广东外语外贸大学南国商学院

全日制本科课程教案

2018～2019学年 第一学期

课 程 名 称 Java语言程序设计

课 程 性 质 必修课

教 材 名 称 Java语言程序设计

适用专业（方向） 软件工程专业

学生年级、人数 16级、36人（1个班）

开 课 单 位 信息科学技术学院

授课教师姓名、职称 戴长秀、中级

教 务 处 制

二0一八年九月

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第1章Java语言概述 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第一周 |
| 教学内容：   1. 了解计算机语言的发展； 2. 掌握Java语言的特点； 3. 掌握JDK的安装与配置。   基本内容：  前言：   * Java语言是一门跨平台的热门语言 。 * Java语言可以一次编写，多种平台应用。 * 在国外，Java语言成为计算机世界的主流语言 。 * 在国内，计算机等级增加了Java等级考试，所以Java语言成为程序员需要掌握的一门语言。   跨平台:平台无关性，源程序（编译或解释）---->机器语言才能执行，因此，运行源程序的机器上必须装有该种语言的编译程序或解释程序，但编译程序或解释程序和机器的硬件结构，指令系统及所装的操作系统密切相关，这就使得通常的高级语言都是面向单机的—编程语言及其编译或解释程序都和特定的机型及所配的操作系统绑定了。  计算机网络的发展和普及提出了程序设计语言跨平台的要求，在网络上，程序代码可以轻松的由此及彼传送，但互联网上的计算机是异构的，计算机硬件类型和操作系统都存在差异，需要一种跨平台的编程语言，使得它编写的程序能够在网络中各种计算机上运行。针对每一机型及所配的操作系统，都配上该语言的编译或解释程序，这种语言写的程序就能在网络上的通行啦，但是此种办法不现实。因为源程序到机器语言程序的编译或解释工作量大，耗内存，机型多操作系统类型多，软硬件成本都大。  Java语言的设计都们想出了一个好办法，就是在原来的机器操作系统平台之上再搭建一个新平台，为这个平台设计一了部假想的计算机，假想的计算机与真实的计算机一样，CPU等五脏俱全，并针对该假想的计算机编制了一套机器语言，把用这种机器语言写出的程序代码称之为字节码，字节码的特点是在任一种机型及所配操作系统之下都很容易解释为本地机的机器语言，而且只占用很小的内存，新平台的主要若能就是把字节码解释为本地的机器语言。  于是，Java语言被设计成容易被编译为字节码文件的语言，为Java语言配置的编译程序只是把Java源程序编译为字节码文件，把字节码文件发送到网络上，于是网络上的计算机，只要搭建了这个新平台，即配备了把字节码翻译为本地机器语言的解释程序，就能运行发送来的Java语言程序。虽然解释程序也是和本地机及所配置的操作系统捆绑在一起，但是对不同的机型及操作系统配置不同的解释器难度小，成本降低了，这样也就解决了跨平台问题。Java语言同C语言一样有丰富的类库供用户调用，类库由压缩的字节码文件组成，成为新平台的一部分。  我们把上述的新平台称之为Java虚拟机，JVM主要由字节码解释程序和类库组成。规定Java语言源程序的扩展名为Java,编译后生成的字节码文件扩展名为Class,Java语言的跨平台性不是在源代码级实现的，而是目标代码级（Class文件）实现的。Java语言的跨平台性就是指Java语言的执行程序即Class文件可以不经修改而直接在网络上装有虚拟机的计算机上执行。  1.1Java语言简介  Java语言，是由原Sun Microsystems公司于1995年5月推出的Java程序设计语言和Java平台的总称。  Java分为三种平台：  Java SE(Java Platform Standard Edition，Java标准版平台)  Java EE(Java Platform Enterprise Edition，Java企业版平台)  Java ME(Java Platform Micro Edition，Java微型版平台)。  1.1.1Java语言的发展  Java语言起源于1991年，是原SUN公司为一些智能消费性电子产品设计的一个通用语言（Oak）。Oak语言是Java的前身。项目最初的目的只是为了开发一种独立于平台的软件技术，而且在网络出现之前，Oak可以说是默默无闻，甚至差点夭折。但是，网络的出现改变了Oak的命运。  为了促销和法律的原因，1995年Oak更名为Java。很快Java被工业界认可，许多大公司如IBM、Microsoft、DEC等购买了Java的使用权，并被美国杂志PC Magazine评为1995年十大优秀科技产品。  Java发展历程中的重大事件：  1995年5月，java语言诞生  1996年1月，第1个java语言开发环境JDK1.0诞生。  1996年4月，10个最主要的操作系统供应商声明将在其产品中嵌入java技术。  1996年9月，约8.3万个网页应用了java技术来制作。  ….  2009年4月20日，甲骨文宣布收购SUN。该交易价值约为74亿美元。  2010年1月21日，甲骨文宣布正式完成对SUN公司的收购。  原SUN公司宣布，2010年9月，JDK1.7发布。但由于收购原因，JDK1.7推迟发布。  1.1.2Java语言的特点  SUN对Java的定义是：Java是一种简单的、面向对象的、分布式的、健壮的、安全的、体系结构中立的、可移植的、高性能的、多线程的动态语言。  1.Java语言是简单的  Java去掉了C和C++中的多继承、指针和内存管理等。Java提供了自动的“垃圾”回收机制，使得程序员不必为内存管理而担忧。Java的简单性减低学习的难度。  2.Java语言是面向对象的  Java语言编程更符合人们的思维习惯。  3.Java语言是分布式的  Java从诞生就与网络联系在一起，Java的主要优点是面向网络的编程，Java支持网络应用程序的编程，使其成为一种分布式程序设计语言。  4.Java语言是健壮的  健壮性又称为稳定性。  编译和执行过程中进行严格的语法检查，以减少错误的发生。  利用自动“垃圾”回收机制管理内存，防止程序员在管理内存时产生错误。  使用异常处理机制捕获并响应异常情况，从而使程序在发生异常的时候能够继续运行。  5.Java语言是安全的  Java通常被用在网络环境中。为此，Java提供了一个安全机制以防止恶意代码的攻击。  6.Java语言是体系结构独立的  Java源代码不会针对一个特定平台进行编译，而是被转换成一种中间格式（字节码），字节码与体系结构无关，可以在任何有Java虚拟机（JVM）的计算机上运行，而Java虚拟机与平台相关.  7.Java语言是可移植的  这种可移植性来源于体系结构独立性，另外，Java语言设计目标就是让程序不用修改就可以在任何平台上运行。  8.Java语言是解释型的  Java程序在Java平台上被编译为字节码格式， 然后可以在安装Java平台的任何系统中运行。在运行时，Java平台中的Java解释器对这些字节码进行解释执行，执行过程中需要的类在连接阶段被载入到运行环境中。  这是Java语言的一个缺点，因为解释执行的语言一般会比编译执行的语言（如C和C++）执行效率低。  9.Java是高性能的  为了提高运行速度，Java语言提供一种即时编译JIT(Just-In-Time)编译器技术，随着JIT编译器技术的发展，Java程序执行速度越来越接近于C++。  10.Java语言是多线程的  多线程是指在一个程序中可以运行多个任务。C和C++采用单线程体系结构，而Java语言支持多线程技术。采用多线程机制能够提高程序运行效率，充分发挥硬件资源，但同时也增加了程序的设计难度。  11.Java语言是动态的  Java语言的设计目标之一是适应于动态变化的环境。Java程序需要的类能够动态地被载入到运行环境，也可以通过网络来载入所需要的类，这也有利于软件的升级。  1.2JDK的安装与配置  1．JDK简介与下载  JDK是一个可以编译、调试、运行Java应用程序或者Applet小应用程序的开发环境。它包括一个处于操作系统层之上的运行环境以及开发者编译、调试和运行Java程序的工具。自从Java推出以来，JDK已经成为使用最广泛的Java SDK。  Java SE的JDK可以在http://www.oracle.com/technetwork/java/javase/downloads/index.html网站下载。本书下载的是JDK7.0。  2．JDK的配置：  右击我的电脑，选择属性，在弹出的系统属性对话框中选择高级选项卡，单击环境变量按钮进行设置：  Path:在其值前添加“C:\program\java\jdk1.6.0\_20\bin”，C:\program\java是JDK安装的路径，也是默认安装路径。Java平台提供的可执行文件都放在bin包内，配置好path变量后，javac、java等命令编译或执行java应用程序时，就能够直接找到  classpath:“.; C:\program\java\jdk1.6.0\_20\lib”.代表当前路径，lib是JDK类库的路径。所需要的可执行文件。  JAVA\_HOME：“C:\program\java\jdk1.6.0\_20”，方便引用，引用路径时只需要输入“%JAVA\_HOME%”即可。  测试JDK配置是否成功，单击运行输入CMD，进行任意目录输入Javac按回车键，出现如下图所示的信息即表示配置成功。    练习：开发第一个Java程序  功能简介：Application程序例子。输出三行字符串。  \*/  /\*\* FirstJava类是一个Application程序。\*/  public class FirstJava  {  public static void main(String args[])  {  System.out.println("欢迎学习Java程序！");  System.out.println("一分耕耘，一分收获！");  System.out.println(“我将成为一名优秀的Java程序员！");  }//main()方法结束  }//类FirstJava结束    重、难、疑点：  重点：  JDK的安装与配置  难点：  无 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业：  实验1 开发第一个Java程序 | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第九章GUI | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第一周 |
| 教学内容：  掌握Java应用程序的开发  基本内容：  1.3Java程序的工作原理  1、Java程序的工作原理可用下图表示。    2、运行Java程序的基本过程：  （1）利用编辑器编写 Java源程序 源文件名：主类名.java  （2）利用编译器将源程序编译成字节码 字节码文件名：源文件名.class  （3）利用虚拟机（解释器）运行  工作过程：载入、代码校验、解释执行  1.4Java应用程序  1、Java程序有两种形式：  Java Applet（小应用程序）：不是一个完整的应用程序，而是框架程序中的一个模块，所以只能在WWW 浏览器环境下运行。  Java Application（Java应用程序）：是一个完整的应用程序，可以独立地运行。  开发Application（应用程序）主要有3步：  编辑：使用编辑器编写java源代码，.java文件。  编译：编译器译成字节码，.class文件  2、Java应用程序运行过程  （1）编辑Java源程序  （2）编译Java源程序  （3）运行Java应用程序  案例：  【例1-1】          重、难、疑点：  重点：Java应用程序的开发  难点：Java应用程序的开发 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第2章Java编程基础 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第二周 |
| 教学内容：   1. 了解Java代码的基本格式 2. 了解注释及应用 3. 了解标识符的命名、关键字的含义，转义字符的含义 4. 掌握常量、变量的应用   基本内容：  2.1 Java的基本语法  2.1.1 Java代码的基本格式  Java中的程序代码都必须放在一个类中。类需要使用class关键字定义，在class前面可以有一些修饰符，格式如下：  屏幕剪辑  Java代码关键点   1. 1、Java中的程序代码可分为结构定义语句和功能执行语句，其中，结构定义语句用于声明一个类或方法， 2. 功能执行语句用于实现具体的功能。每条功能执行语句的最后都必须用分号(;)结束。   屏幕剪辑  2、Java语言是严格区分大小写的。  3、虽然Java没有严格要求用什么样的格式来编排程序代码，但是，出于可读性的考虑，应该让自己编写的程序代码整齐美观、层次清晰。以下两种方式都可以，但是建议使用后一种。  屏幕剪辑  屏幕剪辑   1. 4、Java程序中一句连续的字符串不能分开在两行中书写，例如，下面这条语句在编译时将会出错： 2. 屏幕剪辑   如果为了便于阅读，想将一个太长的字符串分在两行中书写，可以先将这个字符串分成两个字符串，然后用加号（+）将这两个字符串连起来，在加号（+）处断行，上面的语句可以修改成如下形式  屏幕剪辑  2.1.2 Java中的注释  在编写程序时，为了使代码易于阅读，通常会在实现功能的同时为代码加一些注释。注释是对程序的某个功能或者某行代码的解释说明，它只在Java源文件中有效，在编译程序时编译器会忽略这些注释信息，不会将其编译到class字节码文件中去。  Java中的注释有三种类型，具体如下：  单行注释  单行注释通常用于对程序中的某一行代码进行解释，用符号“//”表示，“//”后面为被注释的内容，具体示例如下：  屏幕剪辑  多行注释  多行注释顾名思义就是在注释中的内容可以为多行，它以符号“/\*”开头，以符号“\*/”结尾，多行注释具体示例如下：  屏幕剪辑  文档注释，类似于说明书  档注释是以“/\*\*”开头，并在注释内容末尾以“\*/”结束。文档注释是对代码的解释说明，可以使用javadoc命令将文档注释提取出来生成帮助文档.html，关于这点将在后面的章节做详细讲解。  屏幕剪辑  2.1.3 Java中的标识符  在编程过程中，经常需要在程序中定义一些符号来标记一些名称，如，包名、类名、方法名、参数名、变量名等，这些符号被称为标识符。  标识符可以由任意顺序的大小写字母、数字、下划线（\_）和美元符号（$）组成，但不能以数字开头，不能是Java中的关键字。  建议初学者在定义标识符时还应该遵循以下规则：  1、包名所有字母一律小写。例如：cn.itcast.test。  2、类名和接口名每个单词的首字母都要大写。如：ArrayList、Iterator。  3、常量名所有字母都大写，单词之间用下划线连接。例如：DAY\_OF\_MONTH。  4、变量名和方法名的第一个单词首字母小写，从第二个单词开始每个单词首字母大写。例如：lineNumber、getLineNumber。  5、在程序中，应该尽量使用有意义的英文单词来定义标识符，使得程序便于阅读。例如使用userName表示用户名，password表示密码。  2.1.4 Java中的关键字  关键字是编程语言里事先定义好并赋予了特殊含义的单词，也称作保留字。和其它语言一样，Java中保留了许多关键字，例如，class、public等，下面列举的是Java中所有的关键字。  屏幕剪辑  2.1.5 Java中的常量  常量就是在程序中固定不变的值，是不能改变的数据。  在Java中，常量包括整型常量、浮点数常量、布尔常量、字符常量等。  特殊字符——反斜杠（\）  反斜杠（\）是一个特殊的字符，被称为转义字符，它的作用是用来转义后面一个字符。下面列出一些常见的转义字符：  \r 表示回车，将光标定位到当前行的开头，不会跳到下一行。  \n 表示换行，换到下一行的开头。  \t 表示制表符，将光标移到下一个制表符的位置，就像在文档中用Tab键一样。  \b 表示退格符号，就像键盘上的Backspace。  刚才介绍的字符都有特殊意义，无法直接表示，所以用斜杠加上另外一个字符来表示。  \' 表示单引号字符，Java代码中单引号表示字符的开始和结束，如果直接写单引号字符（'）,程序会认为前两个是一对，会报错，因此需要使用转义（\'）。  \" 表示双引号字符，Java代码中双引号表示字符串的开始和结束，包含在字符串中的双引号需要转义，比如"he says,\"thank you\"."。  \\ 表示反斜杠字符，由于在Java代码中的斜杠\是转义字符，因此需要表示字面意义上的\，就需要使用双斜杠\\。  2.2.1 变量的定义  变量就是数据存储空间的表示。  在程序运行期间，随时可能产生一些临时数据，应用程序会将这些数据保存在一些内存单元中，每个内存单元都用一个标识符来标识。这些内存单元我们称之为变量，定义的标识符就是变量名，内存单元中存储的数据就是变量的值。  2.2.2 变量的数据类型  变量要先声明才能使用，声明变量的目的以便给变量分配内存空间。Java是一门强类型的编程语言，它对变量的数据类型有严格的限定。在定义变量时必须声明变量的类型，在为变量赋值时必须赋予和变量同一种类型的值，否则程序会报错。数据类型所占内存空间大小固定，与平台无关性。  屏幕剪辑  2.2.3 变量的类型转换  当把一种数据类型的值赋给另一种数据类型的变量时，需要进行数据类型转换。  根据转换方式的不同，数据类型转换可分为两种：自动类型转换和强制类型转换。  2.2.4 变量的作用域  变量需要在它的作用范围内才可以被使用，这个作用范围称为变量的作用域。在程序中，变量一定会被定义在某一对大括号中，该大括号所包含的代码区域便是这个变量的作用域。  2.3 Java中的运算符  表达式：  屏幕剪辑  运算符用于对数据进行算术运算、赋值和比较等操作。  在Java中，运算符可分为算术运算符、赋值运算符、比较运算符、逻辑运算符和位运算符。  2.3.1 算数运算符  屏幕剪辑  2.3.1 算数运算符  接下来就针对其中比较重要的几点进行详细地讲解，具体如下：  1、在进行自增（++）和自减（--）的运算时，如果运算符（++或--）放在操作数的前面则是先进行自增或自减运算，再进行其它运算。反之，如果运算符放在操作数的后面则是先进行其它运算再进行自增或自减运算。  2、在进行除法运算时，当除数和被除数都为整数时，得到的结果也是一个整数。如果除法运算有小数参与，得到的结果会是一个小数。  3、在进行取模（%）运算时，运算结果的正负取决于被模数(%左边的数)的符号，与模数(%右边的数)的符号无关。如：(-5)%3=-2，而5%(-3)=2。  2.3.2 赋值运算符  赋值运算符的作用就是将常量、变量或表达式的值赋给某一个变量。  在赋值运算符的使用中，需要注意以下几个问题：  1、在Java中可以通过一条赋值语句对多个变量进行赋值。  在上述代码中，一条赋值语句将变量x，y，z的值同时赋值为5。需要特别注意的是，下面的这种写法在Java中是不可以的。  2、除了“=”，其它的都是特殊的赋值运算符，以“+=”为例，x += 3就相当于x = x + 3，首先会进行加法运算x+3，再将运算结果赋值给变量x。-=、\*=、/=、%=赋值运算符都可依此类推。  课堂练习：  任意输入一个三位的正整数，计算该数据的百位、十位及个位上的数字，并输出计算结果。  2.3.3 比较运算符  比较运算符用于对两个数值或变量进行比较，其结果是一个布尔值，即true或false。  屏幕剪辑  比较运算符在使用时需要注意一个问题，不能将比较运算符“==”误写成赋值运算符“=”。  2.3.4 逻辑运算符  逻辑运算符用于对布尔型的数据进行操作，其结果仍是一个布尔型。  屏幕剪辑  在使用逻辑运算符的过程中，需要注意以下几个细节：  1、逻辑运算符可以针对结果为布尔值的表达式进行运算。如：x > 3 && y != 0。  2、运算符“&”和“&&”都表示与操作，当且仅当运算符两边的操作数都为true时，其结果才为true，否则结果为false。当运算符“&”和“&&”的右边为表达式时，两者在使用上还有一定的区别。在使用“&”进行运算时，不论左边为true或者false，右边的表达式都会进行运算。如果使用“&&”进行运算，当左边为false时，右边的表达式不会进行运算，因此“&&”被称作短路与。  3、运算符“|”和“||”都表示或操作，当运算符两边的操作数任何一边的值为true时，其结果为true，当两边的值都为false时，其结果才为false。同与操作类似，“||”表示短路或，当运算符“||”的左边为true时，右边的表达式不会进行运算。  4、运算符“^”表示异或操作，当运算符两边的布尔值相同时（都为true或都为false），其结果为false。当两边布尔值不相同时，其结果为true。  重、难、疑点：  重点：变量的声明及应用  难点：变量的声明及应用 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业：  任意输入两个数，并计算它们之和、之差、之乘、之除，然后输出计算结果到屏幕上。  提示：  输出语句：  System.out.println(“两数之和是”，sum);  输入语句：  Scanner in=new Scanner(System.in);  int a=in.nextInt();  double d=in.nextDouble();  使用Scanner类时，在程序的最前边加入import java.util.\*;语句 | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第2章Java基础编程 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第二周 |
