面向对象程序设计Java

江春华

电子科技大学信息与软件工程学院

内　容

第11章 GUI编程

1. Applet编程
2. GUI编程基础
3. AWT组件
4. Swing组件
5. AWT事件编程

Applet编程

- Applet简介：
  - Applet与Application两种程序都是Java编程。
  - Applet是能够嵌入到一个HTML页面中，且可通过Web浏览器下载和执行的一种Java类。
  - 它是Java的一种类型容器(container)，其执行方式不同于Application程序是从main()方法被调用开始的，一个Applet的运行生命周期则要复杂些。

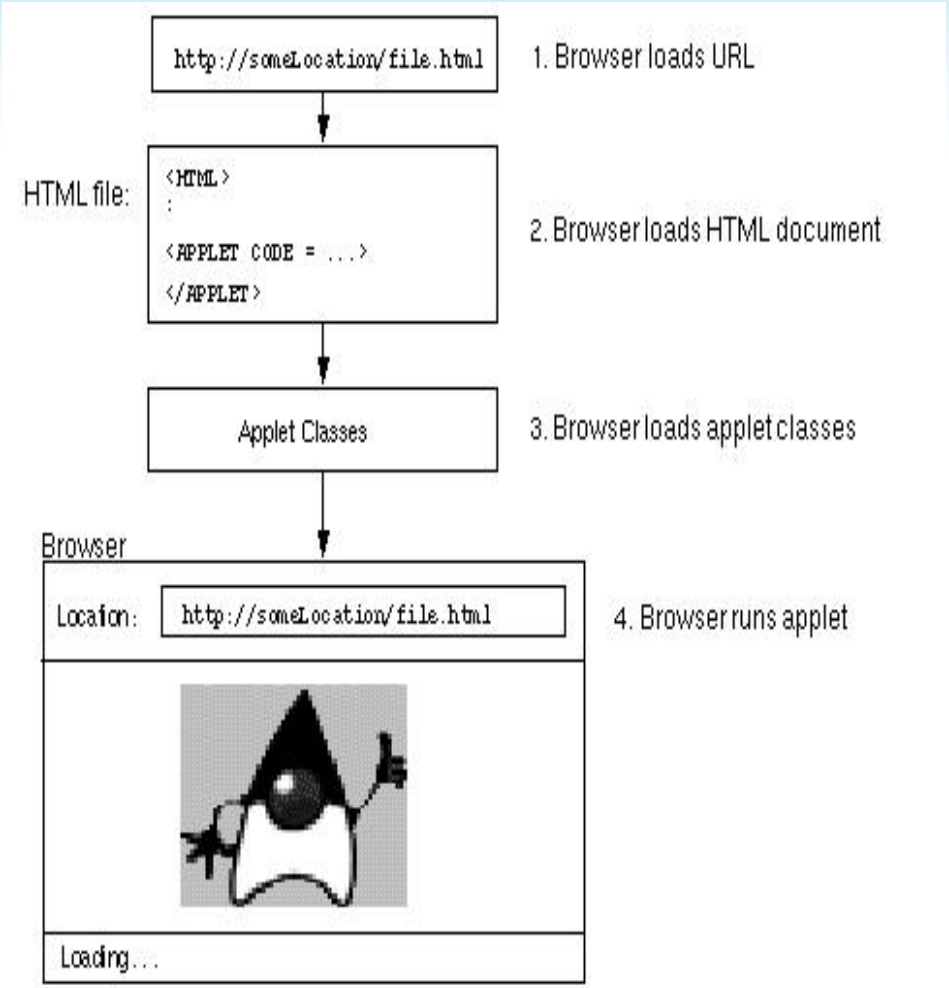
Applet编程

```
<Applet code=Hello.class width=400
        height=300>
</Applet>
```

- Applet是一
  并通过浏览
- 嵌入到一个HTML页面中；
- 可通过Web浏览器下载并执行的一种Java类；
- 它是客户端程序，不同于JavaEE的Servlet的服务器端程序，它是在客户端的浏览器上运行的；
- 它的运行会受到一定的安全限制；
- 它对多媒体应用支持好。

Applet编程

- Applet装载的步骤是：
  - 浏览器装入URL；
  - 浏览器装入HTML文档；
  - 浏览器装入Applet类；
  - 浏览器运行Applet。



