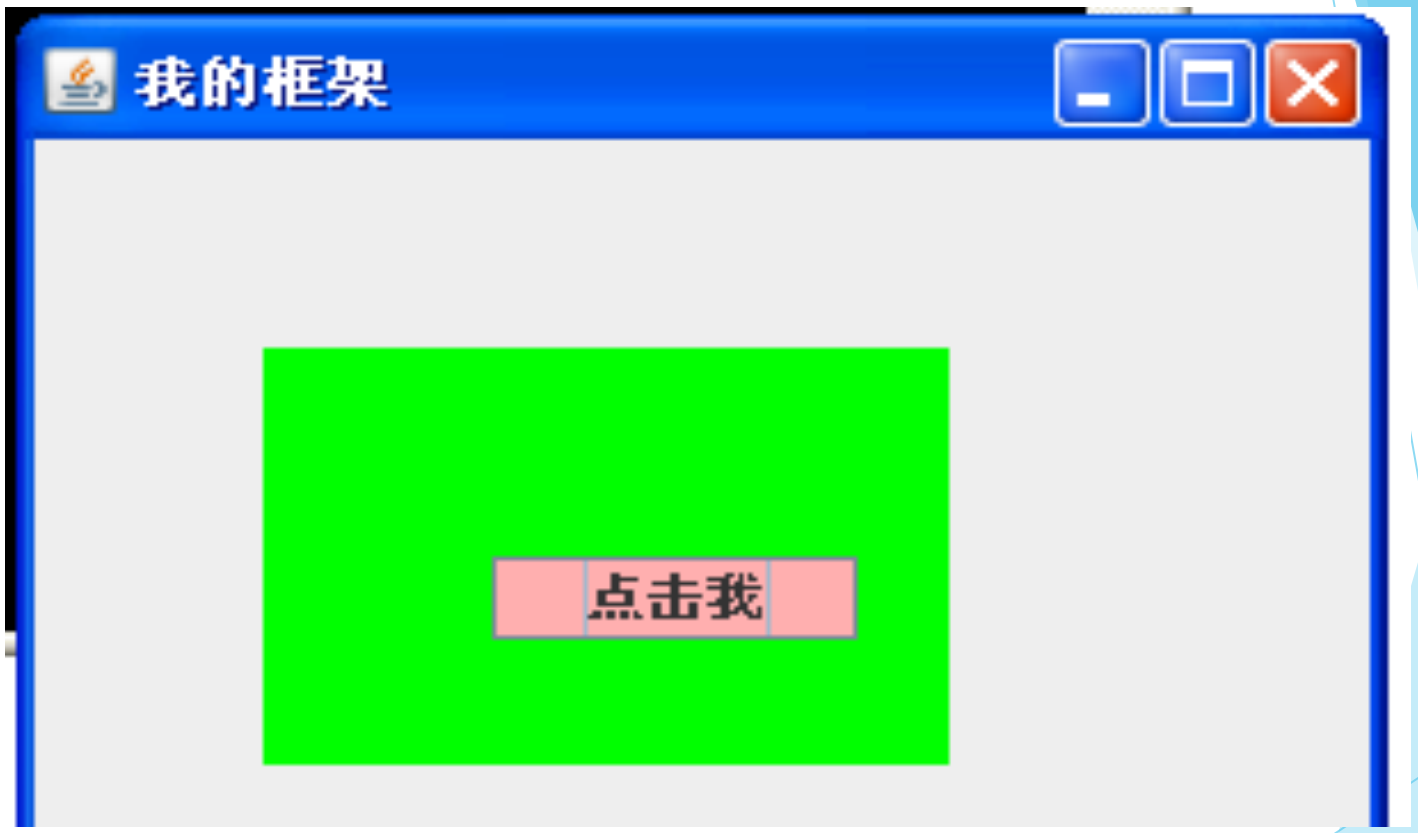
在Java应用程序中，一般独立应用程序主要使用框架Frame做容器，在Frame上通过放置Panel面板来控制图形界面的布局；如果应用到浏览器中，主要使用Panel的子类Applet来做容器。



【例10.2】在框架窗口中加入panel。

```
import java.awt.*;    //加载java.awt类库里的所有类
public class  PanelDemo
{
  public static void main(String args[])
  {
    JFrame frm=new Frame(“我的框架”);
    frm.setSize(300,200);
    frm.setLocation(500,400);
    frm.setBackground(Color.lightGray);
    JPanel pan=new Panel();
    pan.setSize(150,100);
    pan.setLocation(50,50);
    pan.setBackground(Color.green);
```

```
    JButton bun=new Button(“点击我”);
    bun.setSize(80,20);
    bun.setLocation(50,50);
    bun.setBackground(Color.pink);
    frm.setLayout(null);     //取消frm的默认布局管理器
    pan.setLayout(null);     //取消pan的默认布局管理器
    pan.add(bun);          //将命令按钮加入到面板中
    frm.add(pan);          //将面板加入到窗口中
    frm.setVisible(true);
  }
}
```


