**第2课 Java 基本语法**

|  |  |  |
| --- | --- | --- |
| **课 题** | Java 基本语法 | |
| **课 时** | 10课时（450 min）。 | |
| **教学目标** | **知识技能目标：**  1．掌握变量的声明和初始化方法，理解变量的作用域。  2．掌握一维、多维数组的声明、初始化和使用方法。  **思政育人目标：**  让学生通过学习Java 基本语法，掌握分支控制、循环控制和中断控制等流程控制语句的使用方法和技巧。 | |
| **教学重难点** | **教学重点：**常量和变量  **教学难点：**算术表达式与赋值表达式 | |
| **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）  第7节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第8节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第9节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第10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】**展示常量和变量  **任务描述**  下列程序的功能是输入圆的半径，计算圆的面积。分析其中包含的数据元素。  /\* 源程序文件名：Ex0201.java\*/  import java.io.\*;  public class Ex0201  {  public static void main(String args[])  {  final double PI = 3.14;  double fr, farea;  try  {  BufferedReader inObj = new BufferedReader(new InputStream  Reader(System.in));  System.out.println(" 请指定圆的半径：");  fr = Double.parseDouble(inObj.readLine());  inObj.close();  farea = PI\*fr\*fr;  System.out.println(" 圆的半径是：\t"+fr);  System.out.println(" 圆的面积是：\t"+farea);  }  catch(Exception e)  {  System.out.println(e);  }  }  }  **任务目标**  1. 能够正确区分程序中的变量和常量。  2. 能够为程序中的变量定义合理的数据类型和初始化，理解变量的作用域。  3. 能够遵循变量的命名规则，并能为变量取一个正确且合理的名字。  4. 具备“无规矩不成方圆”的意识，养成良好的编程习惯就如同一个公民小到遵守家规、校规，大到遵守国家法律法规，坚持“全面依法治国”，才能国泰民安。  **预备知识**  **一、常量**  在程序运行的过程中，其值不可改变的量称为常量。Java 语言中的常量有两种类型：一种称为直接常量，也叫值常量，如 23、−65、3.14；另一种称为符号常量，符号常量需要明确定义。  **（一）直接常量**  直接常量有以下几种。  1. 整型常量（byte、short、int、long）  整型常量又称整数，可以用来给整型变量赋值，整数可以使用十进制、八进制、十六进制形式来表示。  分析下列三个程序的功能，说出它们的执行结果。  程序 1：  public class Exam  {  public static void main(String args[])  {  int a,b,sum;  a = 36; /\* a,b 为十进制 \*/  b = 51;  sum = a+b;  System.out.println(sum);  }  }  程序 2：  public class Exam  {  public static void main(String args[])  {  int a,b,sum;  a = 036; /\* a,b 为十进制 \*/  b = 051;  sum = a+b;  System.out.println(sum);  }  }  程序 3：  public class Exam  {  public static void main(String args[])  {  int a,b,sum;  a = 0x56; /\* a,b 为十进制 \*/  b = 0x61;  sum = a+b;  System.out.println(sum);  }  }  程序分析：  这三个程序的功能相同，均是用来求 a 和 b 两个整数的和。区别在于三个程序采用的整数表示形式不同，分别是十进制，八进制和十六进制。  程序 1 的执行结果是十进制：87。  程序 2 的执行结果是八进制：71。  程序 3 的执行结果是十六进制：183。  2. 浮点型（实型）常量（float、double）  浮点型常量有两种表示方式，如表 2−2 所示。    数值后面加上 f 或 F 表示 float 型浮点数，占 4 个字节。  数值后面加上 d 或 D 表示 double 型浮点数，若没有则默认为 double 型，占 8 个字节。  3. 字符型常量（char）  字符型常量是无符号常量，占 2 个字节，表示的字符范围为 0～65535。Java 中的字符采用国际标准 Unicode 编码，它所包含的信息量远远多于 8 位的 ASCII 编码。计算机在处理时把这些字符当作不同的整数对待（十六进制编码 \u000～\uffff）。采用 Unicode（标准万国码）编码格式，是为了避免各国在编码过程中使用的冲突产生。例如：可以将汉字作为字符型变量的值，这为程序的国际化提供了方便。  字符型常量分为一般字符常量和转义字符常量。对于可输入字符，用单引号括起来，如‘a’‘A’‘计’等。对于不可输入字符，以反斜杠“\”开头的字符序列表示，  称为“转义字符”，用于表示 Unicode 字符集内的控制代码和某些功能定义，例如：换行、回车等。  4. 布尔型常量（boolean）  布尔型常量只有两个值：true 和 false，分别代表布尔逻辑中的“真”和“假”。必须小写。在 Java 语言中布尔常量不能转换成任何其他数据类型，true 常量不等于 1，而false 常量也不等于 0。即非 0 不代表 true，0 也不代表 false。  **（二）符号常量**  Java 中使用 final 关键字来定义符号常量，如下所示：  final double PI = 3.1415926;  符号常量可以是任何类型，定义符号常量时必须要初始化。符号常量通常要用大写字母来表示。若常量名由两个或两个以上单词组成，则单词间用下划线连接，例如：  final int MAX\_SCORE = 100;  符号常量实际上是一种特殊变量，也是占用内存空间的，不同于 C 语言中的宏。  **二、变量**  变量是指在程序运行过程中其值可以被改变的量，变量主要用于保存输入、输出和程序运行过程中的中间数据，可以理解为保存数据的容器（即存储单元）。要使用一个变量，必须先声明变量的名字、类型等特征，并对其进行初始化后才能使用它。简单变量的声明和初始化格式如下：  type name = value;  其中，type 表示变量所属的数据类型，name 指定变量的名字，value 给出变量的初始。下面是关于数据类型和变量名字（标识符）的详细内容。  **（一）数据类型**  Java 中的数据类型，如图 2−1 所示。  1. 整型数据  整型数据有 byte（字节型）、short（短整型）、int（标准型）和 long（长整型）四种。  byte：以 1 个字节存储的整型数。  short：以 2 个字节存储的整型数。  int：常使用的整数类型，以 4 个字节存储的整型数。  long：以 8 个字节存储的整型数。声明为 long 类型的整数值后需要加上“L”或“l”。  以上四种数据类型的声明用法如下：  byte b = 0x1f; // 指定变量 b 为 byte 型十六进制整数  short s = 16; // 指定变量 s 为 short 型十进制整数  int i = 123; // 指定变量 i 为 int 型十进制整数  long l = 123L; // 指定变量 l 为 long 型十进制整数  2. 浮点型（实型）数据  浮点型有 float（单精度）和 double（双精度）两种。double 类型比 float 类型具有更高的精度和更大的表示范围。  float：以 4 个字节存储的浮点型数。声明为 float 类型的数值后需要加上“F”或“f”。  double：以 8 个字节存储的浮点型数。声明为 double 类型的数值后需要加上“D”或“d”，也可以省略不加。  