| 教学内容：   1. 掌握流程语句的应用   基本内容：  2.4 选择结构语句  Java中有一种特殊的语句叫做选择语句，它也需要对一些条件作出判断，从而决定执行哪一段代码。选择语句分为if条件语句和switch条件语句。  2.4.1 if条件语句  if条件语句分为三种语法格式，每一种格式都有其自身的特点。  1、if语句：如果满足某种条件，就进行某种处理。  屏幕剪辑屏幕剪辑  2、if…else语句：指如果满足某种条件，就进行某种处理，否则就进行另一种处理。  屏幕剪辑屏幕剪辑  3、if…else if…else语句：用于对多个条件进行判断，进行多种不同的处理。  屏幕剪辑屏幕剪辑  2.4.2 switch条件语句  只能针对某个表达式的值作出判断，从而决定程序执行哪一段代码。  课堂练习：计算个人实发工资（税前工资通过控制台输入）  应纳个人所得税税额=应纳税所得额×适用税率-速算扣除数  扣除标准3500元/月（2011年9月1日起正式执行  个税计算7级标准  全月应纳税所额 税率 速算扣除数（元）  全月应纳税额不超过1500元 3% 0  全月应纳税额超过1500元至4500元 10% 105  全月应纳税额超过4500元至9000元 20% 555  全月应纳税额超过9000元至35000元 25% 1005  全月应纳税额超过35000元至55000元 30% 2755  全月应纳税额超过55000元至80000元 35% 5505  全月应纳税额超过80000元 45% 13505  2.5.1 while循环语句  格式：    while语句会反复地进行条件判断，只要条件成立，{}内的执行语句就会执行，直到条件不成立，while循环结束。执行流程用下图表示：  说明: 2-24  例子：打印1-4    课堂练习：  宰相的麦子：相传古印度宰相达依尔，是国际象棋的发明者。有一次，国王因为他的贡献要奖励他，问他想要什么。达依尔说：“只要在国际象棋棋盘上（共64格）摆上这么些麦子就行了：第一格一粒，第二格两粒，……，后面一格的麦子总是前一格麦子数的两倍，摆满整个棋盘，我就感恩不尽了。”国王一想，这还不容易。于是令人扛来一袋麦子，可很快用完了，又扛来一袋，很快用完了……国王很奇怪，请你帮国王计算一下，要多少立方米的麦子。（1立方米的麦子约为1.42×108粒）  2.5.2 do while循环语句  do…while循环语句和while循环语句功能类似。  格式：  说明: 屏幕剪辑  例子：打印1-4  说明: 屏幕剪辑  2.5.3 for循环语句  for循环语句是最常用的循环语句，一般用在循环次数已知的情况下。  格式：    例子：打印1-4  说明: 屏幕剪辑  2.5.4 循环嵌套  嵌套循环是指在一个循环语句的循环体中再定义一个循环语句的语法结构。while、do…while、for循环语句都可以进行嵌套，并且它们之间也可以互相嵌套，如最常见的在for循环中嵌套for循环，格式如下。  说明: 屏幕剪辑  2.5.5 跳转语句（break、continue）  跳转语句用于实现循环执行过程中程序流程的跳转，在Java中的跳转语句有break语句和continue语句。接下来分别进行详细地讲解。  1、break语句  在switch条件语句和循环语句中都可以使用break语句。当它出现在switch条件语句中时，作用是终止某个case并跳出switch结构。当它出现在循环语句中，作用是跳出循环语句，执行后面的代码。  2、continue语句  continue语句用在循环语句中，它的作用是终止本次循环，执行下一次循环。  课堂练习：  用一张一元票换1分、2分和5分的硬币，每种至少一枚， 问有哪几种换法（各几枚）？  重、难、疑点：  重点：流程语句  难点：流程语句 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第3章面向对象程序设计基础 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第三周 |
| 教学内容：   1. 掌握类的定义及使用。 2. 掌握对象的创建及使用。 3. 掌握this、static关键字。   基本内容：  3.2类与对象  类和对象是Java的核心和本质。它们是Java语言的基础，编写一个Java程序，在某种程度上来说就是定义类和创建对象。定义类和建立对象是Java编程的主要任务。  3.2.1 类的定义  1、类的声明格式：  <类首声明>  {<类主体>}  类首声明定义类的名字、访问权限以及与其它类的关系等。  类主体定义类的成员，包括变量（数据）和方法（行为）  （1）类首的声明  [<修饰符>] class <类名> [extends <超类名>] [implements <接口名>]  class: 类定义的关键字；  extends: 表示类和另外一些类（超类）的继承关系；  implements: 表示类实现了某些接口；  修饰符: 表示类访问权限（public、缺省方式等）和一些其它特性（abstract、final等）；一个类可以同时有多个修饰符(任意排序)，但不能有相同的修饰符。  缺省方式：class没有修饰符，只能被同一个包名中的类访问  public：能被所有的类访问  abstract:修饰的类被称为抽象类。抽象类就是没有具体对象的概念类。  final：不能被继承，即类不能有子类  关键字 public表明所定义的类是一个公共类  一个java文件可以包含多个类，但最多只能包含一个公共类，而且这个公共类必须与其所在的文件同名  例如：public class J\_HelloJava 文件名命名为 J\_HelloJava.java  （2） 类主体  <类首声明> { 　　 //类首，以下为类主体  <成员变量的声明>  <成员方法的声明及实现>  }  成员变量即类的数据，反映了类的属性和状态。  成员方法即类的行为（对数据的操作）  成员方法可以是普通方法、构造方法、main方法  （3）成员变量的声明格式  [<修饰符>][static][final][transient] <变量类型> <变量名>  static: 表示是一个类成员变量（静态变量）;  final: 表示是一个常量(最终成员变量）；  例：final double PI=3.1415926;  transient: 表示一个临时变量,不能被javap命令反编译。反编译格式：javap 类名或javap  private 类名  修饰符: 表示变量的访问权限（缺省访问、public、protected和private）  作用域：在整个类中都有效，与声明的位置无关，但不提倡放在类中其它位置，一般置于类体前。  如果成员变量没有赋初值，在使用时候取该数据类型默认值，对成员变量的操作只能放在方法中，但可在声明的同时赋初值。默认初值如下图所示。    （4）成员方法的声明格式  [<修饰符>]<返回类型> <方法名> ([<参数列表>])  [throws <异常类>] {  方法体  }  修饰符：  方法的访问权限(缺省的、public、protected和private)  static: 类方法（静态方法）；  abstract: 抽象方法（无方法体的方法）；  final: 最终方法（不能被子类改变）。  throws: 表示抛出异常  2、构造方法  类的构造方法是一种特殊方法，作用是在创建对象时Java系统调用构造方法去初始化新建对象的成员变量;  例：Student s1=new Student(“张三”,’男’,20080301);  如果没有自定义构造方法，则Java调用类的默认构造方法。Java将使用缺省值来初始化成员变量。  例：Student s2=new Student();  构造方法的特性  类的构造方法名必须和类名相同；  构造方法没有返回值（在构造方法名字前连void也不要加）；  可以有public, protected, private等任何访问的修饰词或没有修饰。 但不能有以下非访问性质的修饰： abstract, final, native, static 或 synchronized。  3、main方法  public static void main (String args[])  main方法是所有java应用程序执行的入口，可以运行的Java应用程序必须含有main方法  main方法是JVM（java虚拟机）自动调用，需要让JVM可见，所以main方法需要public修饰。同理，main方法所在的类也需要public修饰符。  由于main方法是所有程序的入口，也就是main被调用时没有任何对象创建，不通过对象调用某一方法，只有将该方法定义为静态方法，所以main方法是一个静态方法，即需要static修饰。  JVM对于java程序已经是最底层，由它调用的方法的返回值已经没有任何地方可去，因此，main方法返回值为空，即需用void修饰。  String args[ ]我们用于在接受命令行传入的参数。  3.2.2 对象的创建与使用  当创建一个类时，就可以实例化该种类型的对象。实例化就是为对象分配内存，并同时对对象进行初始化，通过new运算符和构造方法完成。  创建对象的两种方式：  (1)先声明对象，然后实例对象  例：Student s; //声明对象  s=new Student(); //实例化对象  (2)声明对象的同时进行对象实例化  <类名> <对象名>=new <类名>(参数)；  例： Student s=new Student();  对象的成员的使用  类具有成员变量和成员方法，因此对象也具有成员变量和成员方法  成员变量的引用  引用格式：引用对象名.变量名  如:Student s1=new Student();  s1.name=“张三”;  成员方法的调用  调用格式：引用对象名.方法名([实际参数列表])  如: s1.setName("张三");  3、变量的作用域   * 类成员变量的作用域 * 参数和方法中定义变量的作用域 * 语句块中定义变量的作用域 * 成员变量与方法中定义的变量同名时的情况   3.3 构造方法  3.3.1 构造方法的定义  在一个类中定义的方法如果同时满足以下三个条件，该方法称为构造方法，具体如下：  方法名和类名相同  方法名的前面没有返回值类型的声明  方法中不能使用return语句返回一个值  在Java中的每个类都至少有一个构造方法，如果在一个类中没有定义构造方法，系统会自动为这个类创建一个默认的构造方法，这个默认的构造方法没有参数，在其方法体中没有任何代码，即什么也不做。  下面程序中Person类的两种写法效果是完全一样的。    如果为该类定义了构造方法，系统就不再提供默认的构造方法了    3.4 this关键字  1、通过this关键字可以明确地去访问一个类的成员变量，解决与局部变量名称冲突问题，具体示例如下：    在上面的代码中，构造方法的参数被定义为age，它是一个局部变量，在类中还定义了一个成员变量，名称也是age。在构造方法中如果使用“age”，则是访问局部变量，但如果使用“this.age”则是访问成员变量。  2、通过this关键字调用成员方法，具体示例代码如下：    在上面的speak()方法中，使用this关键字调用openMouth()方法。注意，此处的this关键字可以省略不写，也就是说上面的第6行代码写成“this.openMouth()”和“openMouth()”，效果是完全一样的  3、构造方法是在实例化对象时被Java虚拟机自动调用的，在程序中不能像调用其它方法一样去调用构造方法，但可以在一个构造方法中使用“this([参数1,参数2…])”的形式来调用其它的构造方法通过this关键字调用成员方法    注意：  1、只能在构造方法中使用this调用其它的构造方法  2、在构造方法中，使用this调用构造方法的语句必须位于第一行，且只能出现一次，下面的写法是非法的。  3、不能在一个类的两个构造方法中使用this互相调用,原因？  3.5static关键字  3.5.1 静态变量  在一个Java类中，可以使用static关键字来修饰成员变量，该变量被称作静态变量  静态变量被所有实例共享，可以使用“类名.变量名”的形式来访问，如System.out，(InputStream类的对象）  静态变量的应用，如信用卡的透支限额，南国校园卡的每天最高消费限额等。  static关键字只能用于修饰成员变量，不能用于修饰局部变量，否则编译会报错，下面的代码是非法的。  3.5.2 静态方法  被static关键字修饰的方法称为静态方法  同静态变量一样，静态方法可以使用“类名.方法名”的方式来访问，也可以通过类的实例对象来访问。  静态方法的应用，如Math.pow(x,y)。  在一个静态方法中只能访问用static修饰的成员，原因在于没有被static修饰的成员需要先创建对象才能访问，而静态方法在被调用时可以不创建任何对象。  3.5.3 静态代码块  在Java类中，使用一对大括号包围起来的若干行代码被称为一个代码块，用static关键字修饰的代码块称为静态代码块  当类被加载时，静态代码块会执行，由于类只加载一次，因此静态代码块只执行一次,类被加载的情况：  1. 创建类的实例 2. 调用类的静态方法 3. 使用类的非常量静态字段 4. 初始化某个类的子类 5. 含有main()方法的类启动时  在程序中，通常使用静态代码块来对类的成员变量进行初始化    重、难、疑点：  重点：类的定义及使用和对象的创建及使用，this及static关键字的使用。  难点：类的定义及使用和对象的创建及使用，this及static关键字的使用。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第3章面向对象程序设计基础 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第三周 |
| 教学内容：  基本内容：  4.1 类的继承  4.1.1 继承的概念  在Java中，类的继承是指在一个现有类的基础上去构建一个新的类，构建出来的新类被称作子类，现有类被称作父类，子类会自动拥有父类所有可继承的属性和方法。在程序中，如果想声明一个类继承另一个类，需要使用extends关键字。  在类的继承中，需要注意一些问题，具体如下：  1、在Java中，类只支持单继承，不允许多重继承，也就是说一个类只能有一个直接父类，例如下面这种情况是不合法的。  屏幕剪辑  2、多个类可以继承一个父类，例如下面这种情况是允许的。  屏幕剪辑  3、在Java中，多层继承是可以的，即一个类的父类可以再去继承另外的父类，例如C类继承自B类，而B类又可以去继承A类，这时，C类也可称作A类的子类。下面这种情况是允许的。  屏幕剪辑  4、Java中的类都是Object的子类，即使在定义类时没有声明父类，Java也会自动将类定义为Object的子类，Object类是Java类库中提供的类。  4.1.1重写父类方法  在继承关系中，子类会自动继承父类中定义的方法，但有时在子类中需要对继承的方法进行一些修改，即对父类的方法进行重写。需要注意的是，在子类中重写的方法需要和父类被重写的方法具有相同的方法名、参数列表以及返回值类型。  屏幕剪辑  当子类重写父类的方法后，子类对象将无法访问父类被重写的方法，为了解决这个问题，在Java中专门提供了一个super关键字用于访问父类的成员。例如访问父类的成员变量、成员方法和构造方法。接下来分两种情况来学习一下super关键字的具体用法。  1、使用super关键字调用父类的成员变量和成员方法。具体格式如下：  屏幕剪辑  2、使用super关键字调用父类的构造方法。具体格式如下：  屏幕剪辑  需要注意的是，子类的构造方法被调用时父类的构造方法也被调用。通过super调用父类构造方法的代码必须位于子类构造方法的第一行，并且只能出现一次。  屏幕剪辑  将例程4-4第11行代码去掉，再次编译程序会报错:  出错的原因是，在子类的构造方法中一定会调用父类的某个构造方法。这时可以在子类的构造方法中通过super指定调用父类的哪个构造方法，如果没有指定，在实例化子类对象时，会自动调用父类无参的构造方法。   1. 为了解决上述程序的编译错误，可以在子类中显示地调用父类中已有的构造方法，当然也可以选择在父类中定义无参的构造方法 2. 构造方法的继承 3. 构造方法不存在继承关系。 4. 如果子类没有构造方法，则它默认调用超类无参数的构造方法； 5. 如果子类有构造方法，那么在创建子类的对象时，则将先执行超类的构造方法，然后再执行自己的构造方法。 6. 对于超类中包含有参数的构造方法，子类可以通过在自己的构造方法中使用super关键字来引用，而且必须是子类构造方法中的第一条语句。   用super调用超类的构造方法：super([参数列表])  子类对象的创建与实例化过程  1、分配对象所需的全部内存空间，并初始化为“0”值  2、按继承关系，自顶向下显式初始化  3、按继承关系，从顶向下调用构造方法  注意：一般子类构造方法的第1行通过super来调用父类的构造方法，若不使用super关键字指定，则自动调用父类的无参数构造方法。如果父类中没有无参数的构造方法，则将产生错误。  需要注意的是，若子类调用父类的构造方法，在子类的构造方法中省略掉super关键字，则系统默认有“super()”，即调用父类不带参数的构造方法。  由于Java语言规定如果一个类中含有一个或多个构造方法，系统不提供默认的构造方法（不含参数的构造方法），所以当在父类中定义了多个构造方法时，应考虑包括一个不带参数的构造方法，以防止子类省略super关键字时出现错误。  继承关系中对成员的访问（最近匹配原则）  1、在子类中访问属性和方法时将优先查找自己定义的属性和方法。如果该成员在本类存在，则使用本类的，否则，按照继承层次的顺序到其祖先类查找。  2、this关键字特指本类的对象引用，使用this访问成员则首先在本类中查找，如果没有，则到父类逐层向上找。  3、super特指访问父类的成员，使用super首先到直接父类查找匹配成员，如果未找到，再逐层向上到祖先类查找  终止继承  出于安全性方面的考虑，要避免子类继承超类的某个方法，可以使用“final”关键字来使继承终止。 这样使此方法不会在子类中被覆盖（即子类中不能有和此方法同名的方法）  不能被继承的类称为最终类。  如：final class Last;  用final说明的成员方法为最终方法。  如：public final void printsuper( )  注：类只有用public、final或缺省定义，如果用final定义的类则不能被继承。  4.2 final关键字  final关键字可用于修饰类、变量和方法，它有“这是无法改变的”或者“最终”的含义，因此被final修饰的类、变量和方法将具有以下特性：  1、final修饰的类不能被继承。  2、final修饰的方法不能被子类重写。  3、final修饰的变量（成员变量和局部变量）是常量，只能赋值一次。  4.2.1 final关键字修饰类  Java中的类被final关键字修饰后，该类将不可以被继承，也就是不能够派生子类。  当一个类的方法被final关键字修饰后，这个类的子类将不能重写该方法。  4.3 抽象类和接口  4.3.1 抽象类  当定义一个类时，常常需要定义一些方法来描述该类的行为特征，但有时这些方法的实现方式是无法确定的。  针对上面描述的情况，Java允许在定义方法时不写方法体，不包含方法体的方法为抽象方法，抽象方法必须使用abstract关键字来修饰，具体示例如下：  屏幕剪辑  当一个类中包含了抽象方法，该类必须使用abstract关键字来修饰，使用abstract关键字修饰的类为抽象类，具体示例如下:  屏幕剪辑  在定义抽象类时需要注意，包含抽象方法的类必须声明为抽象类，但抽象类可以不包含任何抽象方法，只需使用abstract关键字来修饰即可。另外，抽象类是不可以被实例化的，因为抽象类中有可能包含抽象方法，抽象方法是没有方法体的，不可以被调用。如果想调用抽象类中定义的方法，则需要创建一个子类，在子类中将抽象类中的抽象方法进行实现。  抽象类   1. 抽象类也可有普通的成员变量或方法。 2. 抽象类可以有构造方法，但不能直接用来生成实例。一般可通过定义子类进行实例化。 3. 可以生成抽象类的变量，该变量可以指向具体的一个子类的实例。   abstract class Employee{  abstract void raiseSalary(int i) ;  }  class Manager extends Employee{  void raiseSalary(int i ){ ….}  }  Employee e = new Manager( ) ;  4.3.2 接口  如果一个抽象类中的所有方法都是抽象的，则可以将这个类用另外一种方式来定义，即接口。在定义接口时，需要使用interface关键字来声明，具体示例如下：  屏幕剪辑  从示例中会发现抽象方法breathe()并没有使用abstract关键字来修饰，这是因为接口中定义的方法和变量都包含一些默认修饰符。接口中定义的方法默认使用“public abstract”来修饰，即抽象方法。接口中的变量默认使用“public static final”来修饰，即全局常量。  由于接口中的方法都是抽象方法，因此不能通过实例化对象的方式来调用接口中的方法。此时需要定义一个类，并使用implements关键字实现接口中所有的方法。  为了加深初学者对接口的认识，接下来对接口的特点进行归纳，具体如下：  1、接口中的方法都是抽象的，不能实例化对象。  2、当一个类实现接口时，如果这个类是抽象类，则实现接口中的部分方法即可，否则需要实现接口中的所有方法。  3、一个类通过implements关键字实现接口时，可以实现多个接口，被实现的多个接口之间要用逗号隔开。  4、一个接口可以通过extends关键字继承多个接口，接口之间用逗号隔开。具体示例如下：  5、一个类在继承另一个类的同时还可以实现接口，此时，extends关键字必须位于implements关键字之前。  接口和抽象类的区别   1. 接口可以多重继承，抽象类不可以 2. 抽象类内部可以有实现的方法，接口则没有实现的方法 3. 接口与实现它的类不构成类的继承体系，即接口不是类体系的一部分。因此，不相关的类也可以实现相同的接口。而抽象类是属于一个类的继承体系，并且一般位于类体系的顶层。   接口的优势：通过实现多个接口实现多重继承，能够抽象出不相关类之间的相似性。  创建类体系的基类时，若不定义任何变量并无需给出任何方法的完整定义，则定义为接口；必须使用方法定义或变量时，考虑用抽象类。  4.4 类的多态性  类的多态表现为引用多态和方法的多态。  1、引用多态表现为父类指向子类的对象。例如：  Person类是student类的父类，student类是person类的子类，每一个student都属于person，但反之则不然，这个机制映射到程序中就是：  Person p=new student();这时的p既可以是person类的对象，也可以是student类的对象，即p具有多态，但是反之student s=new person();则不成立。可以强制转换  student s=(student)new person();  2、方法的多态  方法的多态性是指同一个名字的若干个方法，有不同的实现（即方法体中的代码不一样）。  多态的两种表现形式：  （1） 同一类中的方法多态，称为方法重载overloading(静态多态性)：同一类中允许多个同名方法，区分要点：参数的个数不同，或者是参数类型不同，或者是参数的顺序不同，返回值可以相同也可以不同  （2） 子类对父类方法的重新定义，称为方法覆盖overriding (动态多态性) ：方法名、返回值和参数形态完全一样。  例1：下面程序中第4行调用哪一行的show方法？  