我的框架
点击我

二、非容器类

非Container类组件，又称为控制组件，简称控件。与容器不同，它里面不能再包含其他组件。

常用的控件有以下几种。

命令类：按钮Button

选择类：单选按钮CheckboxGroup、复选框Checkbox、列表框List、

下拉列表框Choice

文字处理类：文本框TextField、文本区域TextArea

三、图形用户界面设计的步骤

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)：根据具体需要排列界面上各元素的位置关系。

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。

事件处理的基本概念

- 1. 事件(event)
- 在图形用户界面中，事件是指用户使用鼠标或键盘对窗口中的组件进行交互时所发生的事情。
- 如单击按钮、输入文字或单击鼠标等。事件用于描述发生了什么事情。对事件做出响应的程序，称为事件处理程序（event handler）。

事件处理的基本概念

- **2. 事件源(Event Source)**
- 所谓事件源就是能够产生事件的对象。如按钮、鼠标、文本框、键盘等。
- **3. 事件监听者(Listener)**
- 事件监听者是一个对事件源进行监视的对象，当事件源上发生事件时，事件监听者能够监听到，并调用相应的方法对发生的事件做出相应的处理。
- Java语言的事件监听者的典型特征就是实现了监听接口，这些接口都是继承自java.util.EventListener接口。

事件处理的基本概念

- **4. 事件处理接口**

  Java语言的java.awt.event及javax.swing.event包中包含了许多用来处理事件的类和接口。为了处理事件源发生的事件，监听者会自动调用相应的方法来处理事件。那么调用什么方法呢？

  Java语言规定：为了让监听者能对时间源发生的时间进行处理，创建该监听者对象的类必须实现相应的接口，即必须在类体中定义该接口中所有抽象方法的方法体，以供监听者自动调用被类实现的某个接口的方法。