Applet编程

❖Applet安全性限制：

Applet程序的安全限制，保证网络上传输的数据不会受病毒的侵入和数据的完整性。

- 不可加载库或定义本地方法；
- 不可在执行它的主机中以通常方式读写文件；
- 只能与它所在的主机建立网络连接；
- 不可在执行它的计算机上启动其它的程序。

Applet编程

- Applet编程，Applet程序都是Applet类的子类编程。
- Applet类：
  - Applet类是Panel类的子类；
  - Applet类构造方法只有一个：public Applet()
  - Applet类的主要方法有：

    init()、start()、paint()、stop()和destroy()

Applet编程

❖Applet程序编写

- 引入Applet类：

```
import java.applet.Applet;
```

- 创建Applet类的子类：

```
public class MyApplet extends Applet
```

- 覆盖Applet方法：

```
public void init()
```

Applet编程

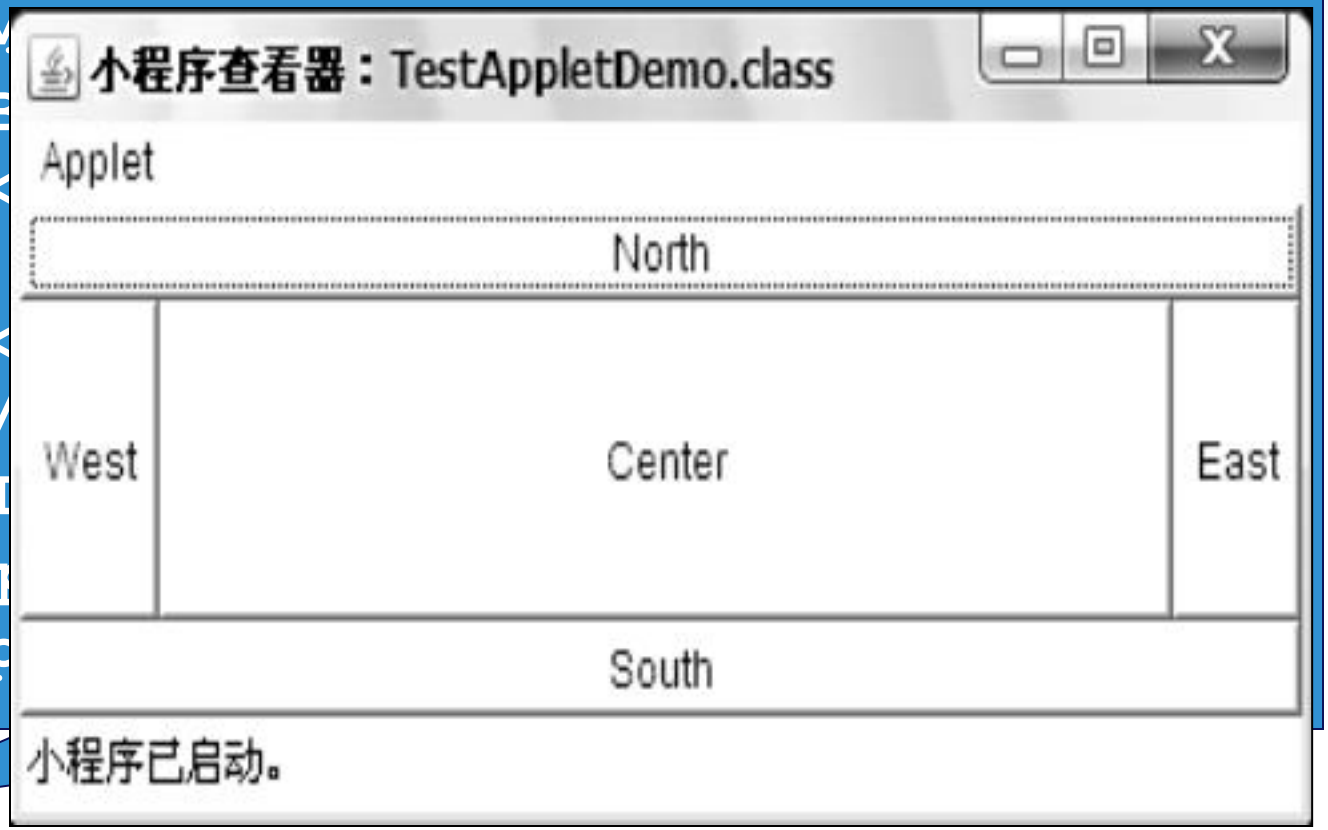
- App

和中

```
impor
impor
publi
   pub
      s
      a
      a
      a
      a
      add("C
   }
}
```

