**第7课 Java 的图形用户界面**

|  |  |  |
| --- | --- | --- |
| **课 题** | Java 的图形用户界面 | |
| **课 时** | 6课时（270 min）。 | |
| **教学目标** | **知识技能目标：**  1．了解 AWT 和 Swing 基本概念。  2．掌握为 GUI 程序添加事件监听的方法。  **思政育人目标：**  让学生通过学习Java 的图形用户界面，软件开发也是一样，不但要保证功能的实现，友好、美观的用户界面也是非常重要的，即用户界面设计。 | |
| **教学重难点** | **教学重点：**Java 图形用户界面程序  **教学难点：**界面布局 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第6节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**展示Java 图形用户界面程序（一）  **任务描述**  编写 GUI 程序，输入两个数并计算其最大公因数。  **任务目标**  1. 能够理解 AWT 和 Swing 的基本概念。  2. 能够掌握 Java 图形用户界面中常用组件的写法。  3. 能够掌握 Java 图形用户界面程序的编写方法。  4. 能够设计出符合用户审美习惯及操作习惯的图形用户界面。  **预备知识**  在常见的图形用户界面应用程序中，都包含诸如按钮、文本标签、文本框等各种图形组件，下面介绍在 Java 中如何使用这些组件。  **一、AWT 和 Swing**  在 Java 1.0 以前，用户接口界面用 AWT（Abstract Window Toolkit）创建，它包含在java.awt 包中。用 AWT 创建的前端应用程序的外观在不同操作平台上是不同的。例如，作为 java.awt.Button 对象实例的按钮，在 Windows 和 Macintosh 环境中是不同的。这导致同一应用程序在不同操作系统平台中有不同的外观。这些小差别意味着程序员必须在不同的用户平台上测试它们的程序代码。  java.awt 包中提供了图形用户界面设计使用的类和接口，其中最核心的类是Component（构件）类。在 Java 1.0 之前，该类是构成 Java 图形用户界面的基础，所有其他组件都是从 Component 类派生而来的。  Swing 是建立在 AWT 之上的，它是包括了大多数轻量组件的集合，它包含在 javax.swing 包中。除提供了 AWT 所缺少的、大量的附加组件外，Swing 还提供了替代 AWT 重量组件的轻量组件。Swing 还包括了一个用于实现包含插入式界面样式等特性的图形用户界面的下层构件。因此，在不同的平台上，Swing 组件都能保持组件的界面样式特性，如双缓冲、调试图形和文本编辑包等。  javax.swing 包提供了管理用户接口组件（如窗口、对话框、菜单、文本标签、文本框、组合框、列表框、滚动条、复选框、单选按钮和按钮等）类的集合。其中的Jcomponent 类是所有用户界面组件的父类，它提供了顶层窗口、可视控件和一些简单元素的公共功能。  **【学生】**思考、讨论。 | **展示Java 图形用户界面程序（一），让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  简述AWT。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示Java 图形用户界面程序（二）  **二、AWT 组件**  **（一）JFrame 框架类**  JFrame 类是一个容器类，它创建一个框架窗体，用于放置其他如文本框、按钮等组件。  JFrame 类的构造函数如下：  JFrame()：创建一个新框架 ;  JFrame(GraphicsConfiguration gc)：创建一个带标题栏图标的新框架 ;  JFrame(String title)：创建一个带标题栏标题的新框架 ;  JFrame(String title, GraphicsConfiguration gc)：创建一个带标题栏标题和图标的新框架。  在创建框架窗口之后，便可调用 JFrame 类的成员函数设置属性，例如：  setVisible(true);  用于设置框架为可见，默认为不可见。  setSize(240,90);  用于设置框架大小为 240×90 像素，默认为 0×0。  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  用于设置默认窗体关闭方式，JFrame.EXIT\_ON\_CLOSE 为函数参数，表示在关闭框架窗口时退出应用程序。  下面的代码用于创建一个空白框架窗口：  import java.awt.\*;  import javax.swing.\*;  class FrameTest extends JFrame  {  public FrameTest()  {  super(" 窗体标题 ");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(240,90);  }  public static void main(String args[])  {  FrameTest F\_T = new FrameTest();  F\_T.show();  }  }  **（二）JButton 按钮类**  JButton 类是一个组件类，用于创建一个按钮，它的构造函数如下：  JButton()：创建一个没有图标和文字的按钮 ;  JButton(Action a)：创建一个没有图标和文字 , 但具有属性的按钮 ;  JButton(Icon icon)：创建一个只有图标的按钮 ;  JButton(String text)：创建一个只有文字的按钮 ;  JButton(String text, Icon icon)：创建一个有图标和文字的按钮。  创建按钮组件后，可以使用 add( ) 方法将其加入到一个容器中，例如：  JButton buttonObj = new JButton(" 按钮 ");  getContentPane().add(buttonObj);  在 Swing 容器（JFrame、JWindows、JDialog 和 JApplet）中，使用 getContentPane 方法获取当前容器。getContentPane 方法返回 Container 类对象，我们可以使用该对象的add 方法将各种组件加入到容器中去。  **（三）JPanel 面板类**  JPanel 类也是一个容器类，与 JFrame 不同的是，JPanel 对象必须放入 JFrame 中才可见。它提供了可以加入其他组件（包括另外的 JPanel 对象）的空间，这样使所有的组件加入到面板，然后将面板加入框架更为简单。JPanel 容器默认为流布局（FlowLayout）格式。  JPanel 的构造函数如下：  JPanel()：创建一个面板 , 各组件按加入顺序放置 ;  JPanel(boolean isDoubleBuffered)：创建一个面板 , 指定是否使用双缓冲策略 ;  JPanel(LayoutManager layout)：创建一个面板 , 并指定使用的布局方式 ;  JPanel(LayoutManager layout, boolean isDoubleBuffered)：创建一个面板 , 并指定使用的布局方式和是否使用双缓冲策略。  在下面的代码中，我们创建一个 JPanel 对象，并将其添加到 JFrame 中，另外将一个 JButton 添加到 JPanel 中。为了和 JFrame 相区分，我们将 JPanel 的背景设置为白色。  JPanel p = new JPanel();  p.setBackground(Color.white);  JButton b = new JButton(" 按钮 ");  p.add(b);  getContentPane().add(p);  这种方式其实是使用 Java 编写 GUI 程序的常用方式。将组件添加到面板，再将面板添加到框架，这样可以更加灵活地进行界面布局和组件管理。  **（四）JLabel 文本标签类**  框架上的静态文本标签是简单的 Swing 组件，用于显示用户界面中的静态文本。