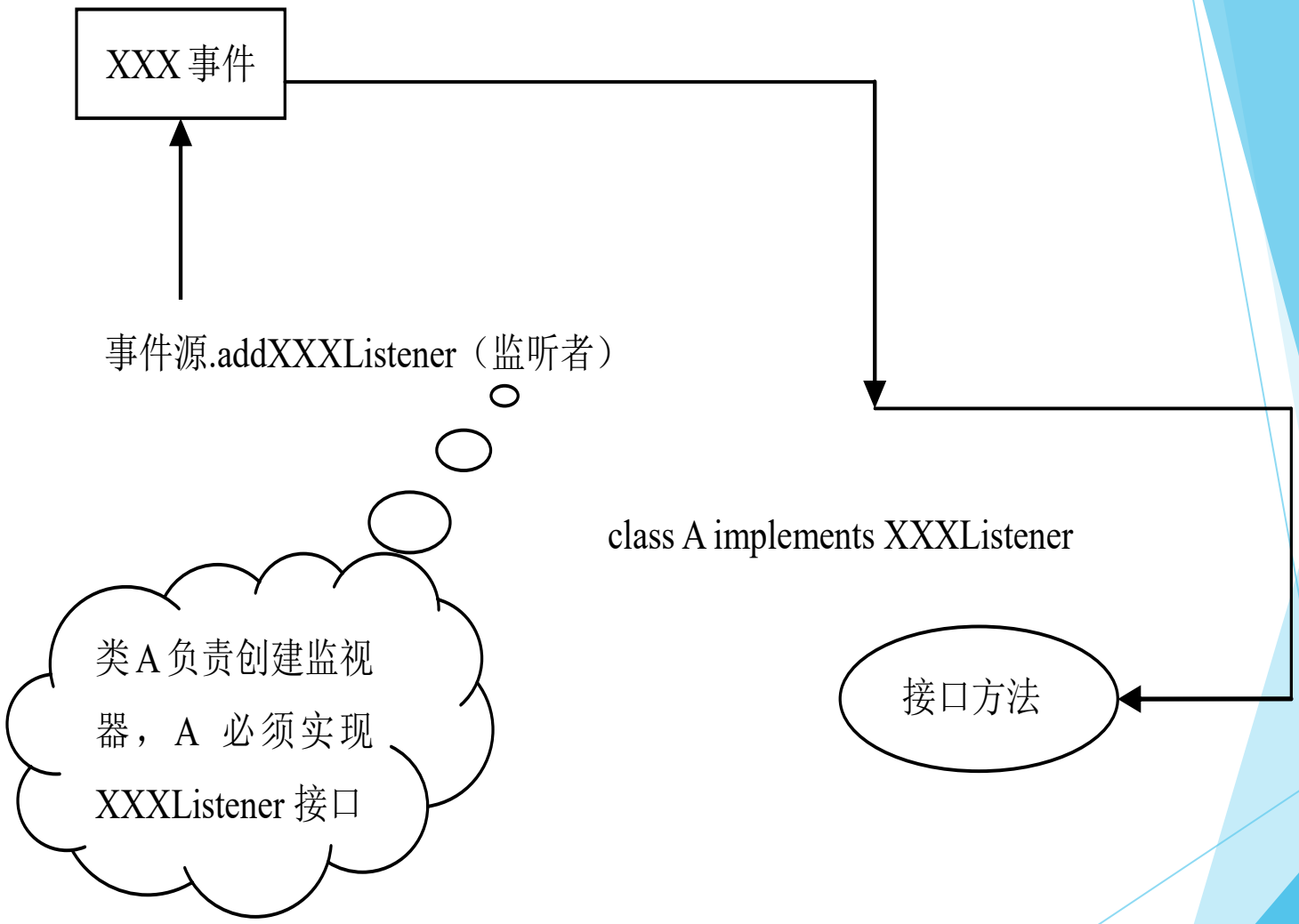
  简言之，处理事件的事件处理方法是定义在接口中的。

- *addWindowListener:*
- ***public void addWindowListener(WindowListener l)***
  - *Adds the specified window listener to receive window events from this window. If l is null, no exception is thrown and no action is performed.*
  - *Parameters:*
    - *l - the window listener*

事件处理机制

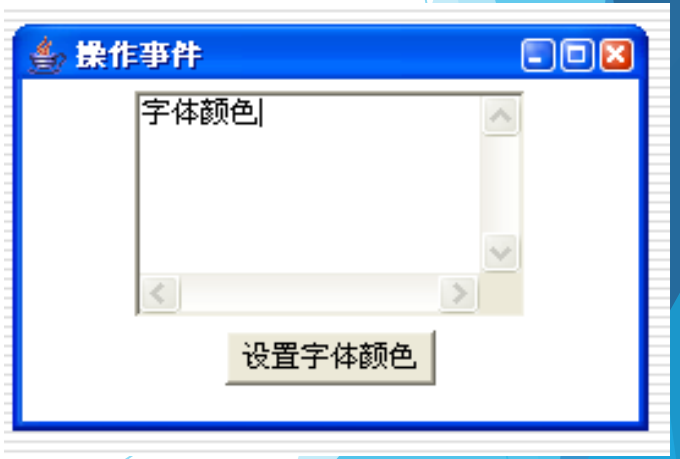
- Java的常用事件类教材表10-8所示。

事件委托处理模型由产生事件的事件源、封装事件相关信息的事件对象和事件监听者构成。例如，当按钮被鼠标单击时，会触发一个操作事件（ActionEvent），Java系统会产生一个事件对象来表示这个事件，然后把该事件对象传递给事件监听者，由监听者指定相关的接口方法进行处理。为了能接收到事件对象的信息，事件监听者事先要向事件源进行注册（register）。


XXX 事件
事件源.addXXXListener（监听者）
类 A 负责创建监视器，A 必须实现 XXXListener 接口
class A implements XXXListener
接口方法

一个静态的程序

```
import java.awt.*;
public class ComponentApp extends Frame
{
   static ComponentApp frm=new ComponentApp();
   static Button bt=new Button("设置字体颜色");
   static TextArea ta=new TextArea("字体颜色",5,20);
   public static void main(String[] args)
   {
      frm.setTitle("操作事件");
      frm.setLayout(new FlowLayout());
      frm.setSize(260,170);
      frm.add(ta);
      frm.add(bt);
      frm.setVisible(true);
    }
}
```