嵌入到TestAppletDemo.html文件中：

```
   <HTM
      <H
         <
         <
      </
   </HT
观察App
      App
```



Applet编程

Applet生命周期：

- **init()**：初始化处理，只在Applet首次装入时被调用，在调用start()之前完成。
- **start()**：在init()方法后被调用，使Applet处于“活动”。当浏览器从图标化后恢复或从链接返回时被调用。
- **stop()**：在Applet不可见时被调用。浏览器被图标化或链接到另一个URL时调用该方法使动画停止。

GUI编程基础

❖GUI设计概念：

- 图形用户界面(Graphics User Interface)简称GUI。
- 使用**图形方式**，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。
- 生成图形用户界面的包是**java.awt**，简称抽象窗口工具集，即**AWT**（Abstract Window Toolkit）。
- GUI中的三个基本元素：**容器**、**组件**和**布局管理器**。

GUI编程基础

- 设计和实现GUI的工作主要有：
  - **创建组件**(Component)

    创建组成界面的各种元素，如按钮、文本框等。
  - **指定布局**(Layout)

    根据具体需要排列组件的位置关系。
  - **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。

AWT组件

- GUI类层次及java.awt包的基本功能
  - **GUI类**：Component的所有子类和LayoutManager布置管理器类。它是实现按钮、列表框、检查框等GUI的基础。
  - **菜单类**：包括MenuBar、Menu、MenuItem、CheckboxMenuItem、PopupMenu和MenuShortcut类用于菜单设计。
  - **图形类**：包括Graphices，Image，Color，Font，FontMetrios类。实现基本的作图功能。
  - **几何类**：包括Point，Polygon，Dimension，Rectangle四个类为GUI设计的辅助类，提供几何尺寸和位置等。
  - **事件类**：包括AWTEventMulticaster，AWTEvent和Event类，完成对事件的响应。
  - **其他类**：包括MediaTracker，Toolkit等。

AWT组件

- 容器、部件和布局管理器：
  - 容器(Container)：由Container抽象类派生的所有类都可生成可以容纳部件的特定对象，如Frame、ScrollPane、Panel、Dialog、FileDialog、Window及Applet。
  - 部件(Component)：由抽象类Component子类生成的对象，如Button、Canvas、Checkbox、Choice、Label、List、Scrollbar、TextArea、TextField及菜单。
  - 布局管理器(LayoutManager)：决定部件“加入”到容器中的位置和部件本身的优选尺寸。

AWT组件

- GUI程序编程步骤：
  - 选择容器

```
Panel panel = new Panel();
```

  - 为容器选择一种布置管理器

```
panel.setLayout(new BorderLayout());
```

  - 将部件加入到容器中

```
panel.add("North",new Label("OK"));
```

AWT组件

- **Frame**类和**Panel**类是容器的两个基本类。
- **Frame**类：
  - Frame类是Window类的子类；
  - 外观像Windows系统下的窗口，有标题、边框、菜单、大小等；
  - 其对象实例化后，是没有大小和不可见，通过调用setSize()设置大小，调用setVisible(true)来设置该窗口为可见；

AWT组件

Frame类：

- 默认的布局管理器是BorderLayout；
- 构造器有两个：

  Frame()和Frame(String s)，s参数是标题。

- 关闭Frame窗口时，通过调用setVisible(false)方法，再实现WindowListener监听器接口的windowClosing()方法，将窗口从屏幕中除去。

AWT组件

- **Panel**类作为面板使用具有以下特点：
  - 它不是顶层窗口，必须位于窗口或其他容器之内；
  - 它可容纳其他组件，在程序中经常用于布局和定位；
  - 默认的布局管理器是`FlowLayout`，可使用`setLayout()`方法改变其默认布局管理器；

AWT组件

❖布置管理器：

- 决定部件“加入”到容器中的位置和部件本身的优选尺寸；
- 容器使用**setLayout()**方法改变其布局管理器；
- 布局管理器常用的有四种：

FlowLayout	流布局管理器
BorderLayout	周边布局管理器
GridLayout	栅格布局管理器
CardLayout	卡片布局管理器

AWT组件

- FlowLayout(流布置管理器):
  - 以add()语句的顺序依次放置部件;
  - 是Applet和Panel容器的缺省布置管理器;
  - 构造器有三个:

FlowLayout()

FlowLayout(int align)

FlowLayout(int align,int hgap,int vgap)