以上两种数据类型的声明用法如下：  double d = 3.1415926; // 指定变量 d 为 double 型  double d = 3.1415926d(D); // 指定变量 d 为 double 型  float f = 3.1415926F; // 指定变量 f 为 float 型  3. 字符型数据  字符型变量的声明用法如下：  char c = 'a'; // 指定变量 c 为字符型 , 且初始值为字符 'a' char d = '\u0061'; /\* 指定变量 d 为字符型 , 且初始值为 Unicode 编码十六进制整数 0061( 十进制 97) 的字符 'a'\*/  4. 布尔型数据  Boolean（布尔型）数据类型只有两个值：true 和 false，分别代表布尔逻辑中的“真”和“假”。在 Java 语言中布尔常量不能转换成任何其他数据类型，true 常量不等于 1，而false 常量也不等于 0。布尔类型的数据一般在 if，while，do，for 或三元运算符 ?: 环境下使用。  boolean c = true; // 指定变量 c 为布尔型 , 且初始值为布尔值 true  **（二）标识符**  标识符用于对变量、常量、类、接口和方法命名。  标识符是除关键字外的任意一串以合法字母、下画线（\_）或美元符号（$）开头的，由合法字母、数字、下画线（\_）和美元符号（$）组成的字符串。  1. Java 标识符命名规则  （1）标识符必须以字母、下画线（\_）或美元符号（$）开头。  （2）Java 中合法字母除大小写的英文字母外，还包括 Unicode 字符集里位置在 00C0 以前的字符（如中文字符、日文字符、阿拉伯字符等）。  （3）Java 的关键字（表 2−4）不能用作标识符。  （4）保留字是为以后 Java 语言扩展使用的，保留字不是关键字，也不能用作标识符。如 true、false、null 等。  （5）Java 是严格区分大小写的语言，First 和 first 分别代表的是两个不同的标识符。  （6）一般标识符用大写字母开头。同一个标识符中，中间的单词以大写字母开头，或用下划线进行分割。  （7）标识符应在一定程度上反映它所表示的变量、常量、对象接口或类的意义。  2. Java 标识符命名规范  （1）尽量能够见名知意，用单词命名，不用中文字符。  （2）通常不限制字符数，但是太长没有意义。  （3）包名：包名是全小写的单词，中间由“.”分隔开，例如：java.awt.event。  ①多人完成，版权属于发起者，包名为：indi. 发起者名 . 项目名 . 模块名……  ②独自完成，公开，版权主要属于个人，包名为：pers. 个人名 . 项目名 . 模块名……  ③独自完成，非公开，版权属于个人，包名为：priv. 个人名 . 项目名 . 模块名……  ④团队项目指由团队发起，并由该团队开发的项目，版权属于该团队所有，包名为：team. 团队名 . 项目名 . 模块名……  ⑤由公司发起，版权由项目发起的公司所有，包名为：com. 公司名 . 项目名 . 模块名……  （4）类名：通常由多个单词组成，要求每个单词的首字母都要大写。例如：HelloWorldApp。  （5）方法名：往往由多个单词组成，第一个单词通常为动词，首字母小写，其后的每个单词的首字母都要大写。例如：balanceAccount、isButtonPressed。  （6）抽象类的命名：通常在表示意义的词组前加 Abs，接口的命名规则与类名类似，只是通常在前面加 I（大写的 i），接口的实现类通常在后加 Impl（第一个是大写的 i，第4 个是小写的 L）。  （7）变量名通常是名词，全部小写。例如：length。  （8）常量的命名全部大写，单词与单词间用下画线 \_ 分隔。例如：YEAR，WEEK\_OF\_MONTH；如果是对象类型的常量，则是大小写混合，由大写字母把单词隔开。  **（三）变量的作用域**  变量经过声明和初始化后就可以使用了，但并不是一个变量声明和初始化后就可以随便在哪里都能使用，这里涉及一个作用域的问题。变量的作用域就是该变量能被访问的区间，在程序源代码中通常以程序段来划分。  在本任务中，定义了二个变量：fr 和 farea，由于这二个变量都是在 main 方法中定义的，所以它们的作用域就是整个 main 方法。也就是说，在 main 方法中的任何地方，都可以访问到这二个变量的值。  再看下面的代码：  public static void main(String[] args)  {  int sum = 0;  for(int i = 1; i<=100; i++)  {  sum+=i;  }  System.out.println("1 到 100 的累加和是："+sum);  }  这里定义了两个变量：sum 和 i。其中，sum 是在 main 方法中定义的，所以可以在main 方法中的任意地方使用；而 i 是在 for 循环中定义的，就只能在 for 循环中使用，当for 循环结束后，变量 i 所占用的内存空间将被释放，这个变量也就不复存在了，所以我们不能在最后的输出语句之后再增加这样一条语句：  System.out.println(" 一共计算了 "+i+" 次。");  否则会出现“找不到符号”的错误提示，无法完成编译。  关于变量的作用域在这里先简单地做一个介绍，当我们学习到类和对象的时候，将更深入地介绍类和对象中的变量的作用域及其生命周期。  **任务实施**  我们来关注这些语句：  final double PI = 3.14;  double fr, farea;  第一条语句是定义常量。定义了一个常量：PI，它是 double 类型，即双精度实型常量，PI 的值是 3.14。  第二条语句是对变量的声明和初始化。所谓声明，就是在使用一个变量前，为这个变量取一个名字，并规定它能够存放的数据类型，而初始化则是为变量设定一个初始值。在这里，声明了二个变量：fr 和 farea，它们都是 double 类型，即双精度实型变量。  fr 和 farea 还没有初始化，此时它们没有具体的值。  fr = Double.parseDouble(inObj.readLine());  在这一行中，变量 fr 接受了从键盘输入的半径值，它的初始化工作到现在才完成。  farea = PI\*fr\*fr;  这里计算圆的面积，并把结果存放在变量 farea 中，完成对 farea 的初始化。  System.out.println(" 圆的半径是：\t"+fr);  System.out.println(" 圆的面积是：\t"+farea);  这里分别输出圆的半径和面积。  **【学生】**思考、讨论。 | **展示常量和变量，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  下列哪些是合法的标识符：  MyClass、24hour、X、Test3、Open-on、\_Value、SunList、boolean、True、$123。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示算术表达式与赋值表达式  **任务描述**  键盘输入一个 3 位整数，求它的百位、十位和个位，并输出。  **任务目标**  1. 能够正确书写并使用 Java 的算术运算符与算术表达式。  2. 能够正确书写 Java 的赋值运算符与赋值表达式。  3. 能够把小事做细，树立“细节决定成败“的意识，脚踏实地为“程序”服务。  **预备知识**  **一、算术运算符与算术表达式**  **（一）算术运算符**  算术运算符用于完成算术运算。Java 提供了两大类共八个算术运算符。  1. 双目运算符  双目运算符是有两个操作数的运算符，Java 中有五个双目算术运算符，如表 2−5 所示。    其中加法（+）运算符除能对数值进行加法运算操作外，还可以用来实现字符串的连接，比如下面的语句：  System.out.println("abc"+"de");  就将输出字符串“abcde”。  另外，与 C、C++ 不同，取余（%）运算的操作数不但可以是整数也可以是浮点数。例如：整数 % 整数，结果是整数，如 12%5=2；浮点数 % 整数，结果是浮点数，如12.3%6 = 0.3；整数 % 浮点数，结果是浮点数，如 19%6.7 = 5.6；浮点数 % 浮点数，结果是浮点数，如 18.6%6.7 = 5.200000000000001。  2. 单目运算符  单目运算符是只有一个操作数的运算符，Java 中有四个单目算术运算符。需要注意的是自增（+ +）和自减（− −）运算符在做前缀和后缀使用时是有区别的。分析下列程序的输出结果，注意其中前置运算和后置运算的区别。  