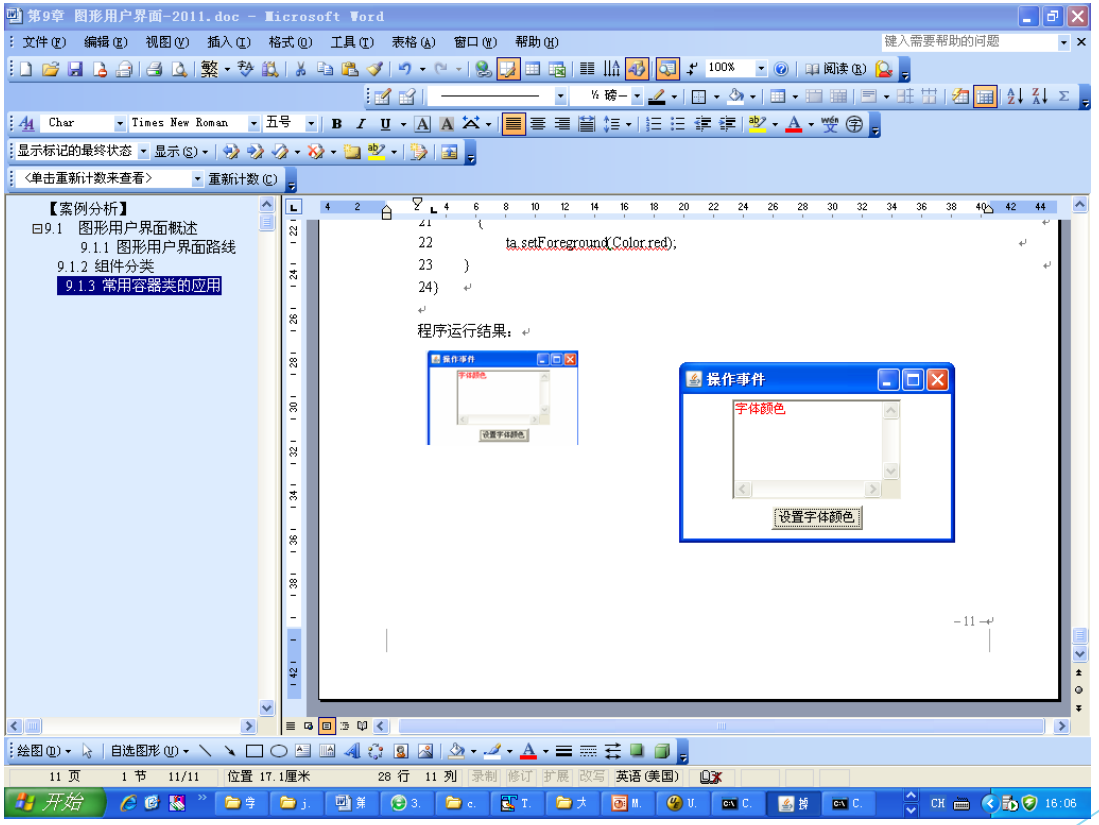


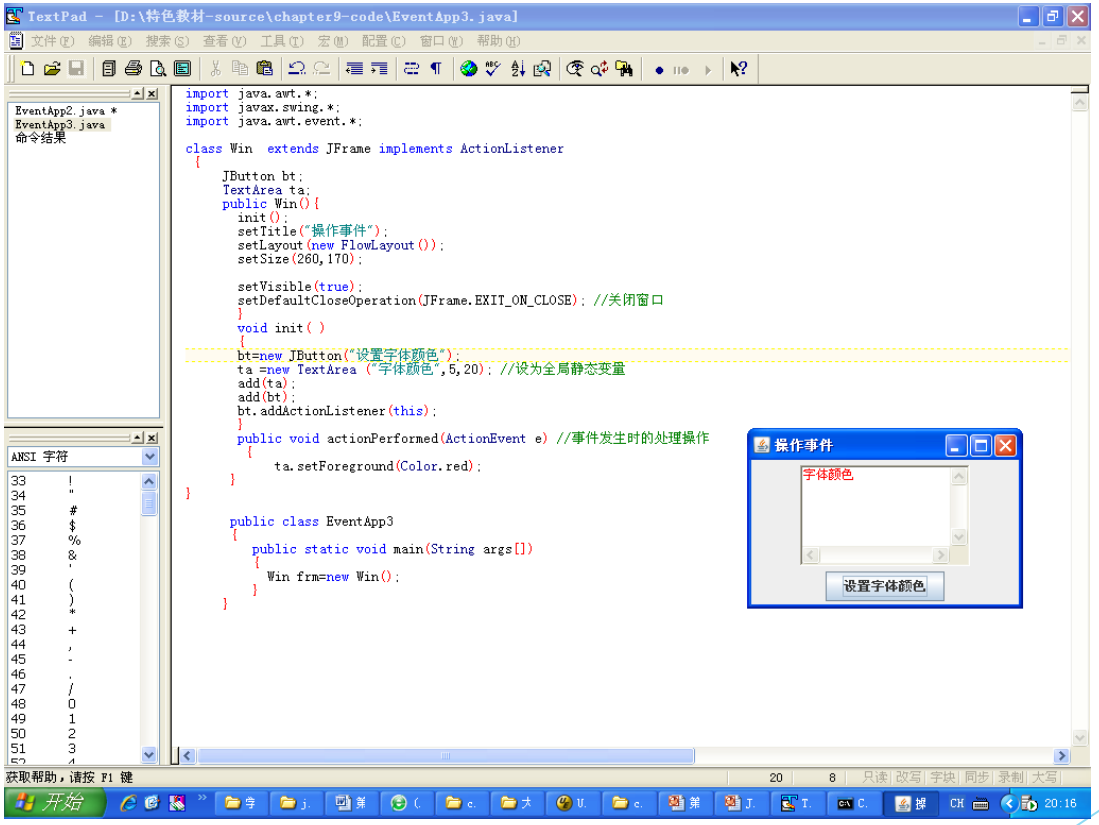
1．让包含事件源的对象来担任监听者

- 在一个窗口中摆放两个组件：命令按钮和文本区。当单击命令按钮后，将文本区中的字体颜色设置为红色（未加入事件处理代码）。
- 1 import java.awt.*;
- 2 import javax.swing.*;
- 3 import java.awt.event.*;
- 4 public class EventApp1 extends JFrame
- 5 {
- 6　static TextArea ta =new TextArea ("字体颜色",5,20); //设为全局静态变量
- 7　public static void main(String args[])
- 8　{
- 9　　EventApp1 frm=new EventApp1();
- 10　　JButton bt=new JButton("设置字体颜色");

1．让包含事件源的对象来担任监听者

```
11      frm.setTitle("操作事件");
12      frm.setLayout(new FlowLayout());
13      frm.setSize(260,170);
14      frm.add(ta);
15      frm.add(bt);
16      frm.setVisible(true);
17      frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//关闭窗口
18   }
19 }
```


第9章 图形用户界面-2011.doc - Microsoft Word
【案例分析】
9.1 图形用户界面概述
9.1.1 图形用户界面路线
9.1.2 组件分类
9.1.3 常用容器类的应用
22 ta.setForeground(Color.red);
23 }
24)
程序运行结果：
操作事件
字体颜色
设置字体颜色


TextPad - [D:\特色教材-source\chapter9-code\EventApp3.java]
EventApp2.java *
EventApp3.java
命令结果
ANSI 字符
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Win extends JFrame implements ActionListener
{
JButton bt;
TextArea ta;
public Win(){
init();
setTitle("操作事件");
setLayout(new FlowLayout());
setSize(260,170);
setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //关闭窗口