用JLabel 类可以创建一个静态文本标签控件。JLabel 类的构造函数如下：  JLabel()：创建一个没有图像和文字的空 JLabel 对象 ;  JLabel(Icon image)：使用指定图像创建一个 JLabel 对象 ;  JLabel(Icon image, int horizontalAlignment)：使用指定的图像和水平对齐方式创建一个 JLabel 对象 ;  JLabel(String text)：使用指定文本创建一个 JLabel 对象 ;  JLabel(String text, int horizontalAlignment)：使用指定文本和水平对齐方式创建一个 JLabel 对象 ;  JLabel(String text, Icon image, int horizontalAlignment)：使用指定的文本、图像和水平对齐方式创建一个 JLabel 对象。  下面的代码用于创建一个静态文本标签，并将其添加到 JFrame 中的 JPanel 中去：  import java.awt.\*;  import javax.swing.\*;  class LabelTest extends JFrame  {  public LabelTest()  {  super(" 窗体标题 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(240,90);  JPanel p = new JPanel();  p.setBackground(Color.white);  JLabel labelobj = new JLabel(" 静态文本标签 ");  p.add(labelobj);  getContentPane().add(p);  }  public static void main(String args[])  {  LabelTest lt = new LabelTest();  lt.show();  }  }  类似的，可以使用下面的代码创建一个图像标签，并将其添加到面板中：  Icon iconobj = new ImageIcon("C:\\1.jpg");  JLabel labelobj = new JLabel(iconobj);  p.add(labelobj);  **（五）JTextField 文本框类**  JTextField 类用于创建一个文本框控件，可以通过该控件接收用户输入的文本内容，也可以在程序运行中改变文本框的内容以输出信息。  JTextField 类的构造函数如下：  JTextField()：创建一个没有初始内容且列数为 0 的文本框 ;  JTextField(String text)：创建一个文本框 , 其初始内容为文本 text 的值 ;  JTextField(int columns)：创建一个宽度 ( 列数 ) 为 columns 的文本框 ;  JTextField(String text, int columns)：用给定的文本和列数 , 创建一个文本框 ;  JTextField(Document doc, String text, int columns)：用给定的文字模式、文本和列数创建一个文本框。  另外，还可以使用 getText 方法来获取文本框里面的内容，该方法的返回值类型是String；对应的，有 setText 方法用于设置文本框内容，该方法的参数是一个 String。关于JTextField 类的具体用法，大家可以参考本单元任务 1 的【任务实施】中的相关代码。  JTextField 类还有一个子类 JPasswordField，这是一个用于输入密码的文本框。当在JPasswordField 控件中输入文本时，只显示出星号（\*），而隐藏了实际内容。JPasswordField 类的使用方法和 JTextField 类似。  **（六）JTextArea 文本区域类**  JTextArea 类用于创建一个文本区域组件，可以用于接收多行文本的输入。JTextArea 类的构造函数如下：  JTextArea()：创建一个新的文本区域组件 ;  JTextArea(Document doc)：用给定的文字模式创建一个新的文本区域 ;  JTextArea(String text)：用给定的文本内容创建一个新的文本区域 ;  JTextArea(int rows, int columns)：用给定的行数和列数创建一个新的文本区域 ;  JTextArea(String text, int rows, int columns)：用给定的文本内容以及行列数创建一个新的文本区域 ;  JTextArea(Document doc, String text, int rows, int columns)：用给定的文字模式、文本内容以及行列数创建一个新的文本区域。  下面的代码用于向窗体中的面板里添加一个 JTextArea 控件。  import java.awt.\*;  import javax.swing.\*;  public class TextAreaTest extends JFrame  {  JPanel panelobj;  JTextArea textareaobj;  public TextAreaTest()  {  super(" 文本区域测试程序 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(240,150);  panelobj = new JPanel();  getContentPane().add(panelobj);  textareaobj = new JTextArea(" 文本区域 ",5,20);  panelobj.add(textareaobj);  }  public static void main(String args[])  {  TextAreaTest ta = new TextAreaTest();  ta.show();  }  }  在上面的界面中输入文字的时候，可以看到文本区域的大小会随着文本行列数的增加自动扩展，这是在实际使用中不应该出现的。为此，可以使用滚动条 JScrollPane 控件。如下面的代码所示，创建一个 JScrollPane 对象，使其与文本区域相关联，并指定水平滚动条和垂直滚动条的显示方式，再将其添加到面板中。  import java.awt.\*;  import javax.swing.\*;  public class ScrollPanelTest extends JFrame  {  JPanel panelobj;  JTextArea textareaobj;  JScrollPane spaneobj;  public ScrollPanelTest()  {  super(" 文本区域＋滚动条演示 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(300,200);  panelobj = new JPanel();  getContentPane().add(panelobj);  textareaobj = new JTextArea(" 文本区域 ",5,20);  panelobj.add(textareaobj);  spaneobj = new JScrollPane(textareaobj,  JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS,  JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS);  panelobj.add(spaneobj);  }  public static void main(String args[])  {  ScrollPanelTest spt = new ScrollPanelTest();  spt.show();  }  }  **（七）JList 列表框类**  JList 类用于构造一个列表框，可以在其中选择一个或多个选项，其样式和文本区域类似。  JList 的构造函数以字符串数组作参数，行数和列数根据字符串数组的个数和最长字符串确定，这一点和 JTextArea 相似。在使用时通常也使用 JScrollPane 为其添加滚动条，以固定列表框的大小。  下面的代码用于在窗口中的面板里添加一个带滚动条的列表框。  