public class Test{  public static void main(String argv[]) {  Test t = new Test();  t.show(1+3);  }  public void show(int Val1,int Val2){  System.out.println(Val1+Val2);  }  public void show(int Val1) {  System.out.println(Val1);  }  public void show(int Val1,int Val2,int Val3) {  System.out.println(Val1+Val2+Val3);  }  }  方法重载的原则  方法名相同。  方法的参数类型、个数、顺序只要有一项不相同。  返回类型、修饰符可相同或不相同。  public int area(int r)  public float area(int h,float r)  public float area(float h,int r)  是正确的方法重载，而下面的则不是正确的方法重载：  public int area(float h,float r)  public float area(float h,float r)  构造方法也能重载  例2：下述哪一组方法，是一个类中方法重载的正确写法？  A) int addValue( int a, int b ){return a+b;}  float addValue ( float a, float b) {return a+b;}  B) int addValue (int a, int b ){value=a+b; }  float addValue ( int a, int b) {return (float)(a+b);}  C) int addValue( int a, int b ){return a+1;}  int addValue ( int a, int b) {return a+b;}  D) int addValue( int a, int b ) {return a+b;}  int addValue ( int x, int y ) {return x+y;}  例3：对于下列代码：  public class SupperClass {  public int sum( int a, int b) {  return a+b;  }  }  class SubClass extends SupperClass { }  下述哪个方法可以加入类SubClass? ( )  A) int sum ( int a, int b ){ return a+b;}  B) public void sum (int a, int b ){ return ;}  C) public float sum ( int a, int b){ return a+b;}  D) public int sum ( float a, int b ) { return a+b;}  例4：public class TestDemo {  int m=5;  public void some(int x) {  m=x;  }  public static void main(String args []) {  new Demo().some(7);  }  }  class Demo extends TestDemo {  int m=8;  public void some(int x) {  super.some(x);  System.out.println(m);  }  }  A．5    B．8   C．7   D．无任何输出  E．编译错误  方法的覆盖（重写）  在子类和超类中有同名的方法（参数也相同），子类中的方法覆盖超类的方法。  如果超类和子类有同名且参数相同的方法，那么超类的对象调用超类的方法，子类的对象调用子类的方法。  通过覆盖可以使同名的方法在不同层次的类中有不同的实现  注意：子类中重写的方法和父类中被重写的方法要具有相同的名字，相同的参数表和相同的返回类型，只是函数体不同。  方法覆盖的原则（1）  1、子类方法的名称、参数签名和返回类型必须与其父类的方法的名称、参数签名和返回类型一致  public class Base{  public void method(){...}  }  public class Sub extends Base{  public void method(){...}  public int method(int a){  return 0;  }  }  方法覆盖的原则（2）  2、子类方法不能缩小父类方法的访问权限  public class Base{  public void method(){...}  }  public class Sub extends Base{  protected void method(){...}//编译错误，子类方法缩小了父类方法的访问权限  }  方法覆盖的原则（3）  3、子类方法不能抛出比父类方法更多的异常  4、方法覆盖只存在于子类和父类（包括直接父类和间接父类）之间。在同一个类中方法只能被重载，不能被覆盖。  5、父类的静态方法不能被子类覆盖为非静态的方法，反之亦然；  6、子类可以定义与父类的静态方法同名的静态方法，以便在子类中隐藏父类的静态方法。  7、父类的私有方法不能被覆盖  8、父类的抽象方法可以被子类覆盖：子类实现父类的方法或重新声明父类的抽象方法。  课堂练习：多态性程序应用  采用多态方法实现一个工资支付程序。使用抽象类，根据员工的类型多态地计算其应得的工资。  某公司支付员工工资系统。该企业有4类员工：领固定周薪的员工；按销售额提成的员工；带底薪并按销售额提成的员工；计时取酬的员工(如果一周工时超过40小时， 则还需对额外的工时支付加班费（正常的1.5倍）)。目前，公司决定对带底薪并按销售额提成的成员工增加15%的底薪。该公司要开发一个Java应用程序，多态地进行工资计算。  提示：该Java程序包含的类有： Employee（员工类抽象类）， SalariedEmployee（周薪）， CommissionEmployee（销售提成）， HourlyEmployee（小时工）， BasePlusCommissionEmployee（底薪+销售提成）， PayrollSystemTest （测试类）。初值自定。  4.4 多态  4.4.1 引用多态  在Java中为了实现多态，允许使用一个父类类型的变量来引用一个子类类型的对象，根据被引用子类对象特征的不同，得到不同的运行结果。  4.4.2 对象的类型转换  将例程4-14中的animalShout()方法进行修改，具体代码如下：  4.4.2 对象的类型转换  屏幕剪辑  需要注意的是，在进行类型转换时也可能出现错误，例如在例程4-14中调用animalShout()方法时传入一个Dog类型的对象，这时进行强制类型转换就会出现出错。  针对这种情况，Java提供了一个关键字instanceof，它可以判断一个对象是否为某个类(或接口)的实例或者子类实例，语法格式如下  屏幕剪辑  4.4.3 Object类  在JDK中提供了一个Object类，它是所有类的父类，即每个类都直接或间接继承自该类。先来看一个例子，如例程4-16所示。  因为Animal默认继承自Object类，在Object类中定义了toString()方法，在该方法中输出了对象的基本信息。  在实际开发中，通常希望对象的toString()方法返回的不仅仅是基本信息，而是一些特有的信息，这时重写Object的toString()方法便可以实现，如例程4-17所示。  4.4.4 匿名内部类  前面多态的讲解中，如果方法的参数被定义为一个接口类型，那么就需要定义一个类来实现接口，并根据该类进行对象实例化。除此之外，还可以使用匿名内部类来实现接口。对于初学者而言，可能会觉得匿名内部类的写法比较难理解，接下来分两步来编写匿名内部类，具体如下：  1、在调用animalShout ()方法时，在方法的参数位置写上new Animal (){}，这相当于创建了一个实例对象，并将对象作为参数传给animalShout ()方法。在new Animal ()后面有一对大括号，表示创建的对象为Animal的子类实例，该子类是匿名的。具体代码如下所示：屏幕剪辑  4.6import和包  包(package)是类的逻辑组织形式。  Java提供的用于程序开发的类就放在各种包中。也可以自己创建包。  一般在创建项目时为了便于管理，源文件和字节码文件是分开保存在不同的目录中的。  项目的文件夹为myproject;  源文件保存在myproject下的src文件夹中；  字节码文件保存在myproject下的bin文件夹中  包与文件夹  在实际项目开发中，一般每个类都应该定义在某个包中。这样一方面易于管理，另一方面可以有效地防止命名冲突。包的名称一般按公司的域名反向书写。如：com.misxp  如果一个类是这样定义的：  package com.misxp;//声明一个包的语句要写在源程序文件的第一行。  public class PackageExercise  …  那么这个类所在的文件PackageExercise.java应该保存在myproject/src/com/misxp下  它的编译好的字节码文件应该保存在myproject/bin/com/misxp下  Java的类和包  java常用的包有：  java.lang 语言包，唯一一个不要把它明确引入程序的包  java.util　 实用包  java.awt 抽象窗口工具包  java.text 文本包  java.io 输入输出流的文件包  java.applet Applet应用程序  java.net 网络功能  引用Java定义的包  如果要使用Java类中的包，要在源程序中用import语句导入。  import <包名1>[.<包名2>[.<包名3>……]].<类名>|\*;  如果有多个包或类,用“.”分割，“\*”表示包中所有的类。  如：  import java.applet.Applet;//导入Java.applet包中的Applet类  import java.awt.\* //导入Java.awt包中所有的类  Java源程序文件的结构，一个Java的源程序文件依次有以下部分：  包的声明语句 (package…)  包的导入语句 (import …)  类的声明 (public class …)  包和访问权限  一个包中只有访问权限为public的类才能被其它包引用（创建此类的对象），其它有默认访问权限的类只能在同一包中使用。  在不同包中类成员的访问权限。  public类的public成员可以被其它包的代码访问。它的protected成员可以被由它派生的其它包中的子类访问。  默认访问权限类的成员，不能被其它包的代码访问  4.6.3 给Java应用打包  给Java应用打包有很多好处，接下来简单介绍一下打包jar文件的好处，如下所示：  安全：可以对jar文件进行数字签名，让能够识别数字签名的用户使用。  节省空间：当把.class文件打成jar压缩文件，会节省空间，如果将jar文件在网络上传输，也会加快传输速率。  可移植性：只要有Java虚拟机，jar包就可以在任何平台上运行。  4.6.3 给Java应用打包  在命令行下使用命令打包请自己查看课本。  Eclipse下打包步骤查看文档。  4.7 访问控制  在Java中，针对类、成员方法和属性提供了四种访问级别，分别是private、default、protected和public。接下来通过一个图将这四种控制级别由小到大依次列出:  4.7 访问控制  private(类访问级别) ：如果类的成员被private访问控制符来修饰，则这个成员只能被该类的其它成员访问，其它类无法直接访问。实现类的良好封装。  default(包访问级别) ：如果一个类或者类的成员不使用任何访问控制符修饰，则称它为默认访问控制级别，这个类或者类的成员只能被本包中的其它类访问。  protected(子类访问级别) ：如果一个类的成员被protected访问控制符修饰，那么这个成员既能被同一包下的其它类访问，也能被不同包下该类的子类访问。  public(公共访问级别) ：这是一个最宽松的访问控制级别，如果一个类或者类的成员被public访问控制符修饰，那么这个类或者类的成员能被所有的类访问，不管访问类与被访问类是否在同一个包中。  4.7 访问控制  接下来通过一个表将这四种访问级别更加直观的表示出来，如表4-2所示。  屏幕剪辑    重、难、疑点：  重点：  难点： | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第3章面向对象程序设计基础 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第五周 |
| 教学内容：  掌握类的定义及使用。  基本内容：   1. 银行模拟系统   问题描述：银行有存、取款和修改密码操作，约定银行的初始密码为6个8，每天的取款额度低于20000，存款不限，请编写Java程序实现以上三种操作的业务。每一种操作都有相应提示。运行界面参考如图：    参考代码：  package paper3;  import java.util.Scanner;  public class BankSys {  private double money=0;  static int maxqk=20000;  String psd="888888";  public void ck(double ckmoney){  money=money+ckmoney;  }  double getmoney(){  return money;  }  public void qk(double qkmoney){  if(qkmoney>maxqk) System.out.println("超过最大取款额");  else if(qkmoney>money) System.out.println("余额不足");  else {  money=money-qkmoney;System.out.println("取款成功，你的余额是"+money);  }  }  public void setpsd(){  Scanner in=new Scanner(System.in);  System.out.println("请输入旧密码：");  String oldpsd=in.next();  if(oldpsd.equalsIgnoreCase(psd)) {  System.out.println("请输入新密码：");  String newpsd=in.next();  psd=newpsd;  System.out.println("恭喜你，密码修改成功。");  }  else{  System.out.println("遗憾，你的密码有误，密码修改不成功！");  }  }  public static void main(String[] args) {  BankSys bank=new BankSys();  System.out.println("请选择你的操作：0：修改密码；1：存款；2：取款;3:退出系统");  Scanner in=new Scanner(System.in);  int c=in.nextInt();  while(c!=3){  switch(c){  case 0:bank.setpsd();break;  case 1:System.out.println("请输入你的存款金额");  double ckmoney=in.nextDouble();  bank.ck(ckmoney);  System.out.println("你的余额："+bank.getmoney());  break;  case 2:System.out.println("请输入你的取款金额");  double qkmoney=in.nextDouble();  bank.qk(qkmoney);  break;  case 3:break;  }  System.out.println("请选择你的操作：0：修改密码；1：存款；2：取款;3:退出系统");  c=in.nextInt();  }  }  }   1. 工资计算模拟系统   问题描述：  实现一个工资支付程序。使用抽象类，根据员工的类型多态地计算其应得的工资。  某公司支付员工工资系统。该企业有4类员工：领固定周薪的员工；按销售额提成的员工；带底薪并按销售额提成的员工；计时取酬的员工(如果一周工时超过40小时， 则还需对额外的工时支付加班费（正常的1.5倍）)。目前，公司决定对带底薪并按销售额提成的成员工增加15%的底薪。该公司要开发一个Java应用程序，多态地进行工资计算。  运行结果参考图：  屏幕剪辑  代码参考：  package paper3;  public class SalarySys {  public static void main(String[] args) {  weeksal w=new weeksal("军霞","010010",1000);  hourssal h=new hourssal("育贡","010011",40,99);  salesal s=new salesal("文化","010012",30000,0.06);  basesalesal b=new basesalesal("江伟","010013",1000,300,0.04);  System.out.println("周薪工资:");  System.out.println("员工姓名:"+w.getname());  System.out.println("员工工号:"+w.getnum());  System.out.println("员工工资:"+w.getsalary());  System.out.println("小时工工资:");  System.out.println("员工姓名:"+h.getname());  System.out.println("员工工号:"+h.getnum());  System.out.println("员工工资:"+h.getsalary());  System.out.println("销售提成工工资:");  System.out.println("员工姓名:"+s.getname());  System.out.println("员工工号:"+s.getnum());  System.out.println("员工工资:"+s.getsalary());  System.out.println("销售提成+底薪工资:");  System.out.println("员工姓名:"+b.getname());  System.out.println("员工工号:"+b.getnum());  System.out.println("员工工资:"+b.getsalary());  }  }  abstract class sal{  double salary=0;  String name=null;  String num=null;  public abstract double getsalary();  }  class weeksal extends sal{  double weeksal=0;  public weeksal(String name,String num,double weeksal){  this.name=name;  this.num=num;  this.weeksal=weeksal;  }  public double getsalary(){  salary=weeksal;  return salary;  }  public String getname(){  return name;  }  public String getnum(){  return num;  }  }  class salesal extends sal{  double sales=0;  double tc=0;  public salesal(String name,String num,double sales,double tc){  this.name=name;  this.num=num;  this.sales=sales;  this.tc=tc;  }  public double getsalary(){  salary=sales\*tc;  return salary;  }  public void setsales(double sales){  this.sales=sales;  }  public String getname(){  return name;  }  public String getnum(){  return num;  }  }  class basesalesal extends sal{  double sales=0;  double basesal=0;  double tc=0;  public basesalesal(String name,String num,double sales,double basesal,double tc){  this.name=name;  this.num=num;  this.sales=sales;  this.basesal=basesal;  this.tc=tc;  }  public double getsalary(){  salary=basesal+sales\*tc;  return salary;  }  public void setsales(double sales){  this.sales=sales;  }  public String getname(){  return name;  }  public String getnum(){  return num;  }  }  class hourssal extends sal{  double hours=0;  double hourssal=0;  public hourssal(String name,String num,double hours,double hourssal){  this.name=name;  this.num=num;  this.hours=hours;  this.hourssal=hourssal;  }  public double getsalary(){  if(hours>40) {  salary=40\*hourssal+(hours-40)\*1.5\*hourssal;  }  else {  salary=hours\*hourssal;  }  return salary;  }  public double gethours(){  return hours;  }  public double gethourssal(){  return hourssal;  }  public String getname(){  return name;  }  public String getnum(){  return num;  }  }  重、难、疑点：  重点：无  难点：无 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第4章数组与字符串 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第五周 |