}
void init( )
{
bt=new JButton("设置字体颜色");
ta =new TextArea ("字体颜色",5,20); //设为全局静态变量
add(ta);
add(bt);
bt.addActionListener(this);
}
public void actionPerformed(ActionEvent e) //事件发生时的处理操作
{
ta.setForeground(Color.red);
}
}
public class EventApp3
{
public static void main(String args[])
{
Win frm=new Win();
}
}
操作事件
字体颜色
设置字体颜色
获取帮助，请按 F1 键
开始

2．定义内部类来担任监听者并实现接口

- 使用内部类充当监听者。
- 1 import java.awt.*;
- 2 import javax.swing.*;
- 3 import java.awt.event.*;
- 4 public class EventApp4
- 5 {
- 6　static TextArea ta =new TextArea ("字体颜色",5,20); //设为全局静态变量
- 7　public static void main(String args[])
- 8　{
- 9　　**JFrame frm=new JFrame();**　　//创建一个独立Java窗口

2．定义内部类来担任监听者并实现接口

- 使用内部类充当监听者。
- 10　　JButton bt=new JButton("设置字体颜色");
- 11　　**bt.addActionListener(new MyListener());**
- 12　　frm.setTitle("操作事件");
- 13　　frm.setLayout(new FlowLayout());　　//布局是流式布局管理
- 14　　frm.setSize(260,170);
- 15　　frm.add(ta);
- 16　　frm.add(bt);
- 17　　frm.setVisible(true);　　//设置窗口可见

```
18      frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //关闭窗口
19   }
20
21  //定义内部类MyListener并实现ActionListener接口
22  static class MyListener implements ActionListener
23  {
24    public void actionPerformed(ActionEvent e)
25    {
26      ta.setForeground(Color.red);
27    }
28  }
29}
```