其中: align为对齐: LEFT、CENTER和RIGHT;
hgap和vgap是其行、列间隔的象素。

AWT组件

❖ **FlowLayout**应用示例。

```
import java.awt.*;
import
public
   publi
      set
      for
         a
         v
         s
         t
         c
      }
   }
}
```

AWT组件

❖BorderLayout(周边布置管理器):

- 按东、南、西、北、中5个方位来放置部件;
- 是容器Frame, Window和Dialog的缺省布置管理器。
- 每个部件的尺寸都填满相应空间;
- 只有处于"中间"的部件具有特权，它可以在其余四个周边部件缺少一个或几个时，扩大"中间"部件尺寸，占满剩余空间。
- 构造器有二个:

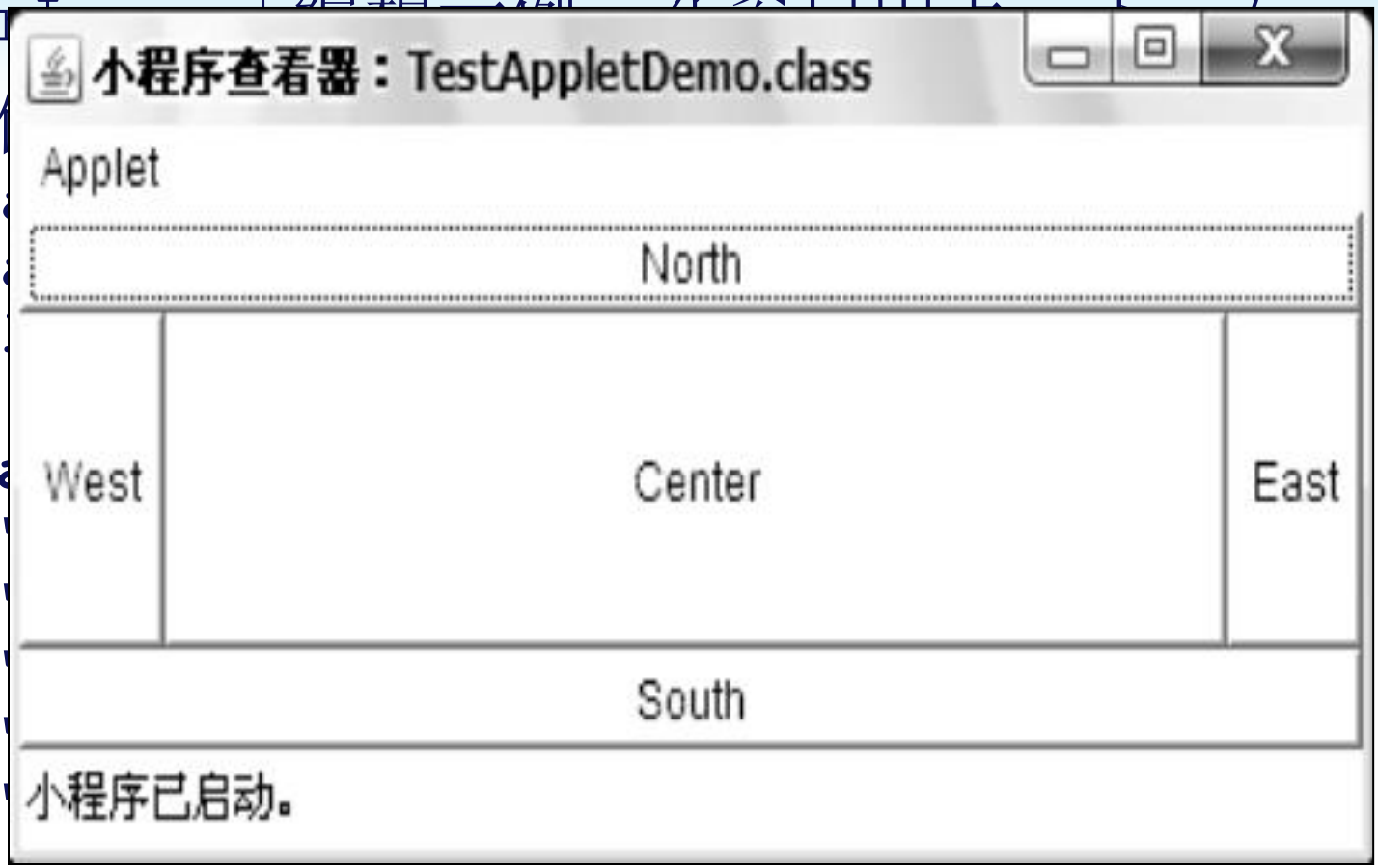
BorderLayout()和**BorderLayout(int hgap,int vgap)**