import java.awt.\*;  import javax.swing.\*;  public class ListTest extends JFrame  {  JPanel panelobj;  JList listobj;  JScrollPane spaneobj;  ListTest()  {  super(" 列表框演示 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(300,150);  panelobj = new JPanel();  getContentPane().add(panelobj);  String liststr[] = {" 列表项 1"," 列表项 2"," 列表项 3"," 列表项 4"};  listobj = new JList(liststr);  listobj.setFixedCellWidth(240);  listobj.setVisibleRowCount(3);  panelobj.add(listobj);  spaneobj = new JScrollPane(listobj,  JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS,  JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS);  panelobj.add(spaneobj);  }  public static void main(String args[])  {  ListTest lt = new ListTest();  lt.show();  }  }  **（八）JCheckBox 复选框类**  JCheckBox 类用于构造一个复选框组件。复选框在界面应用中起到一个开关的作用，选择时复选框被勾选为“√”符号，再次选择即去掉“√”符号。界面中可以有多个复选框，且彼此的状态相互独立，不会产生影响。  下面的代码用于创建两个复选框并添加到界面中。  import java.awt.\*;  import javax.swing.\*;  public class CheckBoxTest extends JFrame  {  JPanel panelobj;  JCheckBox cbox1,cbox2;  public CheckBoxTest()  {  super(" 复选项演示 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(200,100);  panelobj = new JPanel();  getContentPane().add(panelobj);  Icon imageobj = new ImageIcon("c:\\a.jpg");  cbox1 = new JCheckBox(" 复选项 1",imageobj,false);  cbox2 = new JCheckBox(" 复选项 2",null,true);  panelobj.add(cbox1);  panelobj.add(cbox2);  }  public static void main(String args[])  {  CheckBoxTest cbt = new CheckBoxTest();  cbt.setVisible(true);  }  }  JCheckBox 类的构造函数有三个参数：第一个参数是复选框的文本标签；第二个参数是一个 Icon 对象，用于在复选框位置放置一个图像图标；  第三个参数指明复选框的选中状态，true 表示选中，false 表示未选中。  **（九）JRadioButton 单选按钮类**  JRadioButton 类用于构造一个单选按钮组件。单选按钮组件也叫单选框，用于多选一的情况。  下面的代码用于创建两个单选按钮并添加到界面中。  import java.awt.\*;  import javax.swing.\*;  public class RadioButtonTest extends JFrame  {  JPanel panelobj;  JRadioButton rbutton1,rbutton2;  public RadioButtonTest()  {  super(" 单选按钮演示 ");  setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(200,100);  panelobj = new JPanel();  getContentPane().add(panelobj);  rbutton1 = new JRadioButton(" 单选按钮 1",false);  rbutton2 = new JRadioButton(" 单选按钮 2",true);  panelobj.add(rbutton1);  panelobj.add(rbutton2);  ButtonGroup bgroupobj = new ButtonGroup();  bgroupobj.add(rbutton1);  bgroupobj.add(rbutton2);  }  public static void main(String args[])  {  RadioButtonTest rbt = new RadioButtonTest();  rbt.setVisible(true);  }  }  JRadioButton 类的构造函数有两个参数：第一个参数是单选按钮的文本标签；第二个参数是按钮的选中状态。  JRadioButton 类只是创建一个独立的单选按钮，但各个按钮之间没有任何关系，这时的单选按钮不能实现多选一。只有创建一个 ButtonGroup（按钮组）类的对象，并使用add 方法将单选按钮添加到按钮组中，才能够在这多个按钮中实现多选一。  这里介绍的都是 javax.swing 包中的，以大写字母“J”开头的组件，又叫做 J 组件。几乎每一个 J 组件都能够在 java.awt 包中找到一个与之对应的组件，如 JButton 对应的Button，JTextField 对应的 TextField 等。多数 awt 组件的使用方法和 swing 组件的使用是相同的，只有少数组件的使用有一定差异，详细内容大家可以参考 SDK 技术文档和其他相关书籍。  **任务实施**  /\* 源程序文件名：Ex0701.java\*/  import java.awt.\*;  import javax.swing.\*;  import java.awt.event.\*;  public class Ex0701 extends JFrame implements ActionListener  {  JTextField answer;  JTextField input1, input2;  JButton ok;  Ex0701()  {  super("Java GUI 演示程序 ");  Container c = getContentPane();  c.setLayout(new FlowLayout());  input1 = new JTextField(4);  input2 = new JTextField(4);  answer = new JTextField(4);  ok = new JButton(" 计算 ");  c.add(new JLabel(" 请输入要求最大公因数的两个数："));  c.add(input1);  c.add(input2);  c.add(new JLabel(" 最公因数："));  c.add(answer);  c.add(ok);  ok.addActionListener(this);  }  public static void main(String args[])  {  Ex0701 ex = new Ex0701();  ex.addWindowListener(new WindowAdapter()  {  public void windowClosing(WindowEvent e)  {  System.exit(0);  }  });  ex.setSize(320,100);  ex.show();  }  public void actionPerformed(ActionEvent e)  {  int num1 = Integer.parseInt(input1.getText());  int num2 = Integer.parseInt(input2.getText());  if(num1<0) num1 = -num1;  if(num2<0) num2 = -num2;  if(num1<num2){  int temp = num1;  num1 = num2;  num2 = temp;  }  do  {  int x;  x = num2;  num2 = num1%num2;  num1 = x;  }  while(num2!