| 教学内容：   1. 掌握数组的定义及使用 2. 掌握字符串的应用   基本内容：  4.1数组  数组是具有相同类型变量的集合。在数组中，各个变量称之为元素。其中，同一数组中的所有元素都有相同的名字，只是下标不同。只有一个下标的数组称之为一维数组，有多个下标的数组称之为多维数组。  4.1.1一维数组      数组名代表的是数组的首地址，下标则是数组元素到数组开始的偏移量。系统为数组在内存分配的是一片连续的存储的单元，如定义了“int a[ ]”有10个元素，则它的10个元素在内存中的排列情况如图所示：    屏幕剪辑    经常采用循环结构来给数组元素进行初始化，先声明一个数组，然后在循环中使用赋值语句逐个初始化数组元素。  例如：   int a[ ] = new int[10];  for(i = 0; i < 10; i++)  {  a[i] = i + 1;  }  通过循环，数组下标 i 从 0逐个递增到9(因为当 i = 10时，条件i < 10为假，不能进入循环体)。  4.1.2 多维数组的应用  在Java中，多维数组是由若干行和若干列组成的数组。在人们工作生活与学习中，要使用二维表格、矩阵、行列式等，都可以表示成多维数组。例如：  int D[ ][ ] = new int[3][4];  该语句声明并创建了一个3行4列的数组D。这个数组在逻辑上可以表示成一个int类型的矩阵。  　　　 25 53 67 19  　 　D = 38 65 90 77  12 83 44 92  也就是说，这个数组在逻辑上可以表示为：  D[0][0] D[0][1] D[0][2] D[0][3]  D[1][0] D[1][1] D[1][2] D[1][3]  D[2][0] D[2][1] D[2][2] D[2][3]  4.2String类和StringBuffer类 4.2.1 String类的初始化  Java中定义了String和StringBuffer两个类来封装字符串，并提供了一系列操作字符串的方法，它们都位于java.lang包中，因此不需要导包就可以直接使用。  在操作String类之前，首先需要对String类进行初始化，在Java中可以通过以下两种方式对String类进行初始化，具体如下：  1、使用字符串常量直接初始化一个String对象，具体代码如下：    2、使用String的构造方法初始化字符串对象    4.2.2 String类的常见操作  1．字符串的基本操作    在程序中，需要对字符串进行一些基本操作，如获得字符串长度、获得指定位置的字符等。  2．字符串的转换操作  序开发中，经常需要对字符串进行转换操作，例如将字符串转换成数组的形式，将字符串中的字符进行大小写转换等。  3．字符串的替换和去除空格操作  程序开发中，用户输入数据时经常会有一些错误和空格，这时可以使用String类的replace()和trim()方法，进行字符串的替换和去除空格操作  4．字符串的判断操作  操作字符串时，经常需要对字符串进行一些判断，如判断字符串是否以指定的字符串开始、结束，是否包含指定的字符串，字符串是否为空等。  5．字符串的截取和分割  在String类中针对字符串的截取和分割操作提供了两个方法，其中，substring()方法用于截取字符串的一部分，split()方法可以将字符串按照某个字符进行分割。  💣脚下留心  String字符串在获取某个字符时，会用到字符的索引，当访问字符串中的字符时，如果字符的索引不存在，则会发生StringIndexOutOfBoundsException（字符串角标越界异常）  4.2.3 StringBuffer  为了便于对字符串的修改，在JDK中提供了一个StringBuffer类（也称字符串缓冲区）。StringBuffer类和String类最大的区别在于它的内容和长度都是可以改变的。StringBuffer类似一个字符容器，当在其中添加或删除字符时，并不会产生新的StringBuffer对象。    StringBuffer类和String类有很多相似之处，初学者使用时容易混淆。  1、String类表示的字符串是常量，一旦创建后，内容和长度都是无法改变的。StringBuffer表示字符容器，其内容和长度都可以随时修改。在操作字符串时，如果该字符串仅用于表示数据类型，则使用String类即可，但是如果需要对字符串中的字符进行增删操作，则使用StringBuffer类。  2、String类覆盖了Object类的equals()方法，而StringBuffer类没有覆盖Object类的equals()方法  3、String类对象可以用操作符“+”进行连接，而StringBuffer类对象之间不能  练习1：    练习2：  两个字符串中最大相同的子串  重、难、疑点：  重点：数组的定义及使用，字符串的应用  难点：数组的定义及使用，字符串的应用 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章图形用户界面设计 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第六周 |
| 教学内容：  掌握窗体的创建。  基本内容：  5.1　图形用户界面概述  GUI全称是Graphical User Interface，即图形用户界面，即应用程序提供给用户操作的图形界面，包括窗口、菜单、按钮、工具栏和其它各种图形界面元素  Java为GUI设计提供了丰富的类库，这些类分别位于java.awt和javax.swing包中，简称为AWT和Swing.  AWT （抽象窗口工具包），提供了一组用于开发用户界面组件类，事件处理类，图形绘制类，字体类和布局管理器类等。但是AWT过于依赖操作系统 本身提供的GUI工具，使得程序的可移植性降低。  Swing完全使用Java编写，不仅实现了AWT中的所有功能，而且提供了更加丰富的组件和功能，足以满足GUI设计的一切需求。但是太灵活，学习太复杂。  我们把awt图形用户界面包称为awt组件，也称为重量组件，而把swing图形用户界面包称为swing组件，又称为轻量组件。  AWT是用于创建图形用户界面的一个工具包，它提供了一系列用于实现图形界面的组件，如窗口、按钮、文本框、对话框等。这些组件都位于java.awt包中。  绘图1  **swing组件的层次结构**    **5.2　窗体容器和组件**  JFrame是带有标题、边框的顶层窗体。窗体是一个容器，在其内部可以添加其它组件。 创建窗体有两种方法，一 种是创建Jframe的子类，并重写其构造方法；另一种是创建Jframe的对象。  窗体默认被系统添加到显示器屏幕上，因此不允许将一个窗体添加到另一个容器中。    JFrame类的常用方法  JFrame():创建无标题的窗体  JFrame(String s)：创建标题为S的窗体  setMenuBar(MenuBar bar)：给窗体设置菜单  Dispose()：关闭窗体，释放资源  setVisible(boolean b)：设置窗体可见性  setSize(int w,int h)：设置窗体大小  Validate()：使窗体中的组件能显示出来  setTitle(String s)：设置标题内容  getTitle()：获取标题内容  setDefaultCloseOperation(int op)：设置窗体的关闭按钮可用，其值为EXIT\_ON\_CLOSE.  **【例5-1】通过构造JFrame对象创建最简单窗体。**  import javax.swing.\*;  class Example5\_1  {  public static void main(String[] args)  {  JFrame win=new JFrame("最简单窗体");  win.setSize(300,200);  win.setVisible(true);  win.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }  }  【例5-2】通过构造JFrame子类创建最简单窗体。  import javax.swing.\*;  class Ex5\_2 extends JFrame  {  Ex5\_2()  {  setSize(300,200);  setVisible(true);  setTitle("最简单窗体");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  }    public static void main(String[] args)  {  new Ex5\_2();  }  }  重、难、疑点：  重、难点：窗体的创建。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章图形用户界面 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第六周 |
| 教学内容：  1、掌握事件处理  2、掌握布局  基本内容：  5.2.2　按钮和事件处理  1、按钮JButton类  （1）按钮JButton类的常用方法  （见教材P106 表5.1）  （2）创建按钮对象  创建按钮对象的方法为：  JButton btn = new JButton(String text);  【例5-3】构造一个带按钮的窗体  import javax.swing.\*;  import java.awt.FlowLayout;  class Btn extends JFrame  {  JButton btn=new JButton("确定"); //创建按钮对象  Btn()  {  setSize(300,200);//设置窗体大小  setVisible(true); //设置窗体可见  setDefaultCloseOperation(EXIT\_ON\_CLOSE);//设置关闭窗体，退出程序  setLayout(new FlowLayout());//设置窗体布局  add(btn);//把按钮对象添加到窗体中  validate();//使窗体中的组件为可见  }  }  public class Example5\_3  {  public static void main(String[] args)  {  Btn btn=new Btn();  }  }  2、处理按钮事件  要定义按钮的处理事件，需要用到ActionListener接口。ActionListener是java.awt.event包中的一个接口，它定义了事件的处理方法。java.awt.event包对这个接口的定义是：  public interface ActionListener extends EventListener  {　　　//说明抽象方法  public abstract void actionPerformed(ActionEvent e)  }  在设计按钮对象btn处理事件的类时，就要实现这个接口。其一般形式如下：  class ClassName implements ActionListener  {  　 …  　　　　　btn.addActionListener(this);  …  　　　　 public void actionPerformed(ActionEvent e)  {  …  }  }  5.2.2　按钮和事件处理  在Java 中，要求产生事件的组件向它的监视器注册，这样，事件源与监视器就建立了一个对应关系（关联）。建立对应关系（关联）的语句如下：  对象名.addActionListener(ClassName);  其中，对象是事件源，ClassName是监视器。例如：  　　　　btn.addActionListener(this);  这条语句的意思是：按钮对象（事件源）btn向它的监视器（当前类）注册，也就是产生事件的事件源对象向监视器注册。  【例5-4】设计一个按钮事件程序  import javax.swing.\*;  import java.awt.FlowLayout;  import java.awt.event.ActionListener;  import java.awt.event.ActionEvent;  class BtnIcon extends JFrame implements ActionListener  {  ImageIcon icon = new ImageIcon("win.jpg");  JButton jbtn = new JButton("打开新窗体",icon);  BtnIcon()  {  setSize(200,200);  setVisible(true);  setTitle("按钮功能演示");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout());  add(jbtn);  validate();  jbtn.addActionListener(this);  }  public void actionPerformed(ActionEvent e)  { JFrame newf=new JFrame("新建窗体");  newf.setSize(150,150);  newf.setVisible(true);  }  }  public class Ex5\_4  {  public static void main(String[] args)  { new BtnIcon(); }  }  5.3.1　面板Jpanel  面板JPanel是一个可放置其他组件的容器。作为普通容器，它必须放置到一个顶层容器（窗体）之内。我们可以在JPanel中使用add()方法放置其他组件。面板主要用于合理安排界面布局。  创建面板的一般步骤为：  （1）创建面板对象： Panel myPanel=new Panel();  （2）将面板添加到窗体容器中：　add(myPanel);  （3）把组件放置到面板上：　myPanel.add(其他组件);  【例5-5】面板使用示例。  import java.awt.\*;  import javax.swing.\*;  class PanelTest extends JFrame  {  Panel panel1=new Panel();  Panel panel2=new Panel();  Button button1=new Button("Button1");  Button button2=new Button("Button2");  Button button3=new Button("Button3");  Button button4=new Button("Button4");  PanelTest()  {  setSize(200,150);  setVisible(true);  setTitle("面板容器示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout());  //将面板容器加入到窗体  add(panel1);  add(panel2);  //将其他组件加入到面板容器中  panel1.add(button1);  panel1.add(button2);  panel2.add(button3);  panel2.add(button4);  panel1.setBackground(Color.red);  panel2.setBackground(Color.cyan);  validate();  }  }  public class Ex5\_5  {  public static void main(String[] args)  {  new PanelTest();  }  5.3.2　界面布局策略  Java在java.awt包中定义了5种界面布局策略，它们分别是：FlowLayout、BorderLayout、CardLayout、GridLayout、GridBagLayout。  设置布局的格式为：  容器对象.setLayout(布局策略);  1、浮动布局FlowLayout  　　浮动布局是按照组件的顺序，用add方法将组件从左至右在一行排列，一行放不下时就自动换行。每行组件均按居中方式进行排列。这个布局方式我们在前面的示例中多次使用。  其设置的方法为：  　　　　setLayout(new FlowLayout());  2、边界布局BorderLayout（窗体容器的默认布局）  BorderLayout类把容器划分成5个区域，分别标记为“North”、　“South”、“West”、“East”和“Center”。每个组件用add方法放置到区域中。中间的区域的空间最大。  其设置的方法为：  　　　　setLayout(new BorderLayout());  import java.awt.\*;  import javax.swing.\*;  class BordTest extends JFrame  {  BordTest()  { setSize(300,200);  setVisible(true);  setTitle("边界布局示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new BorderLayout());  //将其他组件加入到窗体  add("East", new JButton("东"));  add("South", new JButton("南"));  add("West", new JButton("西"));  add("North", new JButton("北"));  add("Center", new JButton("中"));  validate();  }  }  public class Ex5\_6  { public static void main(String[] args)  { new BordTest(); }  }  3、网格布局GridLayout  GridLayout类以矩形网格形式对容器中的组件进行布局。容器被分成大小相等的单元格，单元格的大小由最大的构件所决定，用add方法将组件一行一行地从左至右放置到布局的每个单元格中。  其设置的方法为：  　　　　setLayout(new GridLayout(int row, int cols));  row是网格的行数，cols是网格的列数。  【例5-7】网格布局示例。  import java.awt.\*;  import javax.swing.\*;  class GridTest extends JFrame  {  GridTest()  {  setSize(300,200);  setVisible(true);  setTitle("网格布局示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new GridLayout(3,3));  for (int i=1;i<=9 ;i++ )  { add(new JButton("按钮"+i)); }  validate();  }  }  public class Ex5\_7  {  public static void main(String[] args)  { new GridTest(); }  }  4、卡片布局CardLayout  这种布局包含几个卡片，在某一时刻只有一个卡片是可见的，而且第一个卡片显示的内容可用自己的布局来管理。  CardLayout的布局可以包含多个组件，但是实际上某一时刻容器只能从这些组件中选出一个来显示，就象一叠“扑克牌”每次只能显示最上面的一张。如图所示。  卡片布局设置的方法为：  　　　　setLayout(new CardLayout());  卡片的顺序由组件对象本身在容器内部的顺序决定。CardLayout 定义了一组方法，这些方法允许应用程序按顺序地浏览这些卡片，或者显示指定的卡片。  CardLayout的主要方法：  first(Container parent) 显示容器parent中的第一张卡片。  next(Container parent) 显示容器parent中的下一张卡片。  previous(Container parent) 显示容器parent中的前一张卡片。  last(Container parent) 显示容器parent中的最后一张卡片。  【例5-8】卡片布局示例。  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  class CardTest extends JFrame implements ActionListener  {  JButton btn[] = new JButton[5];  CardLayout card = new CardLayout();  Panel p =new Panel();  CardTest()  {  setSize(300,200);  setVisible(true);  setTitle("卡片布局示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  add(p);  p.setLayout(card);  for (int i=1;i<=4 ;i++ )  {  btn[i] = new JButton("卡片"+i);  p.add(btn[i],"卡片标识"+i);  btn[i].addActionListener(this);  }  validate();  }    public void actionPerformed(ActionEvent e)  {  card.next(p);  }  }  public class Ex5\_8  {  public static void main(String[] args)  {  new CardTest();  }  }  重、难、疑点：  重、难点：事件处理。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章图形用户界面 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第七周 |