3．事件处理采用匿名类法

- 改进在程序中使用匿名类充当监听者，监听者注册和事件处理由一条语句实现。
- 1 import java.awt.*;
- 2 import javax.swing.*;
- 3 import java.awt.event.*;
- 4 public class EventApp5
- 5 {
- 6　static TextArea ta =new TextArea ("字体颜色",5,20); //设为全局静态变量
- 7　public static void main(String args[])
- 8　{
- 9　　JFrame frm=new JFrame();
- 10　　JButton bt=new JButton("设置字体颜色");
- 11　　**bt.addActionListener(new ActionListener()**
- 12　　{

```
13          public void actionPerformed(ActionEvent e)
14        {
15          ta.setForeground(Color.red);
16        }

17      });
18      frm.setTitle("操作事件");
19      frm.setLayout(new FlowLayout());
20      frm.setSize(260,170);
21      frm.add(ta);
22      frm.add(bt);
23      frm.setVisible(true);
24  frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //关闭窗口
25   }
26}
```

适配器类

- 并不是所有的事件处理都像按钮点击那样简单。在正规的程序中 ，往往希望用户在确认没 有丢失所做工作之后再关闭程序。当用户关闭框架时，可能希望弹出一个对话框来警告用户设 有保存的工作有可能会丢失，只有在用户确认之后才退出程序。