其中：hgap和vgap是其行、列间隔的象素。

AWT组件

- Borde[illegible]编程示例，在容器中央、下、左、右和中[illegible]

```
import ja
import ja
public cl
  public
    setLa
    add(
    add(
    add(
    add(
    add(
  }
}
```



AWT组件

CardLayout(卡片布置管理器):

- 把不同信息分类记录在不同的卡片上,只有最上面卡片可见;
- 通过对卡片

```
add("1",new Button("Card 1"));
add("2",new Button("Card 2"));
add("3",new Button("Card 3"));
add("4",new Button("Card 4"));
add("5",new Button("Card 5"));
```

first()

last()

通过这五

。

AWT组件

❖CardLayout编程示例：在窗口中加入并显示按钮。

```
import java.awt.*;
import java.applet.*
public class CardLay                                    {
  public void init()
    CardLayout layou
    setLayout(layout)
    add("1",new Butto
    add("2",new Butto
    add("3",new Butto
    for(int i=1; i<4
      if(i == 1) layo
      else layout.nex
      validate();
      try{Thread.slee                                   n e){};
    }
  }
}
```

AWT组件

GridLayout(格栅布置管理器):

- 基于格栅(即行列)来放置部件;
- 构造器有三个:

```
GridLayout()
GridLayout(int row,int col)
GridLayout(int row,int col,int hgap,int vgap)
```

其中:

row,col分别表示行数和列数，hgap和vgap分别表示行和列间隔的象素。

AWT组件

❖GridLayout编程示例：在窗口中加入并显示按钮。

```
import java.a
import java.a
public class (                                    let {
   public void
    setLayout(n
    add(new But
    add(new But
    add(new But
    add(new But
    validate()
    setVisible
    try{Thread                                    on e){};
    add(new But
   }
}
```

AWT组件

AWT常用组件：

- AWT部件是指**Component**子类生成的对象，是设计GUI程序的基本元素；
- Component类是一个庞大的类，它有很多的方法，涉及到事件注册、组件移动和尺寸、位置设置、图形和风格相关、布局管理和容器绘制、父子组件获取、状态设置及判断、图象处理、组件对等、菜单等相关的方法。
- 部件：介绍常用的部件

  Button，**Label**，**Checkbox**，**Choice**，**Scrollbar**，**TextField**，**TextArea**，**Canvas**的使用方法。

AWT组件

- **Button**(按钮):
  - 构造器有二个:
    Button()和Button(String lab)
    其中: 参数lab为按钮标签。
  - 常用方法有:
    getLabel()和setLabel()
    分别获得和设置按钮标签。
  - 与事件处理有关的方法:

```
addActionListener(ActionListener l)
removeActionListener(ActionListener l)
```

AWT组件

❖Button示例：se 的使用。

```
import java.awt.*
import java.apple
public class Butt
   Button b;
   public void ini
     b=new Button(
     add(b);
     validate();
     setVisible(tr
     try{Thread.sl                                    e){};
     String label=
     b.setLabel(la
   }
}
```

AWT组件

- **Label**（标签）：
  - 构造器有三个：

    Label()和Label(String s)：s为字符串标签。

    Label(String s, int align)：align对齐。
  - 常用方法有：

    getText()和setText()分别获得和设置字符串。

    getAlignment()和setAlignment()为对齐参数。
  - 与事件处理有关的方法：

    addActionListener(ActionListener l)

    removeActionListener(ActionListener l)

AWT组件

❖Label示例：Label类方法的使用。

```
import java.awt.*;
import java.applet.*;
public class LabelTes
  public void init()
     setLayout(new Gr
     Label lable1 = n
     Label lable2 = n                                    TER);
     Label lable3 = n                                    HT);
     add(lable1);    a
     validate();     s
     try{Thread.sleep                                    ;
     String str = lab
     lable2.setText(s
     int align = lable
     lable2.setAlignm
  }
}
```

AWT组件

- **TextField**(单行文本区)：单行可编辑文本输入区
  - 构造器有四个：

    TextField()和TextField(String text,int cols)

    TextField(int cols)和TextField(String text)

    其中：text为字符串，cols为列数。
  - 常用方法有：

    getText()、isEditable()、setEditable()、setText()、setEchoChar()、getEchoChar()等。
  - 与事件处理有关的方法：

    addActionListener(ActionListener l)

    removeActionListener(ActionListener l)