public class Exam  {  public static void main(String args[])  {  int i,x,y;  i = 5;  x = i++; /\* 后置运算 , 先把 i 的值赋给 x, 然后 i 的值加 1\*/System.out.println("i = %d,x = %d\n",i,x);  i = 5;  y = ++i; /\* 前置运算 , 先使 i 的值加 1, 然后将 i 的值赋给 y\*/System.out.println("i = %d,y = %d\n",i,y);  }  }  程序运行结果如下所示：  i = 6,x = 5  i = 6,y = 6  **（二）算术表达式**  用算术运算符或圆括号将运算对象（也称操作数）连接起来的、符合 C 语法规则的式子，称为 C 的算术表达式。运算对象包括常量、变量、函数等。算术表达式在使用时要注意书写形式，如表 2−7 所示。    **二、赋值运算符与赋值表达式**  **（一）简单赋值运算符**  赋值运算符“ =”用于将运算符右边表达式的值赋给左边的变量，如“x=5”是把常量 5 赋给变量 x。由赋值运算符将一个变量和一个表达式连接起来的式子称为“赋值表达式”。它的基本使用格式如下：  变量 = 表达式 ;  例如：“x=3+7”是一个赋值表达式。对赋值表达式的求解过程是：先计算表达式“3+7”的值（10），然后再将 10 赋给变量 x。  在赋值表达式后加上分号就构成了赋值语句。例如：  a = 3+7;  赋值语句允许给多个变量连续赋值，但在定义变量时，不允许连续给多个变量赋初值。  例如：a=b=c=d=5；是合法的。而 int a = b = c = d = 5；是不合法的，应该写成：int a=5，b=5，c=5，d=5。  **（二）复合赋值运算符**  除基本赋值运算符外，Java 还提供了一系列复合赋值运算符（快捷赋值运算符），用于在一个运算符里同时完成赋值操作和另一个算术、移位或按位操作。例如：  i = i + j;  可以使用快捷赋值运算符“ + =”：  i+=j;  赋值运算符和复合赋值运算符的优先级高于逗号运算符，结合方向均为“从右到左”。  **（三）赋值表达式**  （1）赋值运算符的左边必须是变量名，被赋值变量的值就是赋值表达式的值。  （2）当赋值表达式中左边的变量与右边的表达式的数据类型不同时，需要进行类型转换。转换的规则是：右边表达式的值被转换成左边变量的数据类型，然后再赋值给变量。  （3）赋值表达式也可以包含复合的赋值运算符。例如：  a+=a− = a\*a  如果 a 的初值为 8，此赋值表达式的求解步骤如下：  ①先进行“a− = a\*a”的运算，它相当于 a=a−a\*a= 8−64 = −56。  ②再进行“a+= −56”的运算，它相当于 a=a+（−56）= −56−56 = −112。  **任务实施**  /\* 源程序文件名：Ex0202.java\*/  import java.io.\*;  public class Ex0202  {  public static void main(String args[])  {  int number,ge,shi,bai;  try  {  BufferedReader inObj = new BufferedReader(new InputStreamReader(System.  in));  System.out.println(" 请输入一个 3 位整数：");  number = Integer.parseInt(inObj.readLine());  inObj.close();  bai = number/100; /\* 分离百位 \*/  shi = number/10%10; /\* 分离十位 \*/  ge = number%10; /\* 分离个位 \*/  System.out.println(" 百位数 :"+bai); /\* 输出百位 \*/  System.out.println(" 十位数 :"+shi); /\* 输出十位 \*/  System.out.println(" 个位数 :"+ge); /\* 输出个位 \*/  }  catch(Exception e)  {  System.out.println(e);  }  }  }  **【学生】**思考、讨论。 | **通过教师讲解，了解算术表达式与赋值表达式的基本理论知识。** |
| **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算术表达式与赋值表达式，让学生能够把小事做细，树立“细节决定成败“的意识，脚踏实地为“程序”服务。** | 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  下面两个程序段执行后，变量 x 的值是多少？  程序段 1：  int x,a = 7;  x = (a++)+3;  System.out.println("%d",x) ; /\* 变量 x 的值是 \*/  程序段 2：  int x,a = 7;  x = (++a)+3;  System.out.println("%d",x) ; /\* 变量 x 的值是 \*/ | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示条件判断  **任务描述**  用关系表达式或逻辑表达式描述下列条件：  （1）判断 x 是否为偶数；  （2）判断字符变量 ch 是否为英文字母；  （3）判断 year 是否闰年。  **任务目标**  1. 能够正确书写 Java 的关系表达式和逻辑表达式，为选择结构程序设计打好基础。  2. 会计算关系表达式和逻辑表达式的值。  3. 能够变“抽象”为“具体”，具有举一反三、自己获取知识的能力。  **预备知识**  **一、关系运算符和关系表达式**  **（一）关系运算符**  关系运算符用来比较两个操作数的值，其运算结果是布尔型值 true 或 false。关系运算符都是双目运算符，Java 语言一共提供了 6 个关系运算符。  Java 中任何类型的数据（包括基本类型和构造类型）都可以使用 = = 和 ! = 来比较是否相等。关系运算符常与条件运算符一起使用，作为流程控制语句的判断条件。  使用关系运算符时，应注意其优先级。  （1）“<”“< =”“>”“> =”这四个运算符的优先级相同，“ = =”和“! =”的优先级相同。前 4 个运算符的优先级高于后两个运算符的优先级。  （2）关系运算符的优先级低于算术运算符而高于赋值运算符。它们的结合性是“自左至右”。例如：  a! = b + c 等价于 a!=（b+c）；a= b>c 等价于 a =（b>c）。  **（二）关系表达式**  关系表达式就是用关系运算符将两个或两个以上运算对象连接起来的式子。其中，运算对象可以是常量、变量或表达式。关系表达式的运算结果有两种：“真”或“假”。在 Java 语言中用 true 表示“真”，用 false 表示“假”。例如：3\*5>6\*2，表达式的执行结果为 true。  **二、逻辑运算符和逻辑表达式**  **（一）逻辑运算符**  逻辑运算符用来进行逻辑运算，操作数类型是布尔型数据，运算结果也是布尔型。使用逻辑运算符时，应注意它的优先级和结合性。  （1）逻辑运算符从高到低的优先次序为：!（逻辑非）→ &&（逻辑与）→ ||（逻辑或）。  （2）逻辑非“！”的结合方向是“自右至左”，逻辑与“&&”和逻辑或“||”的结合方向是“自左至右”。  （3）常见运算符的优先次序如图 2−2 所示。    **（二）逻辑表达式**  用逻辑运算符将关系表达式或逻辑量连接起来的式子称为逻辑表达式。逻辑表达式的值也有两种，即“真”和“假”。有一点需要指出，在逻辑表达式的求值过程中，存在“短路”行为。  （1）在求解 a&&b&&c 的值时，只有 a 的值是真时，才会计算 b 的值；只有 a 和 b均为真时，才会计算 c 的值。如果 a 为假，则整个表达式的值已经确定为假，就不会继续计算 b 和 c 的值。  （2）在求解 a||b||c 的值时，只有 a 为假时，才会计算 b 的值；只有 a 和 b 均为假时，才会计算 c 的值。如果 a 是真，则整个表达式的值已经确定为真，就不会计算 b 和 c的值。  **三、位运算符**  **（一）按位运算符**  按位运算符包括有四种，其使用方法和功能如表 2−11 所示。    逻辑与、或和按位与、或的比较  相同点：都能参与逻辑运算，按位与或完全可以替代逻辑与或。  区别：按位与或可以参与位运算，逻辑与或只能参与逻辑运算；逻辑与或可以“短路”，按位与或不能“短路”。  逻辑与的短路操作：先计算左边的操作数，如果左操作数为 false，此时已经可以得到整个表达式的值为 false，则不会计算右操作数。例如：a>b&&c<d 一旦 a>b 为假，c<d 不再计算。  逻辑或的短路操作：先计算左边的操作数，如果左操作数为 true，此时已经可以得到整个表达式的值为 true，则不会计算右操作数。例如：a>b||c<d 一旦 a>b 为真，c<d 不再计算。  再如下面的例子：  int a = 5, b = 0;  boolean d = (a<0)&(a/b>0);  这里使用的是按位与，所以即使通过左操作数 a<0 已经能够判断 d 应该为 false，也要计算右操作数 a/b>0，从而会抛出一个算术异常。