=0);  answer.setText(String.valueOf(num1));  }  }  这个例子对于刚开始学习 GUI 编程的人来讲可能有一点难度，先分析一下它的结构：  import java.awt.\*;  import javax.swing.\*;  import java.awt.event.\*;  这里有三个导入语句，java.awt 是 GUI 编程的基础包，javax.swing 是用到的 Swing 组件包，java.awt.event 是响应用户事件的包。  public class Ex0701 extends JFrame implements ActionListener 公共类 Ex0701 继承自 JFrame 类，使之可以创建一个框架窗体。同时实现的ActionListener 接口，使其可以作为动作监听器使用，以监听用户操作。  JTextField answer;  answer = new JTextField(4);  这里定义了一个宽度为 4 的文本框组件，以作为数据的输入或输出显示。类似的有文本框 input1 和 input2，按钮 ok 等。  Container c = getContentPane();  这里定义了一个容器 c，使其为当前面板即 Ex0701 这个 JFrame。  c.setLayout(new FlowLayout());  这里定义当前容器 c 的布局方式为流布局。有关布局方式，将在下个任务中介绍。  c.add(new JLabel(" 请输入要求最大公因数的两个数："));  这里使用容器的 add 方法向容器中添加了一个 JLabel（文本标签）组件。同样的，我们使用 add 方法向容器中添加了文本框、按钮等多个组件。  ok.addActionListener(this);  这里在按钮 Ok 上注册了一个动作监听器，用于监听用户在该组件上的动作。由于实现了 ActionListener 接口，所以监听器由类 Ex0701 自身担任。  ex.addWindowListener(new WindowAdapter()  {  public void windowClosing(WindowEvent e)  {  System.exit(0);  }  });  在 main 方法中定义了 Ex0701 的一个实例 ex，上面的代码是为 ex 添加一个窗体监听器，其作用为当单击窗体的关闭按钮时，使程序正常退出。而后续的 setSize 方法和show 方法则分别定义窗体大小以及让窗体可见。  public void actionPerformed(ActionEvent e)  actionPerformed 方法是由 ActionListener 接口中提供的。由于实现了 ActionListener 接口，我们必须在类 Ex0701 中重写 actionPerformed 方法，以定义对用户操作的具体处理。在这里是接收用户在 input1 和 input2 中的输入并转换为整型量，然后用辗转相除法求得最大公因数，最后将结果在 answer 中显示。  在这个例子中，尝试了定义组件、定义容器、添加组件、显示窗体、监听和处理用户动作等 GUI 编程的基本操作，下面的任务中先给大家介绍图形界面组件的用法。  **【学生】**思考、讨论。 | **通过教师讲解，了解Java 图形用户界面程序（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了Java 图形用户界面程序（二），让学生知道说明文根据表达方式的不同，可以分为一般性说明文和文艺性说明文。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  简述JRadioButton 单选按钮类。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示界面布局（一）  **任务描述**  编写程序，使用流布局管理器放置多个组件。  **任务目标**  1. 能够掌握 Java 中常用的布局管理器类，以及设置页面的布局管理方式。  2. 能够掌握 Java 中常用布局管理器类的使用方法。  3. 具有“大局观”，做一个能掌控全局的人。  **预备知识**  在前面的例子中，如果向界面容器中添加多个组件时，组件将以加入的次序，按行一个一个地放置。当到达容器的右边界时，自动开始在下一行放置控件。这种布局方式就是流式布局。Java 中提供了各种布局管理器类来管理各种组件在容器中的放置状态。它会根据平台来调整组件的大小，具有很好的平台无关性。每一个容器有自己默认的布局管理器，例如，JFrame 类的默认布局管理器为 BorderLayout（边界布局管理器），JPanel 类的默认布局管理器为 FlowLayout（流布局管理器）。当创建一个容器时，Java 自动为它创建并分配一个默认的布局管理器，并可以在程序中为容器创建和应用新的布局管理器。  Java 中的布局管理器有：FlowLayout（流式布局管理器）、GridLayout（网格布局管理器）、BorderLayout（边界布局管理器）、CardLayout（卡片布局管理器）、BoxLayout（盒布局管理器）和 GridBayLayout（网格包布局管理器）等。我们将在后面向大家介绍几种常用的布局管理器。  **一、FlowLayout 流式布局**  默认状态下，FlowLayout 管理器使控件对准每一行的中心，其构造函数如下：FlowLayout()：创建一个流布局管理器 , 位置以中心对齐并在组件之间留以 5 像素的水平和垂直间隔 ;  FlowLayout(int align)：创建一个流布局管理器 , 位置按所指定的对齐方式 align 对齐使用指定的对齐方式 (FlowLayout.LEFT,FlowLayout.RIGHT,FlowLayout.Center), 水平和垂直间距为缺省值 5 像素 ;  FlowLayout(int align, int hgap, int vgap)：创建一个流布局管理器 , 位置按所指定的对齐方式 align 对齐并在组件之间留以 hgap 和 vgap 指定的水平和垂直间隔。  **二、GridLayout 网格布局**  GridLayout（网格布局管理器）是 FlowLayout 的扩展，GridLayout 把显示区域编组为矩形格子，然后把组件从左至右，从上到下地放入每个格子。网格布局管理器和流布局管理器不同的是它按指定的行列数自动换行。  在默认状态下，GridLayout 管理器使控件对准每一矩形格的中心。  GirdLayout 的构造函数如下：  GridLayout()：创建一个默认为 1 行的网格布局管理器 ; GridLayout(int rows, int cols)：创建一个指定行列数的网格布局管理器 ; GridLayout(int rows, int cols, int hgap, int vgap)：创建一个指定行列数和水平、垂直间距的网格布局管理器。  下面的代码是设置 JPanel 容器的网格布局管理器：  panelobj = new JPanel();  GridLayout layoutobj = new GridLayout(3,2,5,10);  panelobj.setLayout(layoutObj);  网格布局管理器的一个显著特点是其组件将占满容器的整个区域，当窗口大小改变时，组件的大小也会随着改变。  **【学生】**思考、讨论。 | **教师通过界面布局（一）展示，让学生了解界面布局的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了界面布局（一），让学生具有“大局观”，做一个能掌控全局的人。