AWT组件

- **TextArea**(文本区)：多行可编辑文本区。
  - 构造器有五个：

    TextArea()和TextArea(String txt,int r,int c)

    TextArea(int c)和TextArea(String txt)

    TextArea(String txt,int r,int c,int scr)

    其中：txt为字符串，r和c为行列数，scr为滚动条。
  - 常用方法有：

    getText()、isEditable()、setEditable()、setText()、setEchoChar()、getEchoChar()等。
  - 与事件处理有关的方法：

    addActionListener(ActionListener l)

    removeActionListener(ActionListener l)

AWT组件

- **Checkbox**(检查框):二态true/false选择框，小正方框显示
  - 构造器有四个:

```
Checkbox()和Checkbox(String lab)
Checkbox(String lab,boolean st)
Checkbox(String lab,boolean st,CheckboxGroup grp)
Checkbox(String lab,boolean st,CheckboxGroup grp,
         boolean b)
```

其中：lab为标签，st为初始状态，grp为互斥检查框组中的一个。

  - 常用方法有:

getLabel()、setLabel()、getState()、setState()等。

  - 与事件处理有关的方法:

```
addItemListener(ItemListener l)
removeItemListener(ItemListener l)
```

AWT组件

- **Choice**(选择框):弹出式选项表。
  - 构造器有一个:

    `Choice()`
  - 选择框对象不具有任何选择项，所有的选择项只能使用`addItem()`方法一个个加入。
  - 常用方法有:

    `addItem()`、`insert()`、`remove()`和`getSelectedItem()`等。
  - 与事件处理有关的方法:

    `addItemListener(ItemListener l)`

    `removeItemListener(ItemListener l)`

AWT组件

❖Choice示例：Choice类方法的使用。

```
import java.awt.*
import java.apple
public class Choi
  public void ini
    Choice ch = n
    ch.addItem("C
    ch.addItem("V
    ch.addItem("S
    ch.addItem("A
    add(ch);
  }
}
```

AWT组件

- 菜单设计：
  - 菜单是一个特殊部件，只能被加到Frame容器中；
  - 菜单由MenuComponent类派生的；
  - 类派生了五个类：

    MenuBar，Menu，MenuItem，

    CheckboxMenuItem和PopupMenu。

AWT组件

- MenuBar菜单栏：
  - 用于设计主菜单栏；
  - 主菜单栏可以加入多个主菜单；
  - 每个主菜单的菜单项，由MenuItem，Menu和CheckboxMenuItem组成；

其中：

- ✓MenuItem是普通菜单项；
- ✓CheckboxMenuItem是标记菜单项；
- ✓Menu是子菜单。

AWT组件

- Menu菜单。经常使用到的方法有：
  - add()在菜单中加入菜单项或字符串；
  - addSeparator()在菜单项列表中加入分隔线；
  - insertSeparator()在菜单项中插入分隔符；
  - insert() 在指定位置插入菜单项或字符串；
  - remove() 删除菜单项。

AWT组件

- MenuItem和CheckboxMenuItem菜单项：
  - 是用户的最终选项；
  - 当用户选择了这些项后，菜单将关闭，同时发出相应的命令；
  - 这些命令就是用户要处理的事件。

AWT组件

- 创建菜单栏和菜单项。

  菜单必须进行下列三步操作：

  - 定义主菜单类，加入所有的菜单项，包括子菜单；
  - 生成主菜单栏，加入主菜单；
  - 用`setMenuBar()`把主菜单栏加入到窗口中的固定位置。

AWT组件

- 创建菜单栏和菜单项示例。

```
import java.awt.*;
import java.awt.event.*;
public class MenuTest2 extends Frame {
 public MenuTest2() {
   super("MenuTest2 Window");
   FileMenu fileMenu = new FileMenu();
   HelpMenu helpMenu = new HelpMenu();
   MenuBar mb = new MenuBar();
   fileMenu.setFont(new Font("Fancy", Font.PLAIN, 18));
   helpMenu.setFont(new Font("Fancy", Font.PLAIN, 18));
   mb.add(fileMenu);
   mb.add(helpMenu);
   setMenuBar(mb);
   enableEvents(AWTEvent.WINDOW_EVENT_MASK);
   setSize(new Dimension(400, 300));
   setVisible(true);
 }
```

AWT组件

```
  public void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      setVisible(false);
      dispose();
      System.exit(0);
    } else super.processWindowEvent(e);
  }
  public static void main(String[] args) {
    new MenuTest2();
  }
}
```