如果将上例第二个语句改为：  boolean d = (a<0)&&(a/b>0);  此时不会计算右操作数，当 a>0 时程序运行正常，不会抛出异常，但当 a<0 时还需计算右操作数，从而抛出异常。简洁运算可以在一定程度上减少计算次数，提高程序运行效率，但对程序中的错误检查就不那么全面，所以在使用上需要大家注意。  **（二）移位运算符**  移位运算符包括三种，其使用方法和功能如表 2−12 所示。    移位运算适用数据类型：byte、short、char、int、long，对低于 int 型的操作数将先自动转换为 int 型再移位。左移运算符是将操作数对应的二进制数左移若干位。如17<<2，即（0000000000010001）2 左移两位，相当于使 17 的各位二进制位左移 2 位，右补 2 个 0。在不产生溢出的情况下，左移一位相当于操作数乘 2，左移二位相当于操作数乘 4，17<<2 结果为（0000000001000100）2，即 68，相当于 17×22= 68。而且用左移来实现乘法比乘法运算速度快。右移运算符是将操作数对应的二进制数右移若干位，如88>>2，即（0000000001011000）2 右移两位，相当于使 88 的各位二进制数位右移 2 位，移到右端的低位被舍弃，高位则移入原来高位的值。右移一位相当于操作数除 2 取商，右移二位相当于操作数除 4 取商，88>>2 结果为（0000000000010110）2，即 22，相当于88/22= 22，而且用右移实现除法比除法运算速度快。而无符号右移则专门针对无符号数进行操作，忽略了二进制数左端的符号位的作用，这一点大家可以结合计算机中数值的补码表示方式一起理解。  **四、其他运算符**  Java 中除了前述的几大类运算符之外，还有一些运算符，如表 2−13 所示。    （1）“? :”运算符是唯一的一个三元运算符，它的形式为：  a?b:c  程序运行时先判断 a，如果 a 为 true，则返回 b 的值，否则返回 c 的值。例如：  int a, b = 5, c = 7;  a = (b>c)?b:c; //a 值为 7  可见，利用第二条语句可以自动判断 b 和 c 的大小，并返回其中较大的值。  （2）“[]”运算符用来定义数组，创建数组，或访问数组元素。例如：  int[] aValue = new int[12];  aValue[3] = 12;  关于数组的内容，将在本单元任务 6 中介绍。  （3）“.”运算符用来访问对象的成员函数或成员变量，这部分内容将在下一单元中详细说明。  （4）“（参数 1，参数 2，……）”运算符用于向方法中传递参数，这部分内容也将在下一单元说明。  （5）“（变量类型）”，将在本单元任务 4 中介绍。  （6）“instanceof”运算符的用法为：  a instanceof b  其中，a 是对象名，b 是类名。如果对象 a 是 b 类或其直接或间接子类的实例，则该表达式返回 true；如果对象 a 是 b 类或其直接或间接父类的实例，则表达式返回 false。对象与继承的概念，在下一单元中说明。  **五、运算符的优先级**  对于包含多个运算符的表达式，计算时必须遵循一定的先后次序，就如同在学习四则运算的时候，老师要我们记得“先乘除，后加减”一样。对于相同优先级的运算符，Java 采用从左到右的次序，但赋值运算是从右到左的，如：  a = 14\*3/2\*5/7;  由于乘除法优先级相同，程序将从左到右计算：14\*3 = 42，42/2= 21，21\*5 =105，105/7= 15，最后执行优先级最低的赋值运算，将 15 赋给变量 a。再如：  a = b = c = 10;  首先将 10 赋给变量 c，变量 c 的值是 10，同时表达式“c = 10”的结果也是 10；然后把表达式的结果 10 再赋给变量 b，变量 b 的值是 10，同时表达式“b = c = 10”的结果也是 10，最后把表达式的结果 10 赋给变量 a，变量 a 的值是 10，于是变量 a、b、c 的值都是 10。  Java 的优先级规则有点复杂，但实际上程序员不需要记住这十五级运算符的先后次序，除了四则运算和逻辑运算的优先级外，通常的优先级问题都很好解决，对于确实复杂的运算还可以使用括号“( )”来辅助，同时也可以提高程序的可读性。  **任务实施**  下列条件对应的关系表达式或逻辑表达式为：  （1）判断 x 是否为偶数。  x%2 = = 0 或者 !（x%2）  （2）判断字符变量 ch 是否为英文字母。  ch>=‘a’&&ch<=‘z’||ch> =‘A’&&ch<=‘Z’  （3）判断 year 是否闰年。  year%4 = =0&&year%100! = 0||year%400 = = 0  **【学生】**思考、讨论。 | **教师通过条件判断展示，让学生了解条件判断的基本理论知识。** |
| **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条件判断,能够变“抽象”为“具体”，具有举一反三、自己获取知识的能力。** | 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**  **简述运算符的优先级。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示数据类型的转换  **任务描述**  若有定义：  int b = 15;  float a = 5.5f,c = 2.9f;  则下面表达式的值是多少？  c+(int)(b/4\*(int)(a+c)/2)%4  **任务目标**  1. 能够正确描述 Java 的编译系统对不同类型数据进行运算时的转换规则。  2. 能够根据编程需要进行强制类型转换。  3. 能够领会“变则通，通则存，存则强”的精髓。拒绝做一个墨守成规的人。牢记创新驱动发展，为实现中国式的现代化贡献自己的力量。  **预备知识**  赋值运算的基本要求是运算符左右两边的数据类型必须一致，如果遇到赋值号两边的数据类型不一致的情况，就需要把右边的数据类型转换成左边的数据类型。通常情况下，如果类型不一致，需要程序员在程序中进行强制类型转换，在某些特殊情况下，Java 也可以自动进行类型转换。  **一、强制类型转换**  强制类型转换也叫显式转换，它的一般格式是：  ( 数据类型 ) 变量名或表达式  由于不同数据类型所占用的存储空间不同，转换后可能丢失信息，所以当较长的数据类型转换为较短的数据类型时必须使用强制类型转换，程序员需要自己确保数据在类型转换后的有效性。请看下列语句：  int i;  long l = 0xFFFFFFFFFFL;  double d = 12.34;  i = (int)l; // 将 l 转换成 int 类型 , 有可能丢失信息  i = (int)d; // 将 d 转换成 int 类型 , 有可能丢失信息  l = (long)i; // 将 i 转换成 long 类型 , 不会丢失信息  **二、自动类型转换**  当赋值运算符左右两边数据类型不一致，且右边的数据类型存储长度小于等于左边的数据类型，即转换后不会丢失信息的情况下，Java 可以自动完成数据类型的转换。请看下列语句：  short s = 10;  int i = 100;  float f;  long l;  double d;  f = i; //i 自动转换成 float 类型  l = i; //i 自动转换为 long 类型  d = i; //i 自动转换为 double 类型  i = s; //s 自动转换为 int 类型  s = i; //i 不能自动转换为 short 类型  简单类型除了 boolean 类型以外，其他的数据类型可以自动转换成其他的数据类型，要遵循的原则。  **任务实施**  解题步骤：  （1）先计算 b/4 的值，为 3；得到如下式子：  c+（int）（3\*（int）（a+c）/2）%4  （2）计算（int）（a+c），值为 8，得到如下式子：  c+（int）（3\*8/2）%4  （3）计算 3\*8，值为 24，得到如下式子：  c+（int）（24/2）%4  （4）计算（int）（24/2），值为 12，得到如下式子：  c + 12%4  （5）计算 12%4，然后与 c 相加，最终表达式的值为 2.9。  **【学生】**思考、讨论。 | **通过教师讲解，了解数据类型的转换的基本理论知识。** |