| 教学内容：  掌握组件的使用。  基本内容：  **5.4　 JComponent类组件的使用**  **5.4.1 JComponent类组件**  JComponent类是除顶层容器外的所有swing组件的基类，其常用子类见教材P115 表5.2  **5.4.2　文本组件和标签**  1、JTextComponent类  JTextComponent 是一个允许设置、检索和修改文本的类。它是 swing 文本组件的基类，通过它定义了三个子类： JTextField、JTextArea、JEditorPane。它们的继承关系如图所示。    JTextComponent类的常用方法 ：    由于JTextComponent是文本框JTextField 类、文本区JTextArea类及文本组件JEditorPane共同的父类。因此，下面要介绍的文本框JTextField 类、文本区JTextArea类及文本组件JEditorPane都继承了JTextComponent所具有的方法属性。  **2、文本框JTextField**  文本框JTextField 是对单行文本进行编辑的组件。它用来接受用户的输入码或显示可编辑的文本。  （1）文本框JTextField类的定义  （2）创建文本框  　　创建文本框时，一般要以初始的文本字符串或能容纳的字符数为参数：  **JTextField　text = new JTextField(String str);**  【例5-9】JTextField 类示例。  import javax.swing.\*;  import java.awt.FlowLayout;  public class TxtfdTest extends JFrame  {  JTextField txt; //声明文本框对象  TxtfdTest()  {  setSize(300,200);  setVisible(true);  setTitle("创建文本框示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout()); //设置窗体为浮动布局  txt=new JTextField(20); //对象实例化  add(txt); //将文本框添加到窗体中  validate();  txt.setText("重新设置了文本");//设置文本内容  }  }  public class Ex5\_9  {  public static void main(String[] args)  {  new TxtfdTest();  }  }  **3、密码框JPasswordField**  密码框JPasswordField 是JtextField的子类，允许编辑单行文本，可以设置为不显示键入的原始字符，而是显示指定的回显字符。  　　JpasswordField类的主要方法为  setEchoChar(char c)，  其中的字符c为回显字符。  【例5-10】设计一个密码验证程序。  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  class Passwd extends JFrame implements ActionListener  {  JLabel lb=new JLabel("请输入密码:");  JPasswordField txt1=new JPasswordField(25);  JButton bn=new JButton("确定");  JTextField txt2=new JTextField(25);  Passwd()  { setSize(300,200);  setVisible(true);  setTitle("密码验证");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout());  add(lb);  add(txt1);  txt1.setEchoChar('\*');  add(bn);  add(txt2);  validate();  bn.addActionListener(this);  }  public void actionPerformed(ActionEvent e)  {  if (txt1.getText().equals("abc"))  txt2.setText("密码正确!!");  else  txt2.setText("密码错误!!");  }  }  public class Ex5\_10  {  public static void main(String[] args)  {  new Passwd();  }  }  **4、文本区JTextArea**  文本区JTextArea是对多行文本进行编辑的组件，它能用控制符来控制文本的格式，例如：\n为换行，\t为插入一个tab字符。  （1）创建文本区  　　通常创建文本区时，要说明这个文本区的行、列数或文本内容：  　　　 JTextArea txt1 = new JTextArea(7,35);  （2）主要方法  　　主要继承JTextComponent 类的方法，从文本区的定义可知，它还有append(String str)等自己的方法。  【例5-11】JTextArea 类示例。  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  class AreaTest extends JFrame implements ActionListener  {  JTextArea txt1=new JTextArea(7,35);  JTextField txt2=new JTextField(35);  String str="窗外飘起蒙蒙小雨，\n平添一夜寒意，"+  "\n多少的思绪藏在心底,";  // 轻轻一声叹别离。  AreaTest()  {  setSize(400,200);  setVisible(true);  setTitle("文本组件示例");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout());  txt1.setText(str);  add(txt1);  add(txt2);  validate();  txt2.addActionListener(this);  }  public void actionPerformed(ActionEvent e)  {  String s=txt2.getText();  txt1.append("\n"+s);  }  }  public class Ex5\_11  {  public static void main(String[] args)  {  new AreaTest();  }  }  **6、标签Jlabel类**  标签是用户只能查看其内容但不能修改的文本组件，一般作说明之用。标签JLabel的常用方法见教材P120表5.4。  【例5-12】 创建一个包含带有图标的按钮和标签的窗体。  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  class LbTest extends JFrame implements ActionListener  {  LbTest(String s)  {  setSize(250,250);  setVisible(true);  setTitle(s);  setLayout(new FlowLayout());  ImageIcon icon1=new ImageIcon("s1.jpg");  ImageIcon icon2=new ImageIcon("s2.jpg");  ImageIcon icon3=new ImageIcon("s3.jpg");  JButton jbtn=new JButton("按钮",icon1);  jbtn.setRolloverIcon(icon3);  JLabel jlb=new JLabel("标签",icon2,SwingConstants.CENTER);  jlb.setToolTipText("QQ头像");  add(jbtn);  add(jlb);  jbtn.addActionListener(this);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  validate();  }  public void actionPerformed(ActionEvent e)  {  JInternalFrame in\_window;  in\_window =new JInternalFrame("内部窗体",true,true,true,true);  in\_window.setSize(250,200);  in\_window.setVisible(true);  add(in\_window);  JTextArea text=new JTextArea(5,15);  in\_window.add(text,BorderLayout.CENTER);  }  }  public class Ex5\_12  {  public static void main(String args[])  {  LbTest win=new LbTest("有图标的按钮和标签");  }  }  **5.4.3　单选按钮、复选框和下拉列表**  (1)单选按钮（JRadioButton）和复选框（JCheckBox）  【例5-13】 创建包含有单选按钮和复选框的窗体。  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  class BRTest extends JFrame implements ItemListener,ActionListener  {  JTextField text = new JTextField(15); ;  BRTest(String s)  {  setSize(300,200);  setVisible(true);  setTitle(s);  setLayout(new FlowLayout());  // 添加三个复选框  JCheckBox cb1 = new JCheckBox("C语言");  cb1.addItemListener(this);  add(cb1);  JCheckBox cb2 = new JCheckBox("C++语言");  cb2.addItemListener(this);  add(cb2);  JCheckBox cb3 = new JCheckBox("Java语言");  cb3.addItemListener(this);  add(cb3);  // 添加三个单选按钮  JRadioButton b1 = new JRadioButton("鲜花");  b1.addActionListener(this);  add(b1);  JRadioButton b2 = new JRadioButton("鼓掌");  b2.addActionListener(this);  add(b2);  JRadioButton b3 = new JRadioButton("鸡蛋");  b3.addActionListener(this);  add(b3);  // 定义按钮组,单选按钮只有放到按钮组中才能实现单选功能  ButtonGroup bg = new ButtonGroup();  bg.add(b1);  bg.add(b2);  bg.add(b3);  // 添加文本框  add(text);  validate();  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  }  public void itemStateChanged(ItemEvent ie)  {  JCheckBox cb = (JCheckBox)ie.getItem();  text.setText(cb.getText());  }  public void actionPerformed(ActionEvent ae)  {  text.setText(ae.getActionCommand());  }  }  //主类  public class Ex5\_13  {  public static void main(String[] args)  {  new BRTest("单选按钮和复选框示例");  }  }  重、难、疑点：  重难点：组件的使用。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章图形用户界面 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第七周 |
| 教学内容：  掌握组件的使用。  基本内容：  复习：  1、设计登录界面    2、设计颜色选择显示界面  屏幕剪辑  (2)下拉列表（JComboBox）  　　下拉列表（JComboBox）通常显示一个可选条目，允许用户在一个下拉列表中选择多个不同条目，用户也可以在文本区内键入选择项。JComboBox的构造函数如下：  JComboBox( )  JComboBox(Vector v)  其中，v 是初始选项。  　　要增加选项，则使用方法：  　　void addItem(Object obj)  其中，obj 是加入下拉列表的对象。  【例5-14】创建包括一个下拉列表和一个标签的窗体。标签显示一个图标。下拉列表的可选条目是“France”，“Germany”，“Italy”和“Japan”。当选择了一个图标，标签更新为这个国家的国旗。  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  class CobTest extends JFrame implements ItemListener  {  JLabel jlb;  ImageIcon france, germany, italy, japan;  CobTest(String s)  {  setSize(300,200);  setVisible(true);  setTitle(s);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setLayout(new FlowLayout());  JComboBox jc = new JComboBox();  jc.addItem("中国");  jc.addItem("俄罗斯");  jc.addItem("韩国");  jc.addItem("联合国");  jc.addItemListener(this);  add(jc);  jlb = new JLabel(new ImageIcon("中国.jpg"));  add(jlb);  validate();  }  public void itemStateChanged(ItemEvent ie)  {  String s = (String)ie.getItem().toString();  jlb.setIcon(new ImageIcon(s + ".jpg"));  }  }  //主类  public class Ex5\_14  {  public static void main(String args[])  { new CobTest("下拉列表示例"); }  }  5.4.4 **卡片选项页面（JTabbedPane）**  【例5-15】卡片选项页面示例。  1. /\* 卡片选项页面 \*/  2. import javax.swing.\*;  3. import java.awt.\*;  4. import java.awt.event.\*;  5. class TtpDemo extends JFrame  6. {  7. TtpDemo()  8. {  9. super("卡片选项页面示例");  10. setSize(300,200);setVisible(true);    11. JTabbedPane jtp = new JTabbedPane();  12. ImageIcon icon1 = new ImageIcon("c1.gif");  13. ImageIcon icon2 = new ImageIcon("c2.gif");  14. ImageIcon icon3 = new ImageIcon("c3.gif");  15. jtp.addTab("城市",icon1, new CitiesPanel(),"城市名称");  16. jtp.addTab("文学", icon2, new BookPanel(),"文学书目");  17. jtp.addTab("网站", icon3, new NetPanel(),"精选网址");  18. getContentPane().add(jtp);  19. validate();  20. addWindowListener(new WindowAdapter()  21. {public void windowClosing(WindowEvent e)  22. {System.exit(0);}});  23. }  24. }  25. //定义面板CitiesPanel  26. class CitiesPanel extends JPanel  27. {  28. CitiesPanel()  29. {  30. JButton b1 = new JButton("北京");  31. JButton b2 = new JButton("上海");  32. JButton b3 = new JButton("深圳");  33. JButton b4 = new JButton("厦门");  34. add(b1); add(b2); add(b3); add(b4);  35. }  36. }  37. //定义面板BookPanel  38. class BookPanel extends JPanel  39. {  40. BookPanel()  41 {  42. JCheckBox cb1 = new JCheckBox("西游记");  43. JCheckBox cb2 = new JCheckBox("三国演义");  44. JCheckBox cb3 = new JCheckBox("红楼梦");  45. add(cb1); add(cb2); add(cb3);  46. }  47. }  48. //定义面板NetPanel  49. class NetPanel extends JPanel  50. {  51. NetPanel()  52. {  53. JComboBox jcb = new JComboBox();  54. jcb.addItem("思维论坛");  55. jcb.addItem("百度搜索");  56. jcb.addItem("Java爱好者");  57. add(jcb);  58. }  59. }  60. //主类  61. public class Example5\_15  62. { public static void main(String args[])  63. { new TtpDemo(); }  64. }  重、难、疑点：  重难点：组件的使用。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章用户界面设计 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第8周 |
| 教学内容：  掌握组件使用  基本内容：  **5.4.5　滑杆（JSlider）和进度指示条（JProgressBar）**  【例5-16】滑杆和进度指示条配合使用示例。  1.　/\* 滑杆和进度指示条 \*/  2.　Import java.awt.\*;  3.　Import java.awt.event.\*;  4.　Import javax.swing.\*;  5.　Import javax.swing.event.\*;  6.　Import javax.swing.border.\*;  7. class P extends JPanel {  8. 　JProgressBar pb = new JProgressBar();  9. 　 JSlider sb = new JSlider(JSlider.HORIZONTAL, 0, 100, 60);  10. 　 public P() {  11. 　setLayout(new GridLayout(2,1));  12. 　add(pb);  13. 　sb.setValue(0);  14. 　sb.setPaintTicks(true);  15. 　sb.setMajorTickSpacing(20);  16. 　sb.setMinorTickSpacing(5);  17. 　sb.setBorder(new TitledBorder("移动滑杆"));  18. 　pb.setModel(sb.getModel());  19. 　pb.setStringPainted(true);  20. 　　add(sb);  21. 　 }  22.　｝  23. public class Example5\_16{  24. 　public static void main(String args[]) {  25. 　　 Frame f=new Frame("滑杆和进度指示条");  26. 　　 f.setSize(300,150);  27. f.add(new P());  28. f.show();  29. f.addWindowListener(new WindowAdapter()  30. {public void windowClosing(WindowEvent e)  31. {System.exit(0);}});  32. }  33. }  **5.4.6 表格（JTable）**  表格JTable常用方法  【例5-17】 利用JTable编制员工档案表  1. /\*　简单JTable表格 \*/  2. import javax.swing.\*;  3. import java.awt.\*;  4. import java.awt.event.\*;  5. class TableDemo extends JFrame  6. {  7. public TableDemo()  8. {  9. super("员工档案表");  10. String[] columnNames = {"姓名","职务","电话","月薪","婚否"};  11. Object[][] data = {  12. {"李 强","经理","059568790231",new Integer(5000),new Boolean(true)},  13. {"吴 虹","秘书","059569785321",new Integer(3500),new Boolean(true)},  14. {"陈卫东","主管","059565498732",new Integer(4500),new Boolean(false)},  15. {"欧阳建","保安","059562796879",new Integer(2000),new Boolean(true)},  16. {"施乐乐","销售","059563541298",new Integer(4000),new Boolean(false)}  17. };  18. JTable table = new JTable(data,columnNames);  19. table.setPreferredScrollableViewportSize(new Dimension(500,70));  20. JScrollPane scrollPane = new JScrollPane(table);  21. getContentPane().add(scrollPane, BorderLayout.CENTER);  22. addWindowListener(new WindowAdapter() {  23. public void windowClosing(WindowEvent e) {  24. System.exit(0);  25. }  26. });  27. pack();  28. setVisible(true);  29. }  30. }  31. //主类  32. public class Example5\_18  33. {  34. public static void main(String[] args)  35. {  36. TableDemo frame = new TableDemo();  37. }  38. }   1. 简单的计算器。 2. 复杂的计算器。 3. 日历。   重、难、疑点：  重、难点：组件使用。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第五章用户界面设计 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第八周 |