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述GridLayout 网格布局。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示界面布局（二）  **三、BorderLayout 边界布局**  JFrame 容器的默认布局管理器是 BorderLayout（边界布局管理器），其布局方式按上北（North）、下南（South）、左西（West）、右东（East）和中（Center）五个方位布局。  BorderLayout 的构造函数如下：  BorderLayout()：创建一个边界布局管理器 , 各组件间的水平和竖直间距为缺省值 0 个像素 ; BorderLayout(int hgap, int vgap)：以指定的水平和垂直间距创建一个边界布局管理器。  下面的代码演示了 BorderLayout 的使用：  import java.awt.\*;  import javax.swing.\*;  public class BorderLayoutTest extends JFrame  {  JPanel panelobj;  JButton bobj1,bobj2,bobj3,bobj4,bobj5;  public BorderLayoutTest()  {  super(" 流布局演示 ");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(260,160);  panelobj = new JPanel();  getContentPane().add(panelobj);  BorderLayout layoutobj = new BorderLayout(5,5);  panelobj.setLayout(layoutobj);  bobj1 = new JButton(" 按钮 1");  bobj2 = new JButton(" 按钮 2");  bobj3 = new JButton(" 按钮 3");  bobj4 = new JButton(" 按钮 4");  bobj5 = new JButton(" 按钮 5");  panelobj.add("North",bobj1);  panelobj.add("West",bobj2);  panelobj.add("South",bobj3);  panelobj.add("East",bobj4);  panelobj.add("Center",bobj5);  }  public static void main(String args[])  {  BorderLayoutTest blt = new BorderLayoutTest();  blt.show();  }  }  程序运行结果如图 7−12 所示。    在向应用了 BorderLayout 的容器中添加组件时，需要指定组件的位置，例如：  panelobj.add("North",bObj1);  这是将组件 bobj1 添加到 panelobj 中，并将其放到上方（North）。CardLayout( 卡片布局管理器 ) 将界面看作一系列的卡片 , 任意时刻只有其中之一可见。  CardLayout 的构造函数如下：  CardLayout()：创建一个卡片布局管理器 , 卡片各边和容器的水平和竖直间距为缺省值 0 个像素 ;  CardLayout(int hgap, int vgap)：以指定的水平和垂直间距创建一个卡片布局管理器。  下面的代码演示了 CardLayout 的使用。  import java.awt.BorderLayout;  import java.awt.CardLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JPanel;  import javax.swing.JTextArea;  public class TestCardLayout implements ActionListener  {  private JFrame fm = new JFrame(" 卡片布局演示 ");  private JPanel card,containbtn; //card 为卡片显示区域 ,containbtn 为按钮显示区域  private JButton btn1,btn2,btn3,btn4; // 此处定义诗人的按钮  private JPanel pan1,pan2,pan3,pan4; // 此处定义的面板中显示诗句  private CardLayout cardlayout; // 定义卡片布局  private JTextArea ta1,ta2,ta3,ta4; // 定义放诗句的文本域    public TestCardLayout()  {  // 以下为初始化组件  card = new JPanel();  containbtn = new JPanel();  cardlayout = new CardLayout();  card.setLayout(cardlayout);    btn1 = new JButton(" 毛泽东 ");  btn2 = new JButton(" 周恩来 ");  btn3 = new JButton(" 白居易 ");  btn4 = new JButton(" 孟浩然 ");    ta1 = new JTextArea(30,30);  ta2 = new JTextArea(30,30);  ta3 = new JTextArea(30,30);  ta4 = new JTextArea(30,30);  // 把定义的按钮放到显示按钮的面板中  containbtn.add(btn1);  containbtn.add(btn2);  containbtn.add(btn3);  containbtn.add(btn4);    // 给按钮添加监听器  btn1.addActionListener(this);  btn2.addActionListener(this);  btn3.addActionListener(this);  btn4.addActionListener(this);  // 给定义的文本域中添加诗人相应的诗句  ta1.append(" 钟山风雨起苍黄 ,\r\n");  ta1.append(" 百万雄师过大江。\r\n");  ta1.append(" 虎踞龙盘今胜昔 ,\r\n");  ta1.append(" 天翻地覆慨而慷。\r\n");  ta1.append(" 宜将剩勇追穷寇 ,\r\n");  ta1.append(" 不可沽名学霸王。\r\n");  ta1.append(" 天若有情天亦老 ,\r\n");  ta1.append(" 人间正道是沧桑。");    ta2.append(" 极目青郊外 ,\r\n");  ta2.append(" 烟霾布正浓。\r\n");  ta2.append(" 中原方逐鹿 ,\r\n");  ta2.append(" 博浪踵相踪。");    ta3.append(" 一道残阳铺水中 ,\r\n");  ta3.append(" 半江瑟瑟半江红。\r\n");  ta3.append(" 可怜九月初三夜 ,\r\n");  ta3.append(" 露似珍珠月似弓。");    ta4.append(" 春眠不觉晓 ,\r\n");  ta4.append(" 处处闻啼鸟。