| **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数据类型的转换，让学生知道能够领会“变则通，通则存，存则强”的精髓。** | 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  int x = 3,y = 2;  float a = 2.5,b = 3.5;  则表达式 (x + x\*y)%4 + (int)a/(int)b 的值为 。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示流程控制语句（一）  **任务描述**  编写如下程序。  （1）编写程序，统计全班学生单科成绩的合格率。  （2）编写多分支程序。通过字符变量 grade 的取值来判断输出内容，即通过考试成绩的等级输出相应的分数段。如果 grade 是‘a’，输出 grade is 85 ～ 100；如果 grade 是‘b’，输出 grade is 70 ～ 84；如果 grade 是‘c’，输出 grade is 60 ～ 69；如果 grade 是‘d’，输出 grade is <60；如果 grade 是除 a、b、c、d 外的等级，那么输出“Input Error!”提示。  （3）编写程序，计算 1×9、2×9、……、9×9 的结果并输出。  （4）编写程序，求 100 ～ 200 间的所有素数。  **任务目标**  1. 能正确掌握 Java 的分支语句的写法，根据编程需要编写选择结构程序。  2. 能正确掌握 Java 的循环语句的写法，根据编程需要编写循环结构程序。  3. 能正确掌握 break 语句和 continue 语句的写法，根据编程需要应用。  4. 做一个能灵活运用知识，懂得按“规矩”办事的人，不忘初心牢记使命。  **预备知识**  **一、条件分支**  **（一）if 语句**  if 语句是最基本的分支控制语句，它可以让程序根据某些条件选择性地执行语句。  if 语句的形式如下：  if( 关系表达式 )  {  语句  }  它的含义是：如果关系表达式的值为 true，则执行后面大括号里的语句，否则就不执行大括号里的语句。例如，假设程序要判断字符变量 iChar 中保存的是不是字符“q”，可以使用这样的语句：  if(iChar=='q'||iChar=='Q')  {  System.out.println(" 是 Q。");  }  当字符 iChar 的值是字母 q（不管是大写还是小写）时，关系表达式的值为 true，因此执行大括号里的语句，即输出“是 Q。”这样一条信息。如果 iChar 的值不是字母 q，则不会显示这条信息。  需要注意的是，在上面的程序中，大括号里只有一条语句，因此可以省略大括号，将程序简化为：  if(iChar=='q'||iChar=='Q')  System.out.println(" 是 Q。");  但如果有两条或以上的语句，则必须用大括号括起来。  **（二）if...else 语句**  有时，程序需要控制当条件满足和不满足时分别执行不同的语句，这时可以使用if...else 语句，通过判断一个条件，可以决定程序到底执行哪些语句。  if...else 语句的形式如下：  if( 关系表达式 )  {  语句 1  }  else  {  语句 2  }  它的含义是：如果关系表达式的值为 true，则执行后面大括号里的语句 1，否则执行大括号里的语句 2。语句 1 和语句 2 总会有一个被执行。语句 1 和语句 2 都可以是多行，如果只有一行，则相应的括号可以省略。  例如，假设程序要判断字符变量 iChar 中保存的是不是字符“q”，可以使用这样的语句：  if(iChar=='q'||iChar=='Q')  {  System.out.println(" 是 Q。");  }  else  {  System.out.println(" 不是 Q。");  }  当字符 iChar 的值是字母 q（不管是大写还是小写）时，关系表达式的值为 true，则执行语句 1，即输出“是 Q。”。如果 iChar 的值不是字母 q，则执行语句 2，即输出“不是 Q。”  **（三）if...else 嵌套**  如果我们的程序需要有两个以上的分支，那么单纯的 if...else 结构就不能满足需要了，这时可以使用嵌套的 if...else 结构。例如：判断变量 a 的值是正数、负数还是零，那么相应的代码就应该是：  if(a>0)  {  System.out.println("a 的值大于零。");  }  else if(a<0)  {  System.out.println("a 的值小于零。");  }  else  {  System.out.println("a 的值等于零。");  }  这里出现了 else if 语句，它实际上是在 else 语句后面接了另一个完整的 if...else 结构。由于一个 if...else 结构算是一条语句，所以 else 和 if 之间没有大括号，我们也完全可以在 else 和 if 之间加上大括号，这样更容易理解：  if(a>0)  {  System.out.println("a 的值大于零。");  }  else  {  if(a<0)  {  System.out.println("a 的值小于零。");  }  else  {  System.out.println("a 的值等于零。");  }  }  这种 if...else 的嵌套结构让程序有了三种分支选择，如果需要有更多的选择，也可以使用更复杂的嵌套结构，形成 if...else if...else if...else 结构。当然，这会使程序的可读性降低。另外，在某些情况下，可以使用 switch 语句来处理多分支选择的问题。  **（四）switch 语句**  switch 语句用于处理多分支选择，它的一般形式是：  switch(intVar)  {  case Var1 : 语句 1; break;  case Var2 : 语句 2; break;  ......  case Varn : 语句 n; break;  default: 语句 n+1;  }  intVar 的最终值的类型必须是整型或能自动转换成整型的类型，如字节型、短整型、字符型、整型、枚举（JDK1.5 引入）、字符串（JDK1.7 引入）。switch 语句将这个值与每个 case 后的整型值比较，如果相等，就执行这个 case 后面的语句，如果不能和任何一个 case 后的值对比上，就执行 default 后面的语句。其中的 default 子句是可选的，如果没有 default 子句，而且 intVar 的值也不符合任意一个 case 语句，则整个 switch 结构不  做任何操作。  if 语句（包括 if...else 和 if...else 嵌套）和 switch 语句是 Java 语言提供的两种分支选择语句，它们有各自的特点和适用范围。if 语句的适用性更强，所有的分支选择都可以使用 if 语句（包括 if...else 和 if...else 嵌套）来完成，但多重的 if...else 嵌套将严重降低程序的可读性；switch 语句的可读性比多重 if...else 嵌套要好得多，但它又只能对整型量进行判断。所以在编写程序的时候，应当根据具体情况来选择使用何种分支选择语句。  **【学生】**思考、讨论。 | **通过教师讲解，了解流程控制语句（一）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流程控制语句（一），让学生做一个能灵活运用知识，懂得按“规矩”办事的人，不忘初心牢记使命。** | 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if...else 嵌套 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示流程控制语句（二）  **二、循环**  程序设计中有时需要将一段代码反复执行，这时就需要用到循环控制语句。使用循环控制语句可以提高代码的使用率，减轻程序员负担，增加程序运行效率，让我们用很简洁的语句完成重复的、复杂的操作。  **（一）while 循环**  while 语句实现“当”型循环，即先判断循环条件，如果条件满足，则执行循环体，否则就结束循环。while 语句的一般格式如下：  while( 条件表达式 )  {  循环体  }  while 语句通常用于循环次数未知或不方便计算，而使用某种条件来判断循环是否结束的情况。由于 while 语句是在执行循环体之前先判断循环条件，所以循环体有可能不会被执行。  例如：  int i = 1;  while(i<=9)  {  System.out.println(i+" ×9 = "+i\*9);  i++;  }  具体执行过程请大家自行分析。  **（二）do...while 循环**  do...while 语句的判断过程与 while 语句正好相反，它是先执行循环体，再进行判断以决定是否进行下一次循环。do...while 语句的一般格式如下：  do  {  循环体  }  while( 条件表达式 );  与 while 语句的循环体有可能不被执行不同，do...while 语句的循环体至少会被执行一次。在大多数情况下，for 语句、while 语句和 do...while 语句都可以相互替换，实现同样的功能，但由于 while 语句和 do...while 语句存在着细微差别，所以不是所有使用while 语句的情况都可以用 do...while 语句替换的，这一点一定要注意。  