| 教学内容：   1. 掌握菜单的创建 2. 掌握对话框 3. 了解树   基本内容：  **5.5 菜单与对话框**   * + 1. **菜单**   (1) 一个菜单栏（JMenuBar）包含多个菜单，通过JFrame的setMenuBar方法加入到一个JFrame中。一个菜单栏可以包含任意多个菜单对象，通过Add方法来增加菜单对象。  (2) 一个菜单（JMenu）是菜单项的集合，并且有一个标题，这个标题出现在菜单上，当点击这个标题时，这些菜单项立即弹出。使用它自身的add方法，可以增加菜单项（JmenuItem）或菜单（JMenu）对象。  (3) 菜单项在菜单中表示一个选项，并且可以注册一个动作监听器（ActionListener），以产生动作事件。  与JMenu相关的构造函数和常用方法  建立菜单的步骤：  （1）创建菜单栏对象，并将菜单条对象添加到窗体中  JMenuBar mbar = new JMenuBar();  setJMenuBar(mbar); //窗体类Frame的方法  （2）创建菜单对象，并将菜单对象添加到菜单栏中  menu1=new JMenu(“File”);  menu2=new JMenu(“Edit”)  mbar.add(menu1);  mbar.add(menu2);  （3）创建菜单项对象，并将菜单项对象添加到相应的菜单中  mi1=new JMenuItem(“New”);  mi2=new JMenuItem(“Open”);  mi3=new JMenuItem(“Save”);  mi4=new JMenuItem(“Close”);  menu1.add(mi1);  menu1.add(mi2);  menu2.add(mi3);  menu2.add(mi4);  【例5-19】设计一个菜单程序。  这个程序包含菜单“文件”和“编辑”。菜单下又包含菜单项。“文件”菜单包含的菜单项为“新建文件”、“打开文件”和“退出”，“编辑”菜单包含的菜单项为“剪切”、“复制”、和“粘贴”。除了“文件”和“退出”菜单项外，其他的所有的菜单项功能都暂时被关闭。  public Example5\_19()  { super("菜单演示程序");  JMenu file = new JMenu("文件");  file.add(fileNew); fileNew.setEnabled(false);  file.add(fileOpen); fileOpen.setEnabled(false);  file.addSeparator();  file.add(fileExit); fileExit.setEnabled(true);  JMenu edit = new JMenu("编辑");  edit.add(editCut); editCut.setEnabled(false);  edit.add(editCopy); editCopy.setEnabled(false);  edit.add(editPaste); editPaste.setEnabled(false);  JMenuBar bar = new JMenuBar();  bar.add(file);  bar.add(edit);  setJMenuBar(bar);  fileExit.addActionListener(this);  setSize(250,200);  setVisible(true);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  }  public void actionPerformed(ActionEvent e)  {  if(e.getSource() == fileExit)  System.exit(0);    }  public static void main(String args[])  {  Example5\_19 f = new Example5\_19();  }  }  5.5.2 弹出式菜单  在窗体中单击鼠标右键弹出的菜单称为弹出式菜单，也称为快捷菜单。  【例5-20】在文本框中显示弹出式菜单项。  1 /\* 右键弹出式菜单 \*/  2 import java.awt.\*;  3 import java.awt.event.\*;  4 import javax.swing.\*;  5 public class Example5\_20 extends JFrame implements ActionListener  6 {  7 JPopupMenu popup = new JPopupMenu();  8 JTextField txt = new JTextField(10);  9 public Example5\_21()  10 {  11 super("右键弹出式菜单");  12 setSize(300,250);  13 setVisible(true);  14 setLayout(new FlowLayout());  15 add(txt);  16 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  17 JMenuItem m1 = new JMenuItem("菜单项1");  18 JMenuItem m2 = new JMenuItem("菜单项2");  19 JMenuItem m3 = new JMenuItem("菜单项3");  20 JMenuItem m4 = new JMenuItem("菜单项4");  21 popup.add(m1);  22 popup.add(m2);  23 popup.add(m3);  24 popup.addSeparator();  25 popup.add(m4);  26 m1.addActionListener(this);  27 m2.addActionListener(this);  28 m3.addActionListener(this);  29 m4.addActionListener(this);  30 addMouseListener(new MouseAdapter()  31 {  32 public void mouseClicked(MouseEvent e)  33 {  34 if (e.getButton()==MouseEvent.BUTTON3)  35 {  36 popup.show(e.getComponent(),  37 e.getX(),  38 e.getY());  39 }  40 }  41 });  42 validate();  43 }  44 public void actionPerformed(ActionEvent e)  45 {  46 txt.setText(  47 ((JMenuItem)e.getSource()).getText());  48 }  49 public static void main(String args[])  50 { new Example5\_20(); }  51 }  **5.5.3 对话框**  对话框（JDialog）是一个有边框、有标题的独立存在的容器，是一个从某个窗口弹出的特殊窗口。对话框与JFrame一样，不能被其他容器所包容，但是不能作为程序的最顶层容器，也不能包含菜单。JDialog必须隶属于一个JFrame窗口，并由这个JFrame窗口负责弹出。如它的父窗口JFrame消失，它也随之撤消。  **1、对话框的构造**  (1) “有模式”对话框（Medel dialog）：当这个对话框处于激活状态时，只让程序响应对话框内部的事件，阻塞它所隶属的父窗口对象的输入，而且它将阻塞其他线程的执行，直到该对话框被关闭。  (2) “无模式”对话框（Non-modal dialog）：这种对话框并不阻塞它所隶属的父窗口对象的输入，它可以与父窗口对象并存，除非特别声明，一般的对话框是“无模式”的。  JDialog类的构造方法及其含义  【例5-21】设计一个本对话框与窗口传递数据的程序。  源程序见教材P131  **2、消息对话框**  Java有一种与用户交互操作的特殊消息对话框JOptionPane类。  (1)消息对话框的构造方法  见教材P132表5.11  (2) 消息对话框的静态方法（表5.12）  例如：  显示消息对话框类型  JOptionPane.showMessageDialog(null, "提示的内容", "提示对话框", JOptionPane.ERROR\_MESSAGE);  【例5-22】消息对话框示例。  1. import javax.swing.\*;  2. public class Example5\_22  3. {  4. public static void main(String args[])  5. {  6. JOptionPane d\_input = new JOptionPane();  7. String str = d\_input.showInputDialog(null, "1+2=?");  8. if (str.equals("3"))  9. d\_input.showMessageDialog(null, "回答正确。");  10. else  11. d\_input.showMessageDialog(null, "回答错误!");  12. d\_input.showConfirmDialog(null, "测试完毕!");  13. System.exit(0); //退出程序;  14. }  15. }  **3、文件选择对话框**  实例化JFileChooser类后，调用showOpenDialog方法，能够打开一个文件选择对话框，可用于打开或保存文件时的文件选择，如图所示。  JFileChooser file = new JFileChooser( ) ;  file.showOpenDialog(null);  【例5-23】文件选择选择对话框示例。  1. import javax.swing.\*;  2. import java.awt.\*;  3.  4. public class Example5\_23  5. {  6. public static void main(String args[])  7. {  8. JOptionPane d\_input = new JOptionPane( );  9. JFileChooser file = new JFileChooser( );  10. file.showOpenDialog(null);  11. String str = file.getSelectedFile().toString();  12. d\_input.showMessageDialog(null, str);  13. }  14． }  **4、颜色选择对话框**  实例化JColorChooser类后，调用showOpenDialog方法，能够打开一个颜色选择对话框，如图所示。  JColorChooser color = new JColorChooser() ;  color.showDialog(null,"",null);  【例5-24】颜色选择对话框示例。  1. import javax.swing.\*;  2. import java.awt.\*;  3.  4. public class Example5\_24  5. {  6. public static void main(String args[])  7. {  8. JOptionPane d\_input = new JOptionPane();  9. JColorChooser color = new JColorChooser() ;  10. Color c = color.showDialog(null,"",null);  11. d\_input.showMessageDialog(null, c);  12. }  13. }  **5.6 树（JTree）**  **5.6.2 　树的构造方法**  **1、树的创建**  　　我们可以使用JTree类的构造方法来创建树。JTree的常用构造方法如下：  　　JTree()： 建立一个系统默认的树。  　　JTree(Hashtable value)：应用Hashtable表建立树，不显示根节点。  　　JTree(Object[] value)：应用数组Object[]建立树，不显示根节点。  　　JTree(TreeNode root)：应用节点TreeNode 建立树。  　　JTree(TreeNode root, boolean askAllowsChildren)： 应用节点TreeNode 建立树，并确定是否允许有子节点。  【例5-25】建立一个系统默认的简单树结构。  1. /\* 最简单的JTree示例 \*/  2. import java.awt.\*;  3. import javax.swing.\*;  4.　class TreeDemo extends JFrame  5.　{  6. 　public TreeDemo()  7. 　{  8. setSize(400,300);  9. setTitle("演示怎样使用JTree");  10. show();  11. JScrollPane jPanel = new JScrollPane();  12. getContentPane().add(jPanel);  13. //创建系统默认的树型对象  14. JTree jtree = new JTree();  15. //在面板上添加树型结构  16. jPanel.getViewport().add(jtree, null);  17. validate();  18. //点击"关闭窗口"按钮时，则退出  19. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  20. 　}  21.　}  22. //主类  23. public class Example5\_25  24. {  25. public static void main(String[] args)  26. {  27. TreeDemo frame = new TreeDemo();  28. }  29. }  **5.6.2 　树的构造方法**  树节点由javax.swing.tree包中的接口TreeNode定义，该接口是MutableTreeNode类的子类，而MutableTreeNode又由DefaultMutableTreeNode类实现，因此，在创建树时， 要使用DefaultMutableTreeNode类为该树创建节点。  DefaultMutableTreeNode类的常用构造方法是：  DefaultMutableTreeNode(Object userObject);  DefaultMutableTreeNode(Object userObject, boolean allowChildren);  应用节点TreeNode构造树的步骤如下：  （1）定义节点  DefaultMutableTreeNode n1=new DefaultMutableTreeNode("节点1");  DefaultMutableTreeNode n2=new DefaultMutableTreeNode("节点2");  DefaultMutableTreeNode n3=new DefaultMutableTreeNode("节点3");  …  （2）定义树，同时确定n1为根节点  JTree tree =new JTree(n1);  （3）添加子节点  n1.dd(n2);  n1.add(n3);  【例5-26】建立一个应用TreeNode构造如图结构的树。  **3、利用哈希表(Hashtable)构造树**  【例5-27】建立一个利用哈希表(Hashtable)数据构造树的结构。  源程序见P139，程序运行结果如图所示。  **5、处理节点事件**  树中的节点可以发生选择事件，即用鼠标单击节点时产生事件。一个树对象处理事件的接口是TreeSelectionListener，可以使用  　　树对象.addTreeSelectionListener(this);  方法获得一个监视器。  树对象通过使用方法getLastSelectedPathComponent() 获取选中的节点，使用方法getUserObject()得到与节点相关的信息。  【例5-28】处理节点事件  源程序见教材P140，程序运行结果如图所示。  重、难、疑点：  重点：菜单、对话框。  难点：树。 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第六章Java图形与事件处理 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第九周 |
| 教学内容：  掌握基本图形的绘制  基本内容：  6.1 图形与图形的描绘  6.1.1 坐标及绘图方法paint()  1、坐标  在一个二维的Java图形界面坐标系中，该坐标的原点在组件的左上角，坐标的单位是像素。X轴在水平方向从左至右，Y轴在垂直方向从上向下，如图所示。    2、绘图方法paint()方法  任何一个图形对象（java.awt.Component的子类如canvas、Panel、Frame）都可以使用paint()方法为绘图工具就可以画出线条、矩形、圆形等各种图形来。该方法为：  public void paint(Graphics g)；  paint()方法在程序执行后会被自行调用。而在程序中需要执行paint()方法，一般是调用repaint()方法，以清除旧图，重新绘制新图，该方法称为重绘。  一般以下情况，paint方法会自动被调用：  组件第一次显示时，窗口最大化最小化时，窗口大小发生变化时,repaint刷新组件外观时。  paint ：绘制容器。paintComponent : 绘制此容器中的每个组件  以上两个方法位于java.awt包里的Container类中。如果使用paint绘制图形，要调用super.paint(g)，以包含原组件、原边框和子组件的。  paint ：绘制容器。paintComponen : 绘制此容器中的每个组件paintBorder ：绘制组件的边框paintChildren ：绘制此组件的子组件  以上四个方法位于javax.Swing包中的JComponent类中  在Swing 中，组件绘制 paint() 方法会依次调用 paintComponent()，paintBorder()，paintChildren() 三个方法，所以Swing 编程时，如果继承 JComponent 或者其子类需要重绘的话，只要覆写 paintComponent() 方法而没必要覆写 paint()方法，因为 paintBorder()方法与paintChildren() 方法一般用默认即可。如果还要保留容器中的原本组件就别忘了调用super.paintComponent(g)。  3、画布  画布Canvas类是用来绘制图形的矩形组件，在画布中可以响应鼠标和键盘事件。  创建画布对象：Canvas can=new Canvas();调用setSize方法确定画布大小。画布对象常用方法：paint,update,repaint,当Canvas需要更新时，会自动调用repaint方法。    6.1.2 Graphics类  Graphics类是绘图对象，它是一个抽象类，不能直接创建Graphics对象。  1、绘图方法  Graphics类包含了大量的绘图方法，常用的绘图方法见教材表6.1。      【例6-1】 绘制直线、矩形和圆的简单图形。    【例6-2】 绘制一个多边形图形。    2、设置颜色  （1）颜色    （2）Color类  使用Graphics类设置颜色，首先要创建颜色类Color的对象，创建颜色对象的构造方法为：  public Color(int r, int g, int b);  其中，整型参数r、g、b的取值范围为0～255，分别代表红、绿、蓝三基色。  屏幕剪辑  也可以使用Color类的静态常量作颜色对象的参数，这时，创建颜色对象的构造方法为：  public Color(Color.颜色静态常量 );  其颜色静态常量取值为red(红色)、blue(蓝色)、green(绿色)、orange(橙色)、cyan(青绿色)、yellow(黄色)、pink(粉红色)、white(白色)、black(黑色)等。  Graphics类可以使用表6.3所示的get/set方法控制绘图的色彩和使用不同字体。    【例6-3】 绘制用色彩填充的笑脸图形。    练习：绘制奥运五环    重、难、疑点：  重点：  基本图形的绘制。  难点：  基本图形的绘制 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第六章Java图形与事件处理 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第九周 |