\r\n");  ta4.append(" 夜来风雨声 ,\r\n");  ta4.append(" 花落知多少。");    pan1 = new JPanel();  pan1.add(ta1);  pan2 = new JPanel();  pan2.add(ta2);  pan3 = new JPanel();  pan3.add(ta3);  pan4 = new JPanel();  pan4.add(ta4);  // 把定义好的 4 个面板放到卡片布局管理器中  card.add("b1",pan1);  card.add("b2",pan2);  card.add("b3",pan3);  card.add("b4",pan4);    fm.add(containbtn,BorderLayout.NORTH); // 把显示按钮的组件放到上面显示  fm.add(card,BorderLayout.CENTER); // 把显示文本域的组件放到中间显示    fm.setSize(400,400);  fm.setLocation(200,200);  fm.setVisible(true);  fm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }    public static void main(String[] args)  {  new TestCardLayout();  }  public void actionPerformed(ActionEvent e)  {  /\*\*  \* e.getSource(); 返回值是点击按钮的名字  \* cardlayout.show(card, "b1"); 表示对应组件和显示组件的名称  \* CardLayout.show(Container parent, String name)：显示卡片的实现方法定义  \*/  if(e.getSource()==btn1)  {  cardlayout.show(card, "b1");  }  if(e.getSource()==btn2)  {  cardlayout.show(card, "b2");  }  if(e.getSource()==btn3)  {  cardlayout.show(card, "b3");  }  if(e.getSource()==btn4)  {  cardlayout.show(card, "b4");  }  }  }  程序运行结果如图 7−13 所示。    向应用了 CardLayout 的容器中添加组件 pan1 并命名”b1”，如：  card.add("b1",pan1);  这是将组件 bobj1 添加到 panelobj 中，并将其放到上方（North）。在实际使用的时候，通常不可能仅仅使用一种布局管理器就可以创建一个美观、漂亮的界面。通常的做法是，在框架中指定一种布局管理器，并在其中添加多个面板，而每个面板又分别应用不同的布局管理器，最后向面板中添加组件。有时甚至要用到多重的面板，以此来实现在界面中完成复杂多变的布局。  **任务实施**  /\* 源程序文件名：Ex0702.java\*/  import java.awt.\*;  import javax.swing.\*;  public class Ex0702 extends JFrame  {  JPanel panelobj;  JButton bobj1,bobj2,bobj3,bobj4,bobj5;  public Ex0702()  {  super(" 布局管理器演示 ");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(260,120);  panelobj = new JPanel();  getContentPane().add(panelobj);  FlowLayout layoutobj = new FlowLayout(FlowLayout.LEFT,10,10);  panelobj.setLayout(layoutobj);  bobj1 = new JButton(" 按钮 1");  bobj2 = new JButton(" 按钮 2");  bobj3 = new JButton(" 按钮 3");  bobj4 = new JButton(" 按钮 4");  bobj5 = new JButton(" 按钮 5");  panelobj.add(bobj1);  panelobj.add(bobj2);  panelobj.add(bobj3);  panelobj.add(bobj4);  panelobj.add(bobj5);  }  public static void main(String args[])  {  FlowFrameTest fft = new FlowFrameTest();  fft.show();  }  }  在本例中：  FlowLayout layoutobj = new FlowLayout (FlowLayout.LEFT,10,10);  一句用于创建一个流布局管理器，第一个参数指定对齐方式为左对齐，第二、三个参数指定组件间的水平和垂直间隔。  panelobj.setLayout(layoutobj);  这一句调用容器的 setLayout 方法，将布局方式应用到容器中。流布局的特点是组件位置会根据容器的宽度变化而发生变化。在上面的运行结果中，可以通过拖动窗口边缘增加宽度，观察界面中组件排列的变化。  **【学生】**思考、讨论。 | **教师通过界面布局（二）展示，让学生了解界面布局（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了界面布局（二），让学生了解流布局的特点是组件位置会根据容器的宽度变化而发生变化。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述BorderLayout 边界布局。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示事件监听  **任务描述**  编写程序，通过本金和利率计算利息，使用焦点监听器（FocusListener）和动作监听器（ActionListener）。  **任务目标**  1. 能够理解 Java 的事件处理机制。  2. 能够掌握 Java 中事件响应和处理的方法。  **预备知识**  一般情况下，组件（事件源）都不处理自己的事件，而是将事件处理委托给外部的处理实体（监听器），这种事件处理模型称为事件的授权处理模型。不同的事件，可以交由不同类型的监听器去处理。具体的事件监听处理工作流程如图 7−15 所示。    **一、监听器接口：XxxxListener**  当用户按下键盘或单击鼠标按钮，使用户和应用程序交互时，就产生一个事件，操作系统自动捕获这种事件和与它有联系的数据，然后将这种数据以事件对象的形式传到该事件所属的事件处理程序。  事件对象包括事件的种类（如按键、单击鼠标、移动鼠标、关闭窗口等）、产生事件的组件（如按钮、框架、列表等）和产生事件的事件。  事件处理程序是理解和处理事件的一种方法，即实现在接口中定义的方法，事件处理程序的方法以事件对象为参数。  所有的事件类和接口都放置在 java.awt.event 包中。  常见的事件类型有：  ItemEvent：项目事件，如选中某个项目；  AdjustmentEvent：调节事件，如移动滚动条；  ActionEvent：动作事件，如按下按钮；  TextEvent：文本事件，如改变文本对象；  ComponentEvent：组件事件，如改变组件大小、移动组件、显示或隐藏组件等；  FocusEvent：焦点事件，如组件得到或失去焦点；  ContainerEvent：容器事件，如组件的增加和删除；  PaintEvent：绘画事件，如在组件上绘画；  WindowEvent：窗口事件，如关闭窗口、最小化、最大化、还原窗口等；  KeyEvent：键盘事件，如按下、释放键盘上的键；  MouseEvent：鼠标事件，如按下、释放、单击、移动、拖动鼠标。处理这些事件的接口称为事件监听器，名称形如 XxxxListener，其中 Xxxx 代表事件类型，首字母大写。  使用监听器实现事件处理相应的一般做法如下。①定义一个实现监听器接口的类。如本任务中的类 Ex0703，实现了 FocusListener 和 ActionListener 接口，这个类就可作监听器使用。②重写接口中的方法。针对 FocusListener 接口，需要重写 focusGained 和 focusLost 两个方法，即使我们没有使用其中的某些方法。③注册监听器。在要监听的组件上使用 addXxxxListener 方法注册监听器，这样在该组件上的操作才能被正常捕获。  **二、适配器类：XxxxAdapter**  在使用监听器监听事件时，需要重写（或覆盖）接口中的所有方法，而不管实际需要使用多少个，这样会导致一些无用代码的出现，增加代码长度。  