AWT组件

```
class FileMenu extends Menu {
   public FileMenu() {
      super("File");
      add(new MenuItem("New"));
      add(new MenuItem("Open"));
      MenuItem save = new MenuItem("Save");
      add(save);
      save.setEnabled(false);
      addSeparator();
      MenuItem print = new MenuItem("Print");
      add(print);
      print.setEnabled(false);
      add(new MenuItem("Print setup"));
      addSeparator();
      add(new MenuItem("Exit"));
   }
}
```

AWT组件

```
class HelpMenu extends Menu {
   public HelpMenu() {
      super("Help");
      Menu subMenu = new Menu("Content");
      subMenu.add(new MenuItem("Document"));
      subMenu.add(new MenuItem("Source"));
      subMenu.add(new MenuItem("Class"));
      add(subMenu);
      addSeparator();
      CheckboxMenuItem check =
              new CheckboxMenuItem("Display Time");
      add(check);
      check.setState(true);
      add(new MenuItem("About MenuTest"));
   }
}
```

Swing组件

- 替代AWT的图形界面类Swing。
- Swing是AWT的扩展，它提供了更强大和更灵活的组件集合。
- 许多新的组件，如选项板、滚动窗口、树、表格等。
  - 与AWT组件不同，Swing组件实现了不包括任何与平台相关的代码。
  - Swing组件是纯Java代码，因此与平台无关。一般用轻量级(lightweight)这个术语描述这类组件。

Swing组件

- JApplet类：
  - 在javax.swing包中，是Applet的子类。
  - 与Applet的差别在于：
    - Applet的缺省布局策略是FlowLayout，而JApplet的缺省布局策略是BorderLayout。
    - 向JApplet中加入Swing组件时不能直接用add()方法，而必须先使用JApplet的方法getContentPane()获得一个Container对象，再调用这个Container对象的add()方法将JComponent及其子类对象加入到JApplet中。

Swing组件

JFrame类：

- 直接从Frame继承而来的。
- 重要方法如下：
  - setIconImage() 窗口最小化（在Java中称为图标化）时，把一个Image对象用作图标。setTitle 设置窗口中标题栏的文字。
  - setResizable()设置用户是否可以改变框架大小。
  - dispose()方法 关闭窗口，并回收该窗口的所有资源。
  - setLocation() 设置组件的位置。
  - setBounds() 重新设置组件的大小和位置。

Swing组件

UESTC

JButton类：

- JButton是一个按钮工具，拥有一个图标。
- 这个图标可以是用户自己绘制的图形，也可以是已经存在的.gif图像。
- 与AWT类中Button组件用法类似，但是功能上增强了很多。
- 使用的许多按钮中，具有这样一种功能：当鼠标在按钮上停留很短的几秒钟时，屏幕上将会出现一个简短的关于这个按钮的作用的提示信息。

AWT事件处理

- 事件定义：系统对来源自用户对AWT部件操作的响应。
- 事件的状态信息：
  - 动作命令 与该动作相关的命令名；
  - 修饰符 在动作期间是否有shift, ctrl, alt等修饰键被按下。

AWT事件处理

- 事件源和事件处理器
  - 事件源：产生事件的对象；
  - 事件处理器（监听器）：负责处理事件的方法。
- 委托事件模型
  - 事件源产生一个事件，并把该事件发送到一个或多个监听程序，监听程序等待这个事件并处理它，然后返回。
  - 程序把事件的处理"委托"给一段"代码"。监听程序必须注册一个事件源才能自动的接收这个事件。监听程序必须实现接收和处理这个事件的方法来负责处理事件。

❖委托事件模型进行事件处理三个操作

- 建立监听器；

  public class EventLister implements ActionListener

- 注册事件源；

  exit.addActionListener(new EventLister());

- 进行事件处理。

  public void actionPerformed(ActionEvent e){//code}

事件类型

事件	接口名	接口定义的方法
Action	ActionListener	actionPerformed(ActionEvent)
Item	ItemListener	itemStateChanged(ItemEvent)
Mouse Motion	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
Mouse Button	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
Key	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
Focus	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
Adjustment	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
Component	ComponentListener	componentMoved(ComponentEvent) componentHidden(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)
Window	WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent)
Container	ContainerListener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
Text	TextListener	textValueChanged(TextEvent)