| 教学内容：  掌握事件处理。  基本内容：  6.1.2 Graphics类  设置字体    其中name为逻辑字体名，style(风格)是Font.PLAIN(正常字体)、Font.BOLD(黑体)或Font.ITALIC(斜体)的组合，size是字号大小，字号越大字体越大。  我们可以用Graphics 类的setFont(Font f)方法来设置字体。  【例6-4】编写程序以创建不同的风格和大小的可利用的逻辑字体。    6.1.3 Java 2D  graphics类还有一些不足，例如缺少改变线条粗细的方法，也缺少填充一个对象的方法。Graphics 2D可以解决这些问题。  1、控制线条的粗细  首先使用Line2D类创建直线对象:  Line2D line=new Line2D.Double(50,50,120,50);  再使用BasicStroke类创建一个供画笔paint()方法选择线条粗细的对象。  public BasicStroke(float width, int cap, int join);  width - 画笔线条粗细  cap – 线条两端的形状  join – 线条中角的处理  【例6-5】设置线条粗细    2、填充图形  　　Graphics2D对象调用fill()方法用颜色填充图形。  Graphics2D对象还可以通过GradientPaint类定义一个颜色对象，实现渐变颜色填充图形。GradientPaint类的构造方法为：    【例6-6】用渐变颜色填充图形。    6.1.4 图形应用程序设计实例  在进行图形应用程序设计时，经常把要实现的功能单独设计为一个类，而把显示这个功能设计成另一个类。实现功能的类称为业务逻辑层，而显示功能的类称为表现层。这样，把逻辑层和表现层分开，有利于实现代码重用。  【例6-7】通过一个窗体的文本框输入多边形的边数，随机产生多边形各折线的坐标位置，在画布上绘制出一个多边形。以此例说明，逻辑层和表现层间的数据传递。    【例6-8】设计一个在窗体中可上下左右移动的小方块的程序。    **改写程序，通过键盘移动矩形**  6.2 事件处理  6.2.1 事件处理机制  1、在事件处理的过程中，主要涉及三类对象，它们分别是：  （1）Event事件：用户的操作，以类的形式出现，例如键盘操作对应的事件类是KeyEvent。  （2）Event Source事件源：能够产生事件的对象，事件发生的场所，通常就是各个组件，例如按钮、文本框等。  （3）Event handler事件处理者：需要一个对象对事件进行监视，以便对发生的事件做出处理，也称为监听器。事件源通过调用相应的方法注册到监听器。  2、事件处理部分的基本过程：  定义监听器类，它是一个实现相应监听接口的类。这个类既可以是包含事件源的类，也可以是其它类或者内部类或者匿名类。  事件源组件注册监听器。  6.2.2 事件类  1、低级事件与高级事件  低级事件：KeyEvent（ 键盘事件：键按下、释放） ；MouseEvent（ 鼠标事件：鼠标单击，移动） 。  高级事件 ：ActionEvent（动作事件：按钮按下，TextField中按Enter键等）  2、事件监听器  每类事件都有对应的事件监听接口，如：  MouseEvent --> MouseListener ,  KeyEvent --> KeyListener等。  3、 事件类别及监听器接口 （见教材P175）    6.2.3 鼠标事件  与鼠标的事件有关可以分为两类：  (1)主要针对鼠标的坐标位置进行检测，使用MouseListener接口；方法有：单击、进入、离开、按下、释放。  (2)主要针对鼠标的拖曳状态进行检测，使用MouseMotinListener接口。方法有：拖动、移动。  【例6-9】事件处理示例。    6.2.4　事件适配器  1、事件适配器（EventAdapter）  　　java.awt.event包中定义的事件适配器类包括以下几个：  (1)　　ComponentAdapter( 接收组件事件适配器)  (2)　　ContainerAdapter( 接收容器事件适配器)  (3)　　FocusAdapter( 接收焦点事件适配器)  (4)　　KeyAdapter( 接收键盘事件适配器)  (5)　　MouseAdapter( 接收鼠标事件适配器)  (6)　　MouseMotionAdapter( 接收鼠标运动事件适配器)  (7)　　WindowAdapter( 接收窗口事件适配器)  2、用内部类实现事件处理  　　内部类（inner class）是被定义于另一个类中的类，使用内部类的主要原因是由于：  (1) 一个内部类的对象可访问外部类的成员方法和变量，包括私有的成员。  (2) 实现事件监听器时，采用内部类、匿名类编程非常容易实现其功能。  (3) 编写事件驱动程序，内部类很方便。  【例6-10】使用事件适配器及内部类设计记录鼠标位置的程序。  3、用匿名类实现事件处理  当一个内部类的类声明只是在创建此类对象时用了一次，而且要产生的新类需继承于一个已有的父类或实现一个接口，才能考虑用匿名类，由于匿名类本身无名，因此它也就不存在构造方法，它需要显式地调用一个无参数的父类的构造方法，并且重写父类的方法。所谓的匿名就是该类连名字都没有，只是显式地调用一个无参的父类的构造方法。  【例6-11】 使用匿名类设计记录鼠标位置的程序。  6.2.5 键盘事件  在Java中，当用户使用键盘进行操作时，就会产生KeyEvent事件。监听器要完成对事件的响应，就要实现KeyListener接口，或者是继承KeyAdapter类，实现对类中方法的定义。  在KeyListener接口中有如下3个事件：  (1)KEY\_PRESSED:键盘按键被按下所产生的事件。  (2)KEY\_RELEASED:键盘按键被释放所产生的事件。  (3)KEY\_TYPED:键盘按键被点击所产生的事件。  在实现接口时，对应的上面3个事件的处理方法是：  keyPressed(Event e);  keyReleased(KeyEvent e);  keyTyped(keyEvent e);  【例6-12】设计一个程序，在界面上安放3个按钮，用户可以通过按动键盘上的方向键移动这些按钮组件。    6.2.6 焦点事件  用户对一个组件进行操作，先要让其获得焦点。例如文本框得到焦点时会出现闪烁光标， 而按钮则在按钮标签周围显示矩形框等。  在组件得到或失去焦点时会产生FocusEvent事件。 要处理这些事件使用焦点监听器: FocusListener接口(或缺省适配器类是FocusAdapter)。  它的方法有:    void focusGained(FocusEvent e)  获得键盘焦点     void focusLost(FocusEvent e)  失去键盘焦点  【例6-13】设计一程序，可以用Tab鍵改变文本框的焦点。    【例6-14】设计一个由3段数值组成的序列号输入程序，演示从组件中转移焦点。    重、难、疑点：  重点：事件处理  难点：事件处理 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第六章Java图形绘制与事件处理 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
|  |
| 教学内容：  掌握图形绘制与事件处理  基本内容：  1、绘制时钟  2、贪食蛇  3、验证码  4、围棋  重、难、疑点：  重点：图形绘制与事件处理  难点：图形绘制与事件处理 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第七章异常处理与线程 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第十一周 |
| 教学内容：  掌握异常处理。  基本内容：  7.1 异常处理  异常（Exception）指程序运行过程中出现的非正常现象，例如用户输入错误、需要处理的文件不存在、在网络上传输数据但网络没有连接等。  7.1.1 Java的异常处理机制  在Java中，把异常分为错误（Error）与异常（Exception）两大类。  错误（Error）包含一些严重的、程序不能处理的系统错误，如内存不足，虚拟机错误，栈溢出等，由系统处理，不由程序处理，当然也包括一些非法的情形，由程序员自行检查代码排除，此类异常本章不作介绍。  异常（exception）指非致命性错误，一般指在运行程序时硬件和操作系统是正常的，而程序遇到的运行错，如整数进行除法运算时除数为0，操作数超出数据范围，打开一个文件时发现文件不存在，网络连接中断等。  Exception类有自己的方法和属性。它的构造方法有两个：  public Exception ( )；  public Exception (String s)；  7.1.1 Java的异常处理机制  系统定义的运行异常    在进行Java应用程序设计时，可以使用类库中已经定义好的异常类。系统预定义的类库有时候不能满足用户需要时，程序员编写应用程序时可以自定义需要的异常类。自定义异常类必须继承于已有的异常类，即用户自定义的异常类都必须直接或间接地是Exception类的子类。  7.1.2　异常的抛出  Java应用程序在运行时如果出现了一个可识别的错误，就会产生一个与该错误相对应的异常类的对象，这个过程称为异常的抛出。  1、系统自动抛出的异常  【例7-1】创建一个有错误的程序，测试异常抛出的情况。  1.　 /\* 测试除数为0时抛出的异常 \*/  2.　class Example7\_1  3.　　{  4. 　　 public static void main(String[] args)  5. 　　{  6. int a=5,d=0;  7. 　System.out.println(a/d);  8. 　　}  9.　　}  7.1.2　异常的抛出  2、人为声明抛出异常 ，但不处理异常，异常处理交由调用本方法的主体处理。  用throw语句抛出异常对象的语法格式为：  　修饰符　返回类型　方法名（）　throws 异常类名  ｛  　　　….  throw 异常类名;  ….  }  7.1.3　异常处理  异常处理的方法有二种：一种方法是使用try…catch…finally结构对异常进行捕获和处理；另一种方法是通过throws和throw抛出异常。  try…catch…finally结构对异常进行捕获和处理的形式:  try{  可能出现异常的程序代码  ｝  catch(异常类1　变量1)  {　异常类1对应的异常处理代码　}  　catch(异常类2　变量2)  {　异常类2对应的异常处理代码　}  　...  　[finally]  { 无论异常是否发生都要执行的代码　}  【例7-2】构造一个异常，计算分母为a-5的分式，当输入数字5时，则抛出异常。      【例7-4】数组下标越界引发异常  class Example7\_4  {  　 public static void main(String[] args)  　{  　int a[]={1,2,3,4,5};  　int sum=0;  　try {  for (int i=0;i<=5 ;i++ )  {sum+=a[i];}  System.out.println("sum="+sum);  }catch(ArrayIndexOutOfBoundsException e)  　{ System.out.println("发生异常原因："+e); }  　 finally { System.out.println("程序运行结束！"); }  　 }  }  练习  利用异常处理机制建立一个有效的用户注册程序，要求注册时输入新注册的用户名，但如果用户名已经被别人注册，需要提示用户名已存在，不管是否注册成功都提示当前注册的人数。    重、难、疑点：  重点：异常处理  难点：异常处理 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第七章异常处理与线程 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第十一周 |
| 教学内容：  掌握线程。  基本内容：  7.2 多线程的基本概念  7.2.1进程与线程  1、进程  在一个操作系统中，每个独立执行的程序都可称之为一个进程，也就是“正在运行的程序”。它指的是从代码加载、执行到执行结束这样一个完整的过程。  目前大部分计算机上安装的都是多任务操作系统，即能够同时执行多个应用程序。  在计算机中，所有的应用程序都是由CPU执行的，对于一个CPU而言，在某个时间点只能运行一个程序，也就是说只能执行一个进程。  2、线程  每个运行的程序都是一个进程，在一个进程中还可以有多个执行线路“同时”运行，这些执行线路可以看做程序执行的一条条线索，被称为线程。  操作系统中的每一个进程中都至少存在一个线程。当一个Java程序启动时，就会产生了一个进程，该进程中会默认创建一个线程，在这个线程上会运行main()方法中的代码，该线程称之为主线程。除主线程外，线程无法自行启动，必须通过其他程序来启动它。  3、线程与进程的区别：  可否独立运行？内存空间？  线程不能自动运行，而必须存在于某一进程中，由进程触发执行。属于同一进程的所有线程共享进程的系统资源。   1. 多线程   前面章节所接触过的程序中，代码都是按照调用顺序依次往下执行，没有出现两段程序代码（两条执行线路）交替运行的效果，这样的程序称作单线程程序。  如果希望程序中实现多段程序代码交替运行的效果，则需要创建多个线程，即多线程程序。多线程程序在运行时，每个线程之间都是独立的，它们可以并发执行。  多线程的应用：火车站的售票系统  流媒体的下载与播放  实时更新网站的在线人数  7.2.2 线程的生命周期  线程要经历创建、就绪、运行、阻塞和死亡等5个状态 ，称为生命周期。  1、创建状态  当我们通过new命令创建了一个线程对象，则该线程对象就处于创建状态。如下面语句所示：  Thread thread1 = new Thread();  创建状态是线程已被创建但未开始执行的一个特殊状态。此时线程对象拥有自己的内存空间，但没有分配CPU资源，需通过start()方法调度进入就绪状态等待CPU资源。  7.2.2 线程的生命周期  2、就绪状态  处于创建状态的线程对象通过start()方法进入就绪状态，如下面语句所示：  Thread thread1 = new Thread();  Thread1.start();  当一个线程处于就绪状态时，系统为这个线程分配了它需的系统资源。start()方法同时调用了线程体，也就是run()方法，表示线程对象正等待CPU资源，随时可被调用执行。  7.2.2 线程的生命周期  3、运行状态  若线程处于正在运行的状态，表示线程已经拥有了对处理器的控制权，其代码目前正在运行，除非运行过程中控制权被另一优先级更高的线程抢占，否则这一线程将一直持续到运行完毕。使用yield()方法可以使线程主动放弃CPU控制权。线程也可能由于执行结束或执行stop()方法放弃控制权进入终止状态。  4、阻塞状态（不可运行状态（non Runnable）  调用了sleep()方法；调用了suspend()方法；该线程正在等待I/O操作完成；调用wait()方法； 输入输出流中发生线程阻塞。  如果一个线程处于阻塞状态，那么该线程则无法进入就绪队列。处于阻塞状态的线程通常必须由某些事件唤醒。处于阻塞态的线程序回到可运行态，有以下几种情况：  如果线程调用sleep()方法进入了休眠状态，不能调用任何方法让它脱离阻塞状态，只能等待指定的时间之后，自动脱离阻塞态。  如果线程为了等待一个条件变量而调用了wait()方法进入了阻塞态，需要这个条件变量所在的那个对象调用notify(()或notifyAll()方法。如果一个线程调用suspend()方法被挂起而进入了阻塞状态，必须在其他线程中调用resume()方法。  如果线程由于等待I/O而进入了阻塞状态，只能等待这个I/O操作完成之后，系统调用特定的指令来使该线程恢复可运行状态。  5、死亡状态  死亡状态（或终止状态），表示线程已退出运行状态，并且不再进入就绪队列。  线程的终止一般可通过两种方法实现：自然撤消（线程执行完）或是被停止（调用stop()方法）。线程一旦进入终止状态就不再存在了，也无法改变为其它状态。目前不推荐通过调用stop()来终止线程的执行，而是让线程执行完。  7.3 线程的创建方法  在Java语言中，可采用两种方式产生线程：  创建Thread子类，JDK中提供了一个线程类Thread，通过继承Thread类，并重写Thread类中的run()方法，然后通过该子类创建线程，调用start()方法,系统自动运行run()方法的方法体，便可实现多线程。  生成一个类，声明实现Runnable接口，并重写run()方法，然后再创建该实现类的对象，使用该对象作为Thread参数创建线程，调用start()方法,系统自动运行run()方法的方法体，便可实现多线程。  7.3.1创建Thread子类构造线程  我们可以通过继承Thread类，建立一个Thread类的子类并重新设计（重载）其run()方法来构造线程。  Public class TestThreadextends Thread{  public TestThread(String name) {  super(name);  }  publicvoid run() {  for(int i = 0;i<5;i++){  for(long k= 0; k <100000000;k++);  System.out.println(this.getName()+" :"+i);  }  }  publicstaticvoid main(String[] args) {  Thread t1 = new TestThread("阿三");  Thread t2 = new TestThread("李四");  t1.start();  t2.start();  }  }  【例7-5】创建二个Thread类的子类，然后在另一个类中建立这2个Thread类的对象来测试它，看具体会发生什么现象。  这个例子说明了这样几个事实：  (1)创建独立执行线程比较容易，Java负责处理了大部分细节。  (2)各线程并发运行，共同争抢CPU资源，线程抢夺到CPU资源后，就开始执行，无法准确知道某线程能在什么时候开始执行。  (3)线程间的执行是相互独立的。  (4)线程独立于启动它的线程（或程序）。  7.3.2 实现Runnable接口构造线程  Public class DoSomething implements Runnable {  private String name;  public DoSomething(String name) {  this.name = name;  }  publicvoid run() {  for (int i = 0; i < 5; i++) {  for (long k = 0; k < 100000000; k++) ;  System.out.println(name + ": " + i);  } } }  Public class TestRunnable {  publicstaticvoid main(String[] args) {  DoSomething ds1 = new DoSomething("阿三");  DoSomething ds2 = new DoSomething("李四");  Thread t1 = new Thread(ds1);  Thread t2 = new Thread(ds2);  t1.start();  t2.start();  }  }  两种创建线程的优点：  使用Runnable接口  还可以从其他类继承，可以避免由于Java的单继承带来的局限性。  适合多个相同程序代码的线程去处理同一个资源的情况  把线程同程序代码、数据有效的分离，很好的体现了面向对象的设计思想。  直接继承Thread类  不能再从其他类继承  编写简单，可以直接操纵线程，无需使用  Thread.currentThread()  【例7-6】创建一个实现Runnable接口的线程类，然后在另一个类中建立2个线程对象来测试它，看具体会发生什么现象。  【例7-7】 我们用Thread子类程序来模拟航班售票系统，实现四个售票窗口发售某班次航班的100张机票，一个售票窗口用一个线程来表示。  【例7-8】 用Runnable接口程序来模拟航班售票系统，实现四个售票窗口发售某班次航班的100张机票，一个售票窗口用一个线程来表示。  【例7-9】 设计一个多线程的应用程序，模拟一个台子上有多个弹子在上面滚动。“弹子”在碰到“台子”的边缘时会被弹回来。  线程的暂停和恢复  Thread.sleep(long millis)和Thread.sleep(long millis, int nanos)静态方法强制当前正在执行的线程休眠（暂停执行），以“减慢线程”。当线程睡眠时，它入睡在某个地方，在苏醒之前不会返回到可运行状态。当睡眠时间到期，则返回到可运行状态。如下面语句所示：  Thread thread1 = new Thread();  thread1.start();  try{ thread1.sleep(2000); }  catch(InterruptedException e) { }  (2)yield()方法  线程让步可以通过yield()方法来实现，该方法和sleep()方法有点相似，都可以让当前正在运行的线程暂停，区别在于yield()方法不会阻塞该线程，它只是将线程转换成就绪状态，让系统的调度器重新调度一次。当某个线程调用yield()方法之后，只有与当前线程优先级相同或者更高的线程才能获得执行的机会。  线程的优先级  优先级越高的线程获得CPU执行的机会越大，而优先级越低的线程获得CPU执行的机会越小。线程默认优先级是5。  线程的优先级用1~10之间的整数来表示，数字越大优先级越高。  除了可以直接使用数字表示线程的优先级，还可以使用Thread类中提供的三个静态常量表示线程的优先级  程序在运行期间，处于就绪状态的每个线程都有自己的优先级，例如main线程具有普通优先级。然而线程优先级不是固定不变的，可以通过Thread类的setPriority(int newPriority)方法对其进行设置，该方法中的参数newPriority接收的是1~10之间的整数或者Thread类的三个静态常量。  (3)wait()和notify()方法  7.4 线程同步  由于多线程要共享内存资源，因此有可能一个线程正在使用某个资源，而另一个线程却在更新它，这样，会造成数据的不正确。因此对于多个线程共享的资源，必须采取措施，使得每次只有一个线程能使用它，这就是多线程中的同步(synchronization)问题。  7.4.1使用多线程造成的数据混乱  【例7-10】 设计一个模拟用户从银行取款的应用程序。设某银行帐户存款额的初值是2000元，用线程模拟两个用户从银行取款的情况。  7.4.2 同步线程解决办法  1、Synchronized方法  声明Synchronized方法的一般格式为：  public synchronized 返回类型 方法名()  {  　　　　　　…　　/\* 方法体 \*/  }  同步代码块  synchronized(this)  {  　　　　　　代码块;  }  【例7-11】 改写例7-10，用线程同步的方法设计用户从银行取款的应用程序。  2、管程及wait()、notify()方法  【例7-12】设计一个模拟车辆通过交通路口的程序。  线程同步练习-模拟排队买票  张先生和李先生买电影票，售票员只有两张5元的钱，电影票5元一张。张先生用一张20元的人民币排在李先生的前面买票，而李先生用一张5元的人民币买票。通过编程模拟排队买票的情形。  存数和取数问题  重、难、疑点：  重点：线程  难点：线程 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第八章文件和输入输出流 | 授课类型 | 机房上课 |