一般组件

有了窗口之后，还要创建其他组件，然后将其添加到窗口中，本节将介绍组件的功能及创建方法。

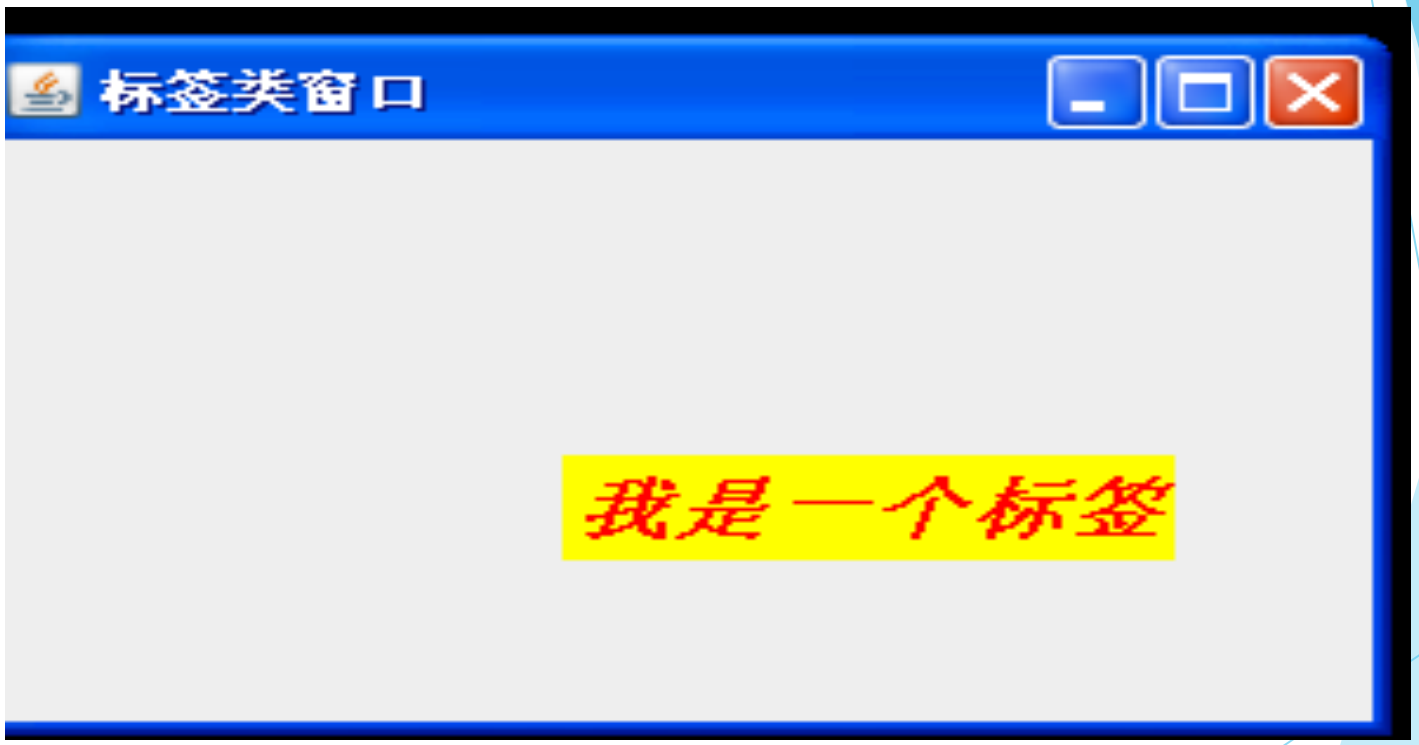
创建标签

标签label是用来在窗口中显示文字的控件，它可用java.awt类库里的Label类创建。

在框架窗口中加入指定大小的标签。

```
//app11_4.java      指定标签对象的大小
import java.awt.*;
public class app11_4
{
  static Frame frm=new Frame("标签类窗口");
  static Label lab=new Label();   //创建标签对象
  public static void main(String args[])
  {
    frm.setLayout(null);   //取消页面设置
    frm.setSize(300,200);
    frm.setBackground(Color.pink);
    lab.setText("我是一个标签");
```

```
lab.setAlignment(Label.CENTER);
 lab.setBackground(Color.yellow);
 lab.setForeground(Color.red);
 lab.setLocation(120,90);
 lab.setSize(130,30);
 Font fnt=new Font("Serief",Font.BOLD+Font.ITALIC,20);
 lab.setFont(fnt);
 frm.add(lab);
 frm.setVisible(true);
  }
}
```


标签类窗口
我是一个标签

创建命令按钮–滑动条

组件名	功能
Button	Button
Canvas	用于绘图的画布
Checkbox	复选框组件（也可当做单选框组件使用）
CheckboxGroup	用于将多个Checkbox 组件组合成一组， 一组 Checkbox 组件将只有一个可以 被选中， 即全部变成单选框组件
Choice	下拉选择框
Frame	窗口， 在 GUI 程序里通过该类创建窗口
Label	标签类，用于放置提示性文本
List	JU表框组件，可以添加多项条目
Panel	不能单独存在基本容器类，必须放到其他容器中
Scrollbar	滑动条组件。如果需要用户输入位于某个范围的值， 就可以使用滑动条组件，比如调 色板中设置 RGB 的三个值所用的滑动条。当创建一个滑动条时，必须指定它的方向、初始值、 滑块的大小、最小值和最大值。
ScrollPane	带水平及垂直滚动条的容器组件
TextArea	多行文本域
TextField	单行文本框

java.awt

Class Button

java.lang.Object
  └ java.awt.Component
    └ java.awt.Button

javax.swing

Class JButton

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

创建文本编辑组件

文本编辑组件是可以接收用户的文本输入并具有一定编辑功能的界面元素。

文本编辑组件分为两种，一种是单行文本编辑组件，简称文本框TextField，另一种是多行文本编辑组件，简称文本区TextArea。

java.awt

Class TextField

java.lang.Object
  └ java.awt.Component
    └ java.awt.TextComponent
      └ **java.awt.TextField**

javax.swing

Class JTextField

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └ javax.swing.text.JTextComponent
          └ **javax.swing.JTextField**

java.awt

Class TextArea

java.lang.Object
  └ java.awt.Component
    └ java.awt.TextComponent
      └ **java.awt.TextArea**

javax.swing

Class JTextArea

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └ javax.swing.text.JTextComponent
          └ **javax.swing.JTextArea**

```
import java.awt.*;
public class TextFieldDemo extends Frame
{
   TextField tf1=new TextField("该文本框不可编辑",20);
   static TextField tf2=new TextField("口令输入框",20);
   public TextFieldDemo(String str)
   {
     super(str);
     tf1.setBounds(20,70,120,20);
     tf1.setEditable(false);
     add(tf1);
    }
```

```
public static void main(String[] args)
    {
        TextFieldDemo frm=new TextFieldDemo("文本编辑功能");
        TextArea ta=new TextArea("您好！
",10,20,TextArea.SCROLLBARS_VERTICAL_ONLY);
        frm.setLocation(200,150);
        frm.setLayout(null);
        tf2.setBounds(20,40,120,20);
        tf2.setEchoChar('*');
        ta.setBounds(20,100,140,100);
        frm.add(tf2);
        frm.add(ta);
        System.out.println(tf2.getText());
        frm.setVisible(true);
     }
   }
```


C:\WINDOWS\system32\cmd.exe
口令输入框
文本编辑功能
******
该文本框不可编辑
您好!
您的口令应经输入

创建复选框组件

复选框(Checkbox)是让用户选取项目的一种组件，用户利用该组件来获得相应的输入。Java语言提供了Checkbox类来创建复选框。复选框分为复选和单选两种，其中复选Checkbox可以单独使用，而单选框还必须配合CheckboxGroup类将其组成组来使用。