同样，我们可以用 do...while 语句改写上面的代码：  int i = 1;  do  {  System.out.println(i+" ×9 = "+i\*9);  i++;  }  while(i<=9);  也可以完成同样的操作。  **（三）for 循环**  for 语句是最常用的一种循环控制语句，它通常用于循环次数已知的情况。for 语句的一般格式如下：  for( 初始化表达式 ; 循环条件表达式 ; 迭代表达式 )  {  循环体  }  for 语句在运行时，首先执行初始化表达式，对循环控制变量和相关的其他变量进行定义和初始化。接下来判断循环条件，如果条件满足，就执行循环体内的语句，否则就结束循环。循环体执行完后，计算迭代表达式，这里通常是对循环控制变量进行增减，以达到控制循环次数的目的。然后再次判断循环条件，一直到循环条件表达式的值为 false，结束循环。例如：  for(int i = 1; i<=9; i++)  {  System.out.println(i+"×9 = "+i\*9);  }  在这里，实际的执行过程是这样的：  （1）定义并初始化循环控制变量 i，此时 i=1；  （2）判断循环条件，此时为 true；  （3）执行循环体，输出“1×9=9”；  （4）执行迭代表达式，此时 i=2；  （5）判断循环条件，此时为 true；  （6）执行循环体，输出“2×9 = 18”；  ……  （27）执行循环体，输出“9×9 = 81”；  （28）执行迭代表达式，此时 i = 10；  （29）判断循环条件，此时为 false，跳出循环，程序结束。  可以看到，在这样一个 for 语句里，其实进行了 1 次定义、29 次判断、输出和计算操作，这也从一个方面体现了使用循环控制语句的优点。关于 for 语句，还有两点需要注意：  （1）初始化表达式和迭代表达式中可以使用逗号分隔多个语句，例如：  for(i = 0, j = 10; i< 10; i++, j--)  {  ...  }  （2）初始化表达式、循环条件表达式、迭代表达式和循环体都可以为空（但分号不能省略），当四者都为空时，相当于一个无限循环，当然在编程时应该尽量避免这种情况。  for 循环的具体应用，请看下面的例子：  求 Fibonacci 数列的前 20 个数。该数列的生成方法为：f1 = 1，f2 = 1，fn = fn−1 + fn−2（n> = 3），即从第 3 个数开始，每个数等于前 2 个数之和。  public class Fib  {  public static void main(String args[])  {  int f1 = 1,f2 = 1;  System.out.print(f1+" "+f2+" ");  for(int i = 1;i<=9;i++)  {  f1 = f1+f2;  f2 = f1+f2;  System.out.print(f1+" "+f2+" ");  }  }  }  f1 和 f2 是数列的前两个数（已知），这里面循环一共迭代 9 次，每次求两个数，反复利用变量 f1 和 f2。  **三、中断控制语句**  在一些循环结构中，可能遇到某种情况，需要中止循环，但 Java 没有提供 goto 语句，不能使用直接跳转语句跳出循环，但是 Java 针对这种情况，提供了 break、continue和 return 语句，用于不同情况下的中断控制。  **（一）break 语句**  在 switch 语句中，break 用来终止 switch 语句的执行，程序流程转到 switch 语句块后的第一条语句。同样在循环语句 for、while 和 do...while 中，break 将立即终止正在执行的循环，程序流程转到循环语句后的第一条语句。如果 break 后面有标号，会终止标号的循环体。  下面是无标号的例子：  int i,j;  for(i = 0; i < 4; i++)  {  for(j = 0; j < 4; j++)  {  if(j==2)  break;  System.out.println(i+":"+j);  }  }  这段代码的运行结果是：  0:0  0:1  1:0  1:1  2:0  2:1  3:0  3:1  当满足 j 等于 2 的时候，执行 break，内层 for 循环终止，继续下一次外层 for 循环。下面是有标号的例子 :  int i,j;  here:for(i = 0; i < 4; i++)  {  for(j = 0; j < 4; j++)  {  if(j==2)  break here;  System.out.println(i+":"+j);  }  }  这段代码的运行结果是：  0:0  0:1  当满足 j 等于 2 的时候，执行 break here，外层 for 循环终止。  **（二）continue 语句**  continue 语句只能用在循环语句中，它有两种使用格式。  一种是不带标号的 continue 语句，它的作用是终止当前这次循环，继续下一次循环。另一种是带标号的 continue 语句，其格式是：continue 标号名。它的作用是终止标号名对应循环的当前这次循环，继续标号名对应循环的下一次循环。  下面是无标号的例子：  int i,j;  for(i = 0; i < 4; i++)  {  for(j = 0; j < 4; j++)  {  if(j==2)  continue;  System.out.println(i+":"+j);  }  }  这段代码的运行结果是：  0:0  0:1  0:3  1:0  1:1  1:3  2:0  2:1  2:3  3:0  3:1  3:3  当满足 j 等于 2 的时候，执行 continue，继续下一次内层 for 循环。  下面是有标号的例子：  int i,j;  here:for(i = 0; i < 4; i++)  {  for( j = 0; j < 4; j++)  {  if(j==2)  continue here;  System.out.println(i+":"+j);  }  }  这段代码的运行结果是：  0:0  0:1  1:0  1:1  2:0  2:1  3:0  3:1  **（三）return 语句**  return 语句从当前函数中返回到调用该函数的语句处，并继续执行后面的语句（有关函数的内容将在单元三中介绍）。return 语句有两种格式：  return 表达式 ; 和 return;  第一种格式是返回一个值给调用该函数的语句。返回值的类型必须与函数声明中的返回值类型一致。  第二种格式是直接结束函数的运行，不返回任何值，相应的该函数的返回值类型应该为 void。  return 语句通常用在一个函数体的最后，退出该函数并返回一个值。单独的 return 语句用在一个函数体的中间时，会产生编译错误，因为此时会有一些语句不会被执行。  可以通过把 return 语句嵌入到某些语句（如 if...else）中，来让程序在没有执行完函数中的所有语句时退出。如下面的例子：  int method(int num)  {  if(num>0)  return num;  else  return –num;  }  函数 method 根据 num 的值是否大于 0，决定是返回 num 还是返回 −num，从而返回了 num 的绝对值。  **任务实施**  (1)/\* 源程序文件名：Ex0203.java\*/  public class Ex0205  {  public static void main(String args[])  {  int a = -23;  if(a>=0)  {  System.out.println("a 的值大于等于零。");  }  else  {  System.out.println("a 的值小于零。");  }  }  }  这是一个使用分支控制语句 if( ){}…else{} 的简单程序，在变量 a 有一个确定值的情况下，使用 if( ) 语句判断分支条件，也就是（a> = 0）是否成立。如果成立，程序就执行 if( ) 语句后面的大括号里的语句；如果不成立，程序就执行 else 后面的大括号里的语句。这样，根据某一个判断条件，可以让程序分别执行不同的语句，实现不同的操作。  (2)/\* 源程序文件名：Ex0204.java\*/  public class Ex0204  {  public static void main(String args[])  {  char grade = 'c';  switch (grade)  {  case a : System.out.println(grade + "is 85 ～ 100"); break;  case b : System.out.println(grade + "is 70 ～ 84"); break;  case c : System.out.println(grade + "is 60 ～ 69"); break;  case d : System.out.println(grade + "is <60"); break;  default : System.out.println("Input Error !");  }  }  }  这里通过字符变量 grade 的取值来判断输出内容，即通过考试成绩的等级输出相应的分数段。