为了解决这个问题，Java 中为一些 Listener 接口提供了相应的 Adapter（适配器）类，可以通过继承事件所对应的 Adapter 类，重写其中需要的方法，对于不需要的方法则可以省略，这样可以精简代码。但由于 Java 的单一继承机制，当需要使用多个适配器或此类已有父类时，这种方式就无法使用了。  Adapter 类中的方法和相应 Listener 接口中的方法一致，常用的 Adapter 类有：  ComponentAdapter：组件适配器 ;  FocusAdapter：焦点适配器 ;  ContainerAdapter：容器适配器 ;  WindowAdapter：窗口适配器 ;  KeyAdapter：键盘适配器 ;  MouseAdapter：鼠标适配器 ;  MouseMotionAdapter：鼠标运动适配器 ;  任务实施  (1)/\* 源程序文件名：Ex0703.java\*/  import java.awt.\*;  import javax.swing.\*;  import java.awt.event.\*;  public class Ex0703 extends JFrame implements FocusListener,  ActionListener  {  JPanel pobj1,pobj2;  JTextField tfobj[] = new JTextField[5];  JButton bobj1,bobj2;  public Ex0703()  {  super(" 事件处理 ");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(350,150);  pobj1 = new JPanel();  pobj2 = new JPanel();  getContentPane().add("North",pobj1);  getContentPane().add("South",pobj2);  pobj1.setLayout(new GridLayout(2,5,5,5));  pobj1.add(new JLabel(" 本金："));  pobj1.add(new JLabel(" 利率："));  pobj1.add(new JLabel(" 存期："));  pobj1.add(new JLabel(" 年利："));  pobj1.add(new JLabel(" 总利："));  for(int i = 0; i<5; i++)  {  tfobj[i] = new JTextField("0",5);  pobj1.add(tfobj[i]);  tfobj[i].addFocusListener(this);  }  bobj1 = new JButton(" 重置 ");  bobj1.addActionListener(this);  bobj2 = new JButton(" 退出 ");  bobj2.addActionListener(this);  pobj2.add(bobj1);  pobj2.add(bobj2);  }  public void focusGained(FocusEvent fe)  {  }  public void focusLost(FocusEvent fe)  {  double intf[] = new double[3];  double outf1 = 0,outf2 = 0;  for(int i = 0; i<3; i++)  {  intf[i] = Double.parseDouble (tfobj[i].getText());  }  outf1 = intf[0]\*intf[1];  outf2 = outf1\*intf[2];  tfobj[3].setText(Double.toString(outf1));  tfobj[4].setText(Double.toString(outf2));  }  public void actionPerformed(ActionEvent ae)  {  if(ae.getActionCommand().equals(" 重置 "))  {  for(int i = 0; i<5; i++)  {  tfobj[i].setText("0");  }  }  else  {  System.exit(0);  }  }  public static void main(String args[])  {  Ex0703 fat = new Ex0703();  fat.setvisible(true);  }  }  在这个例子中，使用到了两种监听器：FocusListener 和 ActionListener，分别用于监听文本框的焦点事件（得到或失去焦点）和按钮的动作事件（单击操作）。  主类中实现了 FocusListener 和 ActionListener 接口，则需要在类中重写接口中的所有方法。对于 FocusListener 接口是 focusGained（FocusEvente）和 focusLost（FocusEvente）方法，分别用于处理组件得到焦点和失去焦点时的操作。对于 ActionListener 接口则是actionPerformed（ActionEvente）方法。  上例中，定义了 focusLost 方法，即在组件失去焦点时，通过本金、利率和存期三个文本框内的值，计算年利和总利，并显示在相应的文本框中。另外还有 actionPerformed 方法，用于判断用户是单击的哪一个按钮，从而执行相应的操作。而 FocusListener 接口中还有一个 focusGained 方法，虽然没有使用，但仍需要重写，只是方法体为空。  (2)/\* 源程序文件名：Ex0704.java\*/  下面代码是将程序 Ex0703.java 改为由 Adapter 类实现：  import java.awt.\*;  import javax.swing.\*;  import java.awt.event.\*;  public class Ex0704 extends JFrame implements ActionListener  {  class FAdapter extends FocusAdapter  {  public void focusLost(FocusEvent fe)  {  double intf[] = new double[3];  double outf1 = 0,outf2 = 0;  for(int i = 0; i<3; i++)  {  intf[i] = Double.parseDouble (tfobj[i].getText());  }  outf1 = intf[0]\*intf[1];  outf2 = outf1\*intf[2];  tfobj[3].setText(Double.toString(outf1));  tfobj[4].setText(Double.toString(outf2));  }  }  JPanel pobj1,pobj2;  JTextField tfobj[] = new JTextField[5];  JButton bobj1,bobj2;  public AdapterTest()  {  super(" 事件处理 ");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setVisible(true);  setSize(350,150);  pobj1 = new JPanel();  pobj2 = new JPanel();  getContentPane().add("North",pobj1);  getContentPane().add("South",pobj2);  pobj1.setLayout(new GridLayout(2,5,5,5));  pobj1.add(new JLabel(" 本金："));  pobj1.add(new JLabel(" 利率："));  pobj1.add(new JLabel(" 存期："));  pobj1.add(new JLabel(" 年利："));  pobj1.add(new JLabel(" 总利："));  FAdapter fa = new FAdapter();  for(int i = 0; i<5; i++)  {  tfobj[i] = new