| 授课时间 | 第1次 |
| 第十二周 |
| 教学内容：  1、掌握File类。  2、掌握字节输入输出流。  基本内容：  8.1　输入/输出流  输入输出是计算机的最基本操作，也是程序设计语言的一项重要的基本功能。例如从键盘输入数据，从文件中读取数据或向文件写入数据，通过网络上传或下载数据等。  Java与其他编程语言一样，文件操作在程序设计中占有很重要的地位，Java的文件操作是通过它的输入输出类库来实现的，Java.io包提供了丰富的流操作类。  流是一可被顺序访问的数据序列，是对计算机输入数据和输出数据的抽象。就方向来说分输入流和输出流。输入流将外部数据引入到计算机中，如从磁盘中读取信息等。输出流将数据引导到外部设备，如向磁盘保存文件等。因此“流”可以看成是数据从一种设备流向另一种设备的过程。  流序列中的数据既可以是未经加工的原始二进制数据，也可以是经一定编码处理后符合某种格式规定的特定数据，所以Java 中的数据流有字节流和字符流之分。字节流分别用java.io.InputStream和java.io.OutputStream表示，字符流的输入输出流分别用java.io.Reader和java.io.Writer表示  字节流与字符流主要的区别是它们处理数据的类型不同，二进制数据的字节流是最基本的数据表示方式，按每8位一个字节的方式来处理数据。字符流是按虚拟机的每16位一个字符的Unicode码来处理字符数据，其内部要进行字符型数据与字节型数据之间的字符集转化。  字节流转换为字符流时：  Publilc String(byte bytes[],String s)  字符流转换为字节流时：  Byte[] String.getBytes(String s);  8.1.2　io类库  1、字节流的层次结构    在JDK中，InputStream提供了一系列与读数据相关的方法，具体如下：    在JDK中，OutputStream提供了一系列与 写数据相关的方法 ，具体如下：    2、字符流的层次结构    8.2.1 文件与目录管理  1、创建File类文件对象  （1）所有对文件的操作都要使用File类.  （2）其构造：  File(“文件名”);  File(“文件所在目录”，“文件名”）；  2、获取文件及目录属性  （1）判断文件或目录是否存在  public boolean exists( ) ;  （2）判断是文件还是目录  public boolean isFile( );  public boolean isDirectory( );  （3）获取文件或目录名称与路径  public String getName( ); 返回文件名或目录名。  public String getPath( ); 返回文件或目录的路径。  （4）获取文件的长度  public long length( );  （5）获取文件读写属性  public boolean canRead( );  public boolean canWrite( );  （6）列出目录中的文件  public String[ ] list ( );  （7）比较两个文件或目录  public boolean equals( File f );  列出目录下的所有文件  import java.io.File;  public class Example26 {  public static void main(String[] args) throws Exception {  File file = new File("C:\\Users\\dcx\\Desktop\\大创"); // 创建File对象  if (file.isDirectory ()) { // 判断File对象对应的目录是否存在  String[] names = file.list (); // 获得目录下的所有文件的文件名  for (int i=0;i<names.length;i++) { System.out.println(names[i]); // 输出文件名  }  }  }  }  import java.io.File;  import java.io.FilenameFilter;  public class Example27 {  public static void main(String[] args) throws Exception {  // 创建File对象  File file = new File("C:\\Users\\dcx\\Desktop\\大创");  // 创建过滤器对象  FilenameFilter filter = new FilenameFilter() {  // 实现accept()方法  public boolean accept(File dir, String name) {  File currFile = new File(dir, name);  // 如果文件名以.java结尾返回true，否则返回false  if (currFile.isFile() && name.endsWith(".doc")) {  return true;  } else {  return false;  }  }  };  if (file.exists()) { // 判断File对象对应的目录是否存在  String[] lists = file.list(filter); // 获得过滤后的所有文件名数组  for (String name : lists) {  System.out.println(name);  }  }  }  }  3、文件及目录操作  File类中还定义了一些对文件或目录进行管理、操作的方法，常用的方法有：  （1）重命名文件  public boolean renameTo( File newFile );  （2）删除文件  public void delete( );  （3）创建目录  public boolean mkdir( );  import java.io.\*;  public class Example30 {  public static void main(String[] args) {  File file = new File("E:\\cn"); // 创建一个代表目录的File对象  deleteDir(file); // 调用deleteDir删除方法  }  public static void deleteDir(File dir) {  if (dir.exists()) { // 判断传入的File对象是否存在  File[] files = dir.listFiles(); // 得到File数组  for (File file : files) { // 遍历所有的子目录和文件  if (file.isDirectory()) {  deleteDir(file); // 如果是目录，递归调用deleteDir()  } else {  // 如果是文件，直接删除  file.delete();  }  }  // 删除完一个目录里的所有文件后，就删除这个目录  dir.delete();  }  }  }  8.2.2 文件流  对文件进行输入输出处理的四个类：  FileInputStream　：字节文件输入流；  FileOutputStream　：字节文件输出流；  FileReader　：字符文件输入流；  FileWriter　：字符文件输出流。  1、字节文件输入流读取文件  （1）FileInputStream类  FileInputStream(String filename)；  FileInputStream(File file)；  FileInputStream( FileDescriptor fdObj)；  （2）从文件输入流中读取字节  int read( )；  int read( byte b[ ])；  int read( byte b[ ],int off, int len)；  【例8-1】在下面的程序中，读取一个文本文件test.dat，并将其显示到对话框上。  在编写程序之前，我们至少需要知道两件事：  （1）怎样把一个流与一个文件联系起来；  （2）用什么方法把从文件中读取到的数据显示到一个对话框中。  2、字节文件输出流写入文件  （1）FileOutputStream类  FileOutputStream( String filename )  FileOutputStream( File file )  FileOutputStream(FileDescriptor fdObj)  （2）把字节发送到文件输出流  write(int b);  write(byte[ ] b);  write(byte[ ] b, int off, int len);  【例8-2】复制图像文件a.jpg，并且更名为b.jpg。  3、字符文件流读写文本文件  （1）FileReader和FileWriter  FileReader( String filename )；  FileReader( File file )；  FileReader(FileDescriptor fdObj)；  FileWriter( String filename )；  FileWriter( File file )；  FileWriter(FileDescriptor fdObj)；  重、难、疑点：  重点：File类与字节输入输出流的应用  难点：File类与字节输入输出流的应用 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第八章文件与输入输出 | 授课类型 | 机房上课 |
| 授课时间 | 第2次 |
| 第十二周 |
| 教学内容：   1. 掌握字符输入输出流。 2. 掌握其它输入输出流。   基本内容：  8.2.2 文件流  字符文件流读写文本文件  （2）用字符流进行读写操作的方法  与字节输入输出流的功能一样，Reader和Writer只是建立一条通往字符文件的通道，而要实现对字符数据的读写操作，还需要读方法和写方法来完成。  从输入流中按行读取字符的方法：  String readLine()；  向输出流写入多个字符的方法：  write(String s, int off, int len)；  将指定的字符串s从偏移量 off 开始的 len 个字符写入文件输出流。  刷新BufferedWriter所建立的缓冲区，一次性将缓冲区中的数据写入文件的方法如下：  flush()；  【例8-3】将文件a.txt读取到文本区，经修改后，另存为b.txt。  8.3　随机存取文件和本地可执行文件  8.3.1随机存取文件流RandomAccessFile类  随机存取文件流RandomAccessFile类可以读写文件中任意位置上的字节、文本等数据。它有二个构造方法：  （1 ）RandomAccessFile ( String filename, String mode )  （2 ）RandomAccessFile ( File file, String mode )  【例8-4】用随机存取文件流读写文件。  1．　/\* 随机流 \*/  2．　import java.io.\*;  3．　class Example8\_4  4．　{  5． public static void main(String[] args)  6． {  7． 　try{  8. RandomAccessFile f = new andomAccessFile("a.txt","rw");  9. 　　f.writeBytes("Zhang si ming");  10. 　　f.close();  11. 　　}  12. 　catch(IOException e){  13. 　　　System.out.println(e);  14. 　　}  15. }  16.　} 改写：实现文件的的追写。  【例8-5】使用随机存取文件流RandomAccessFile实现一个英汉小词典程序。  8.3.2　本地可执行文件  在Java语言中，使用java.lang包中的Runtime类可以运行本地机的可执行文件。每个 Java 应用程序都有一个 Runtime 类实例对象，使应用程序能够与其运行的环境相连接。但应用程序不能创建自己的 Runtime 类实例对象，而要通过该类的静态方法getRuntime()创建Runtime 类对象。  Runtime 类对象有几个常用方法：  （1 ）exit(int status)  通过启动虚拟机的关闭序列，终止当前正在运行的 Java 虚拟机。  （2 ）gc()  运行垃圾回收器。  （3 ）getRuntime()  （4 ）exec(String command)  调用该方法可以在单独的进程中运行由字符串命令指定的本地机上的可执行文件。  【例8-6】使用Runtime 类对象运行例8-5。  1.　public class Example8\_6  2.　{ public static void main(String args[])  3. 　 {  4.　　　　try{  5. 　 Runtime rt=Runtime.getRuntime();  6. 　 rt.exec("java Example8\_5");  7. 　 }  8. 　 catch(Exception e){ System.out.println(e); }  9. 　 }  10.　}  【例8-7】调用Windows系统自带的计算器。  1.　public class Example8\_7  2.　{ public static void main(String args[])  3. 　 {  4.　　　　try{  5. 　 Runtime rt=Runtime.getRuntime();  6. 　 rt.exec("c:/windows/system32/calc.exe");  7. 　 } catch(Exception e){ System.out.println(e); }  8. 　 }  9.　}  【例8-8】应用Runtime类设计一个Java语言简易编译器，该编译器具有编写源程序、编译和运行程序的功能。  8.4　数据流与对象流  有许多应用程序需要将处理的数据作为Java的一种基本类型（如布尔型，字节，整数和浮点数）来使用。这就要用到数据流类DataInputStream类和DataOutputStream类。这两个数据流类是很有用处的，它们允许程序按机器无关的格式读取Java原始数据。  8.4.1　数据流  用下面的构造方法就可以建立DataInputStream类和DataOutputStream类的实例：  public DataInputStream(InputStream in);  public DataOutputStream(OutputStream out);  【例8-9】应用DataInputStream类和DataOutputStream类复制声音文件。  8.4.2 对象流  Java可以将对象作为一个整体通过对象流进行传输和存储。  1、对象流的构造方法  ObjectInputStream类和ObjectOutputStream类的构造方法为：  ObjectInputStream(InputStream in);  ObjectOutputStream(OutputStream out);  【例8-10】编写程序，在窗体中实例化球面板对象，当单击“写入文件”按钮后，能将球面板对象写入到一个文件中。当单击“读取对象”按钮后，再将该球面板对象从文件中取出，并在窗体中显示。  2、对象序列化  序列化是一个很重要的概念，当我们使用对象流写入或读出一个对象时，其前提是这个对象必须是序列化的。这是因为我们把一个对象写入到文件后，能再把这个对象正确地读回到程序中来。人们把将一个对象转化为适合传输或磁盘存储的数据流的过程称为对象序列化。  Java提供给我们的绝大多数类对象都是序列化对象，比如组件等。用户自己定义的类需要序列化时，必须实现Serializable接口。  【例8-11】编写一个简单的程序，保存日期、地址对象到一个对象流中。  程序的第一个任务就是保证Address类序列化，因此必须使这个类实Serializable接口。  要存储对象数据，还需要建立一个ObjectOutputStream对象：  ObjectOutputStream out;  out = new ObjectOutputStream(new FileOutputStream(“sample.data”));  其次，只要简单地使用ObjectOutputStream 类中的writeObject方法，例如下列形式：  Address address = new Address(“abc”, “abc@zsm8.com”);  out.writeObject( address );  根据Date类的定义，它是已序列化的，所以我们要写入流中的所有对象都已序列化.  8.5 Java多媒体技术  在Sun公司的jdk自带的rt.jar 包文件中，有AudioStream.class、AudioPlayer.class类用于  播放声音文件，我们可以使用输入流：  FileInputStream file=new FileInputStream("声音文件.wav");  AudioStream audio=new AudioStream(file);  使用AudioPlayer类的start()进行播放：  AudioPlayer.player.start(audio);  【例8-13】应用输入流播放音频文件。  　 public Sound()  {  try  {  　 file=new FileInputStream("茉莉花.wav");  buf=new BufferedInputStream(file);  　 AudioStream audio=new AudioStream(buf);  　 AudioPlayer.player.start(audio);  　 }  　 catch (Exception e) {System.out.println("音频文件读取错误");　}  }  }  8.5.2　Java多媒体包JMF的应用  Java有一个多媒体包JMF（Java Media Framework），可以用来编写多媒体应用程序。  建立一个多媒体程序有下列几个步骤：  1、创建多媒体播放对象  try{  MediaLocator mrl=new MediaLocator(多媒体文件名);  player=Manager.createPlayer(mrl);  }  catch(MalformedURLException e){ }  catch(IOException e){ }  catch(NoPlayerException e){ }  8.5.2　Java多媒体包JMF的应用  2、向多媒体播放对象注册控制监视器  player.addControllerListener(监视器);  3、让多媒体播放对象对播放媒体进行预提取  player.prefetch();  4、启动多媒体播放对象  player.start();  5、停止并释放多媒体播放对象  player.stop();  player.deallocate();  player.close();  【例8-14】设计一个简易多媒体播放器。  重、难、疑点：  重点：输入输出流的应用  难点：输入输出流的应用 | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |

课 程 教 案

|  |  |  |
| --- | --- | --- |
| 授课题目（教学章节或主题）：  第九章GUI | 授课类型 | 机房上课 |
| 授课时间 | 第20次 |
|  |
| 教学内容：  基本内容：  重、难、疑点：  重点：  难点： | | |
| 教学手段与方法：  机房教学 | | |
| 思考题、讨论题或作业： | | |
| 参考资料（包括辅助教材、参考书、文献等）：   1. Java程序设计基础与实训，主编：程正权，安徽科学出版社 2. Java程序设计案例教程，孙一林、彭波等编，机械工业出版社 3. Java与面向对象程序设计教程，印旻，高等教育出版社 4. Java与面向对象程序设计实验指导与习题集，印旻，高等教育出版社 5. Java入门与提高，徐建华、李玉林、马军、王岩编，科学出版社 6. Java2入门与实例教程，孙燕主编，中国铁道出版社 | | |