如果 grade 是除 a、b、c、d 外的等级，那么输出错误提示。大家可以看到，每个 case 语句后面都有一个 break 语句，它的作用是终止整个 switch 语句的执行。  (3)/\* 源程序文件名：Ex0205.java\*/  public class Ex0205  {  public static void main(String args[])  {  for(int i = 1; i<=9; i++)  {  System.out.println(i+"×9 = "+i\*9);  }  }  }  按照以前的程序设计思路，要进行 9 次计算和输出，至少需要 9 条语句，分别计算1×9、2×9、……、9×9 的结果并输出。但是这些语句里面，除了乘数有变化，而且是有规律的变化之外，其他的计算和输出都是一样的。所以使用了循环控制语句 for，通过循环控制变量 i，来控制计算时的乘数以及计算的次数。这样，原本需要使用 9 条语句完成的程序，在这里被简化为了一个 for 语句段。  (4)/\* 源程序文件名：Ex0206.java\*/  import java.lang.Math;  public class Ex0206  {  public static void main(String args[])  {  int n = 0;  outer: for(int i = 101; i < 200; i++)  {  int k = (int)Math.sqrt(i);  for(int j = 2; j<=k; j++)  {  if(i%j==0)  continue outer;  }  System.out.print(" "+i);  n++;  if(n<10)  continue;  System.out.println();  n = 0;  }  }  }  在这个任务里，使用嵌套的 for 语句，其中外层 for 语句用于控制变量 i，也就是用来判断是否是素数的那个数的递增，内层 for 语句用于控制对 i 是否是素数的判断。注意内层循环里的这条语句：  if(i%j==0)  continue outer;  这里判断 i 是否能被 j 整除，如果可以，说明 i 不是素数，就没有必要再继续后面的运算了，所以使用 continue 语句结束本次循环，继续计算下一个 i。这里不能使用 break或 breakouter，break 是结束内层循环，但会继续执行内层循环后面的语句，将所有的 i都输出，而 breakouter 则是结束 outer 所指的外层循环，这样整个程序都结束了。  后面还有一句使用 continue 的语句：  if(n<10)  continue;  这是属于外层循环的 continue，意思是如果 n 小于 10，即不满足换行条件，就继续执行下次循环；如果 n 大于或等于 10，说明应该换行了，就输出一个换行符，并将 n 置零，重新计数。  **【学生】**思考、讨论。 | **通过教师讲解，了解流程控制语句（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流程控制语句（二），让学生知道使用了循环控制语句 for，通过循环控制变量 i，来控制计算时的乘数以及计算的次数。** | 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  简述中断控制语句。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示数组  **任务描述**  编写如下程序：  （1）统计全班学生单科成绩的合格率；  （2）使用二维数组统计全班学生多科成绩的合格率。  **任务目标**  1. 能够掌握一维数组的定义和使用，根据编程需要灵活使用一维数组。  2. 能够掌握多维数组的定义和使用，主要是二维数组；根据编程需要灵活使用二维数组。  3. 做一个有“组织”的数据，方便组织统一管理。  **预备知识**  在前面的学习中，我们知道了一个变量可以存储一个数据，但是很多程序中的数据量很大，如果每一个数据都用一个变量来存储的话，在变量的命名和使用上会有很大的麻烦。当这些数据类型相同的时候，可以使用数组来存储，这样可以简化程序设计的难度。  **一、一维数组**  **（一）一维数组的声明**  一维数组的声明方式有两种：  数据类型 [] 数组名 ;  数据类型 数组名 [];  数据类型可以是 Java 中的任意数据类型，数组名应为一个合法的标识符，[] 指明这是一个数组。这两种定义没有任何区别，大家可以根据自己的习惯选用。  Java 在声明数组的时候并不为其分配内存空间，因此在声明时可以不用指明数组长度，这样可以提高数组使用的灵活性，但在为其分配空间之前，数组是不能访问的。  **（二）一维数组的初始化**  在使用已经声明好的数组之前，需要对其进行初始化操作，即为数组分配内存空间，并为每个数组元素设置一个默认的初始化值。这里需要使用到运算符 new，其格式如下：  数组名 = new 数据类型 [ 数组长度 ]  通常可以将数组的声明和初始化操作合为一步完成，例如：  int MyArray[] = new int[10];  这里声明了一个整型数组 MyArray，并指定它的长度为 10，同时分配一块内存空间，供数组使用，其元素个数为 10。且此时数组中每个元素的值为 int 类型的默认初始值 0。数组在创建时，每个元素都有相同的默认值。  另外，还可以在数组声明时为数组元素定义另外的初始值，例如：  int a[] = {1,2,3,4,5};  这里声明了一个整型数组 a，数组中有 5 个元素，其初始值分别为 1、2、3、4、5。  **（三）一维数组的使用**  可以通过下面的格式访问数组中的任意一个元素：  数组名 [ 下标 ]  前面我们知道了，Java 中数组的下标是从 0 开始的，最大到数组长度减 1。另外，Java 要对数组元素进行越界检查以确保程序的安全性，如果发现下标越界，程序会抛出ArrayIndexOutOfBoundsException 异常。  下面这段代码是对前面求 Fibonacci 数列的前 20 项的程序的一个修改，这次是利用一维数组实现的，它的运行过程和结果请大家自行分析。  int f[] = new int[10];  f[0] = f[1] = 1;  for(int i = 2; i<10; i++)  f[i] = f[i-1] + f[i-2];  for(int i = 0; i<10; i++)  System.out.println("f["+i+"] = "+f[i]);  在一维数组中有一个最基本的应用是求数组元素的最大最小值，下面这段代码是求数组元素最大值，它的运行过程和结果请大家自行分析。  int a[] = {12,34,5,47,29,8,10},i,max;  max = a[0];  for(i = 0;i<a.length;i++)  if(a[i]>max)  max = a[i];  System.out.println(max);  数组可以用来存储数据类型一致的多个数据，通常和 for 循环配合，访问数组中的每一个元素。数组的长度可以用“数组名 .length”来获得，但不能使用这个方法在程序中重新设置数组长度。  **二、多维数组的使用**  Java 中的多维数组可以看作数组的数组。例如：对二维数组而言，其本身是一个一维数组，数组中的每一个元素又是一个一维数组。下面主要通过二维数组为例来进行说明，多维数组类似。  **（一）二维数组的声明**  二维数组的声明方式为：  数据类型 数组名 [][];  与一维数组一样，这时的数组元素也没有分配内存空间，同样要对数组进行初始化后，才能够访问数组中的每一个元素。  **（二）二维数组的初始化**  对二维数组来说，分配内存空间有两种方法。一种是直接分配，即一次性指定两个维数上的长度，例如：  int buffer[][] = new int[3][4];  二维数组 buffer 示意图如图 2−4 所示。    这样分配后的数组相当于一个矩阵，其每一行的长度是一样的。另一种是在一维数组的基础上，为第二维分配空间，例如：  int a[][] = new int[3][];  a[0] = new int[3];  a[1] = new int[4];  a[2] = new int[5];  这样可以为数组的每一行分配不同的长度，以满足一些特殊的需要。