AWT事件处理示例1

- 有六个GUI组件：创建一个窗口，实现当在文本框中输入内容完毕后，点击回车键或点击“OK”按钮，使得文本框内的字符串添加到文本区中。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class example extends JFrame implements ActionListener{
  JLabel lb=new JLabel("组件和事件处理：");
  JButton bt=new JButton("OK");
  JTextField tf= new JTextField(20);
  JTextArea ta=new JTextArea(10,20);
  public example(){
    super("综合例子1");
    Container c=getContentPane();
    c.setLayout(new FlowLayout());   //指定布局方式为顺序布局
    c.add(lb);                       //将组件添加到容器中
    c.add(ta);
    c.add(bt);
    c.add(tf);
    bt.addActionListener(this);
    tf.addActionListener(this);
    setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE);
    setSize(200,100);
    setVisible(true);
   }
```

```
    public void actionPerformed(ActionEvent e){
        if(e.getSource()==bt)
            ta.append("现在按下按钮引发的事件"+"\n");
        else{
            ta.append(tf.getText()+"\n");
            ta.append("现在因在文本框里回车而引发的事件"+"\n");
        }
    }
    public static void main(String args[]) {
        new example();
    }
}
```

AWT事件处理示例2

- 程序由按钮产生的事件，由它的上层容器来处理按钮产生的事件。

```
import java.awt.*;
import java.awt.event.*;
public class EventTest implements ActionListener{
  Button left, right;
  Label msg = new Label("No change",Label.CENTER);
  public static void main(String[] args) {
    EventTest et = new EventTest();
    Frame win = new Frame("Evnet Test");
    et.left = new Button("Left Button");
    et.right = new Button("Right Button");
    et.left.addActionListener(et);
    et.right.addActionListener(et);
    win.add("West",et.left);
    win.add("East",et.right);
    win.add("Center",et.msg);
    win.setSize(400,80);
    win.setVisible(true);
  }
```

```
    public void actionPerformed(ActionEvent e) {
        String arg = e.getActionCommand();
        if ("Left Button".equals(arg)) {
          msg.setText("Press Left Button");
        } else if ("Right Button".equals(arg)) {
          msg.setText("Press Right Button");
        }
      }
    }
```

思考问题

1. 在Java中GUI编程中类，分别在哪些包中？

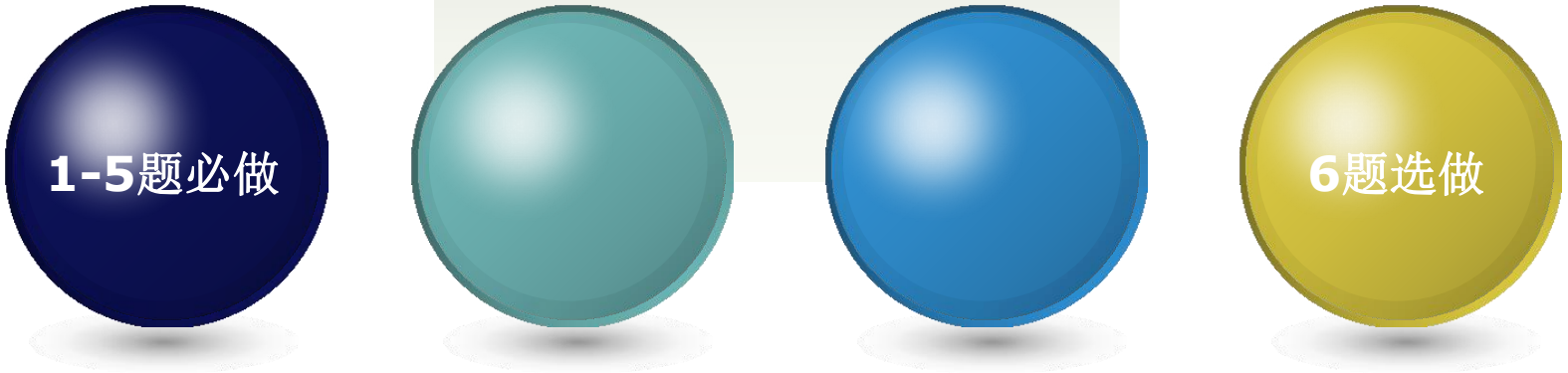
2. 在Java的GUI中的三个基本概念容器、部件和布局管理器是指什么？

3. 在Java事件处理中是如何实现对事件处理的？

第11章作业

本章习题

习题1-6题



http://www.ccse.uestc.edu.cn/teacher/teacher.aspx?id=60

电子科技大学信息与软件工程学院