菜单与对话框

- 菜单是图形用户界面的重要组成部分，由菜单栏（JMenuBar）、菜单（JMenu）、菜单项（JMenuItem）、弹出式菜单（JPopupMenu）和包含复选框的菜单项（JCheckboxMenuItem）等对象组成。
- 在窗口中创建菜单的概念是，将“菜单栏”加入到指定的窗口中，接着将“菜单”加入到“菜单栏”中，然后将“菜单项”和“子菜单”加入到“菜单”中，如图：

菜单与对话框

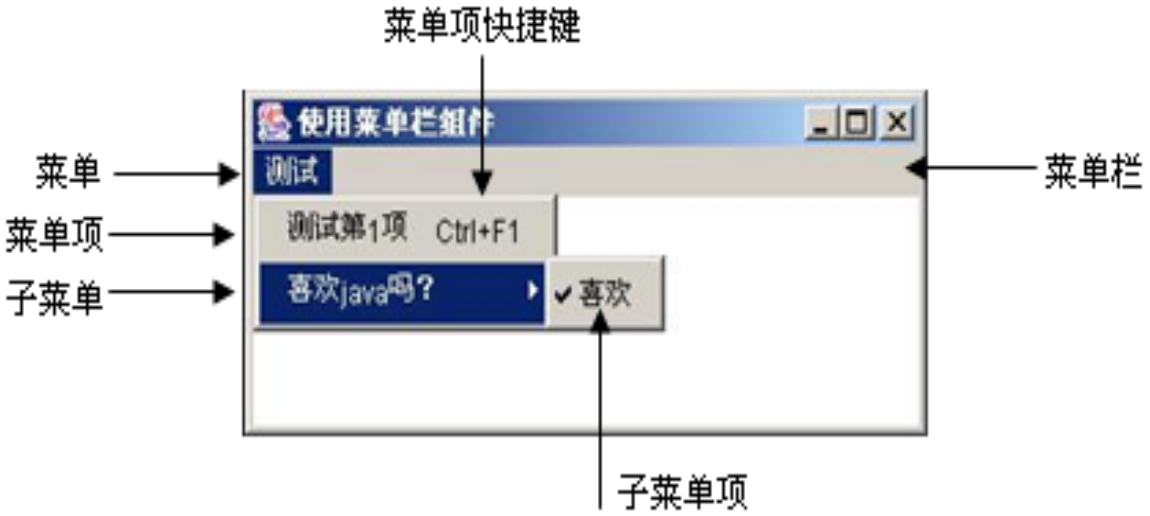


图 9.22 菜单组件窗口

布局管理器

所谓“页面设置”(layout)是指窗口上的组件遵循一定的规则来排列，并会随着窗口大小的变化来改变组件大小与位置的一种布局方式。

Java.awt包中共定义了五种布局管理器，每个布局管理器对应一种布局策略，分别是：流式布局管理器FlowLayout、边界式布局管理器BorderLayout、卡片式布局管理器CardLayout、网格式布局管理器GridLayout和网袋布局管理器GridBagLayout。

流式布局管理器FlowLayout

流式布局是一种最基本的布局，它是一种流式页面设计。流式布局管理器FlowLayout的布局策略是：

1. 组件按照加入容器的先后顺序从左向右排列；

2. 一行排满之后就自动地转到下一行继续从左向右排列；

3. 每一行中的组件都居中排列。

边界式布局管理器BorderLayout

边界式布局管理器BorderLayout将显示区域按地理方位分为东(East)、西(West)、南(South)、北(North)、中(Center)五个区域。

在将组件加入容器中时，都应该指出把这个组件加在哪个区域中，若没有指定区域，则默认为中间。

边界布局管理器的应用见教材【例10.18】

卡片式布局管理器CardLayout

卡片式布局管理器CardLayout的页面配置方式是把“窗口容器”中的所有组件如同堆叠起来的一付“扑克牌”，每次只能显示最上面的一张一样，这个被显示的组件将占据所有的容器空间。

网格式布局管理器GridLayout

网格式布局管理器GridLayout提供的页面布局规则是将容器的空间划分成若干行与列的网格形式，在容器上添加组件时，它们会按从左到右、从上到下的顺序在网格中排列。

网袋布局管理器GridBagLayout

GridBagLayout布局管理器也是将容器中的组件按行、列的位置摆放，但各组件所占据的空间可以是互不相同的。在GridBagLayout布局管理器中可以为每个组件指定其占据的网格个数。