JTextField("0",5);  pobj1.add(tfobj[i]);  tfobj[i].addFocusListener(fa);  }  bobj1 = new JButton(" 重置 ");  bobj1.addActionListener(this);  bobj2 = new JButton(" 退出 ");  bobj2.addActionListener(this);  pobj2.add(bobj1);  pobj2.add(bobj2);  }  public void actionPerformed(ActionEvent ae)  {  if(ae.getActionCommand().equals(" 重置 "))  {  for(int i = 0; i<5; i++)  {  tfobj[i].setText("0");  }  }  else  {  System.exit(0);  }  }  public static void main(String args[])  {  Ex0704 ex = new Ex0704();  ex.setVisible(true);  }  }  其中，定义了一个内部类 FAdapter，继承自 FocusAdapter 类，用来监听文本框的焦点事件。由于我们只使用捕获失去焦点的事件，所以在这里只重写（或覆盖）了 focusLost方法，没有重写 focusGained 方法。程序的运行结果和程序 Ex0703.java 的运行结果一样。  **【学生】**思考、讨论。 | **教师通过事件监听展示，让学生了解事件监听的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了事件监听，能够掌握 Java 中事件响应和处理的方法。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **思考一下，Java 的事件处理机制是什么？** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示Applet 中的事件处理  任务描述  编写 Applet 程序，捕获鼠标单击的坐标，并在相应位置上显示。  任务目标  能够掌握 Applet 中事件的响应和处理方法。  预备知识  在 Applet 中常会使用到鼠标和键盘事件，由于 Applet 类中已经定义了一些事件处  理方法，所以不需要实现监听器接口或使用适配器类。  一、鼠标事件  在 Applet 中定义的鼠标事件如下。  mouseDown：当鼠标按键被按下时产生。由同名的 mouseDown 方法处理，其原型  为 public boolean mouseDown（Event e, int x, int y）。其中的参数 e 是事件本身，x 和 y 是  事件发生时的鼠标坐标。  mouseUp：当鼠标按键释放时产生。由同名的 mouseUp 方法处理，其原型为 public  boolean mouseUp（Event e, int x, int y）。其中的参数含义和 mouseDown 方法的一样。  mouseMove：当鼠标移动时产生。由同名的 mouseMove 方法处理，其原型为 public  boolean mouseMove（Event e, int x, int y）。其中的参数含义和 mouseDown 方法一样。  mouseDrag：当鼠标拖动时产生。由同名的 mouseDrag 方法处理，其原型为 public  boolean mouseDrag（Event e, int x, int y）。其中的参数含义和 mouseDown 方法一样。  mouseEnter：当鼠标进入某个范围时产生。由同名的 mouseEnter 方法处理，其原型为 public boolean mouseEnter（Event e, int x, int y）。其中的参数含义和 mouseDown 方法  一样。  mouseExit：当鼠标移出某个范围时产生。由同名的 mouseExit 方法处理，其原型  为 public boolean mouseExit（Event e, int x, int y）。其中的参数含义和 mouseDown 方法  一样。  二、键盘事件  和鼠标比起来，键盘产生的事件就简单多了。一般来说，我们关心的就是用户按下  了哪一个按键，然后进行相应的处理，而不必像鼠标那样需要考虑按键时的坐标。  Applet 中只有两个键盘事件处理程序，keyDown 和 keyUp，相应的处理程序的原  型是：  public boolean keyDown(Event e, int key)  public boolean keyUp(Event e, int key)  分别用于处理按下和释放按键的事件。其中的参数 e 是事件本身，key 是按下的键。  对于字母和符号按键，key 的值就是相应的字符，而对于一些特殊按键，则在 Event 类  中定义的部分键值如表 7−3 所示：    下面的代码演示了在 Applet 中处理键盘事件。  import java.awt.\*;  import java.applet.\*;  public class KeyEvent extends Applet  {  char presskey = 0;  int x = 5, y = 15;  public boolean keyDown(Event e, int key)  {  switch(key)  {  case Event.UP:y--;break;  case Event.DOWN:y++;break;  case Event.LEFT:x--;break;  case Event.RIGHT:x++;break;  case Event.HOME:x = 5;y = 15;break;  default:presskey = (char)key;  }  repaint();  return true;  }  public void paint(Graphics g)  {  if(presskey!=0)  g.drawString(" 你按下的是 "+presskey+" 键 ",x,y);  }  }  任务实施  /\* 源程序文件名：Ex0705.java\*/  import java.applet.\*;  import java.awt.\*;  public class Ex0705 extends Applet  {  int mx = 50, my = 50;  public boolean mouseDown(Event e, int x, int y)  {  mx = x;  my = y;  repaint();  return true;  }  public void paint(Graphics g)  {  g.drawString(" 坐标：["+mx+","+my+"]",mx,my);  }  }  由于 Applet 中定义了一些鼠标和键盘事件，所以继承自 Applet 的类都可以重写（或  覆盖）这些方法以实现这些事件的相应处理。  上例中的 public boolean mouseDown（Event e, int x, int y）就是重写的 Applet 类中  的相应方法，用于捕获鼠标单击事件，并将单击时的坐标作为参数 x 和 y 传递给方法，  以进行相应的处理。需要注意的是，该方法的原型中定义了返回值类型为 boolean，在  重写时需要定义相同的方法返回值，并在方法内使用 return 语句返回一个 boolean 值。  **【学生】**思考、讨论。 | **教师通过Applet 中的事件处理展示，让学生了解Applet 中的事件处理的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了Applet 中的事件处理，能够掌握 Applet 中事件的响应和处理方法。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **怎样在 Applet 程序中实现动作响应？** | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 加强学生学风的培养，进一步加强学生对基础知识的掌握，特别是基础知识、基本概念的掌握与应用。 | |