另外也可以使用这样的语句来对二维数组进行内存分配和初始化操作：  int[][] allRecord = {{70,85},{59,65},{59,60},{90,95},{65,50},{50,50}};  这样来指定一个 6 行 2 列的数组。  **（三）二维数组的使用**  同一维数组类似，可以通过给出二维数组的两个下标来访问数组中的元素。下面这段代码是求矩阵乘积的，它的运行过程和结果请大家自行分析。  int a[][] = new int[2][3];  int b[][] = {{1,5,2,8},{5,9,10,-3},{2,7,-5,-18}};  int c[][] = new int[2][4];  for(int i = 0; i<2; i++)  for(int j = 0; j<3; j++)  a[i][j] = (i+1)\*(j+2);  for(int i = 0; i<2; i++)  {  for(int j = 0; j<4; j++)  {  c[i][j] = 0;  for(int k = 0; k<3; k++)  c[i][j]+=a[i][k]\*b[k][j];  }  }  System.out.println(" 矩阵 A：");  for(int i = 0; i<2; i++)  {  for(int j = 0; j<3; j++)  System.out.print(a[i][j]+" ");  System.out.println();  }  System.out.println(" 矩阵 B：");  for(int i = 0; i<3; i++)  {  for(int j = 0; j<4; j++)  System.out.print(b[i][j]+" ");  System.out.println();  }  System.out.println(" 矩阵 C：");  for(int i = 0; i<2; i++)  {  for(int j = 0; j<4; j++)  System.out.print(c[i][j]+" ");  System.out.println();  }  与一维数组类似，多维数组通常也和 for 循环配合使用，以访问多维数组中的每一个元素，只是根据数组维数的多少，而使用多重循环来达到目的。多维数组的定义和使用与一维数组基本一致，只是由于多维数组的复杂性，在使用过程中要多加注意。  **任务实施**  (1)/\* 源程序文件名：Ex0207.java\*/  public class Ex0207  {  public static void main(String args[])  {  int[] mathRecord = {76,56,89,100,60,70,61,59,0,88,94,99};  int Total = mathRecord.length;  double pass = 0;  for(int i = 0; i < Total; i++)  {  if(mathRecord[i]>=60)  pass++;  }  System.out.println(" 全班总人数为："+Total);  System.out.println(" 及格率为："+pass/Total\*100+"%");  }  }  (2)/\* 源程序文件名：Ex0208.java\*/  public class Ex0208  {  public static void main(String args[])  {  int[][] allrecord = {{70,85},{59,65},{59,60},{90,95},{65,50},{50,50}};  double[] pass = new double[2];  int total = allrecord.length;  for(int i = 0; i < total; i++)  {  for(int j = 0; j < 2; j++)  if(allrecord[i][j]>=60)  pass[j]++;  }  System.out.println(" 全班总人数为："+ total);  System.out.println(" 语文及格率为："+ pass[0]/total\*100+ "%");  System.out.println(" 数学及格率为："+ pass[1]/total\*100+ "%");  }  }  **【学生】**思考、讨论。 | **通过教师讲解，了解数组的基本理论知识。** |
| **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数组，让学生知道做一个有“组织”的数据，方便组织统一管理。** | 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  简述二维数组的初始化。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示字符串  **任务描述**  编写如下程序：  （1）对 String 对象进行连接、替换、比较和查找等操作；  （2）对 StringBuffer 对象进行追加、插入和修改缓冲区长度等操作。  **任务目标**  1. 能够掌握 Java 中 String 类和 StringBuffer 类的区别。  2. 能够掌握 Java 中 String 类的使用方法。  3. 能够掌握 Java 中 StringBuffer 类的使用方法。  4. 既能理解 String 的“墨守成规”，又能理解 StringBuffer 的“灵活多变”，即而创新驱动发展。  **预备知识**  在许多语言中，字符串是一种基本数据类型，但在 Java 语言中，字符串是一种对，可以通过使用 String 或 StringBuffer 两个类来对其进行处理。  **一、String 类**  String 类封装了有关字符串的操作。这里的字符串是常量，即创建后就不可进行更改和变动。程序中只做字符串的比较、查找等操作时，通常使用 String 类，如果需要进行大量的字符串操作，应当使用 StringBuffer 类或字符数组，最终结果可以转换为 String 格式。  **（一）String 类对象的创建**  使用 String 类的构造函数是创建 String 对象的一种重要方法。除了使用以上构造函数之外，也可以使用更为熟悉的方式来定义字符串对象，例如：  String s1 = "Hello, World! ";  这种方式其实是 String s1 = new String（”Hello, World!”）的简化，为了兼容其他程序设计语言的编程习惯，Java 允许这样使用。  **（二）String 类对象的基本操作**  表 2−17 中列出了 String 类对象的一些基本操作。    请大家自行编写程序，掌握它们的使用方法。  **二、StringBuffer 类**  String 对象是一个常量，创建之后不能更改它的内容。而 StringBuffer 对象则相反，由于使用了缓冲区技术，可以很方便地修改 StringBuffer 对象的具体内容，这是它和String 对象本质上的区别。  缓冲区字符串类 StringBuffer 与 String 类相似，它具有 String 类的很多功能，甚至更丰富。它们主要的区别在于 StringBuffer 类可以方便地在缓冲区内被修改，如增加、替换字符或字符串。StringBuffer 对象可以根据需要自动增加存储空间，所以特别适合处理可变字符串。当完成操作后，可以使用 StringBuffer.toString( ) 方法或 String 类的构造函数将其转换为 String 类型。  **（一）StringBuffer 类对象的创建**  通常情况下，使用 StringBuffer 的构造函数来创建 StringBuffer 对象。  例如：  StringBuffer s1 = new StringBuffer(); // 长度为 16 的空串  StringBuffer s2 = new StringBuffer(40); // 长度为 40 的空串  StringBuffer s3 = new StringBuffer("Hello, World!"); // 内容为 "Hello, World!" 的串  **（二）StringBuffer 类对象的基本操作**  很多 String 类的方法，对 StringBuffer 类也同样适用。  **（三）StringBuffer 类和 String 类的区别**  String 类是常量类型，在对 String 类的对象进行操作后，并没有改变这个字符串的内容，看看下列语句：  String s1 = "Hello, world! ";  System.out.println(s1.replace('l', 'W'));  System.out.println(s1);  程序的输出结果是：  HeWWo, worWd! Hello, world!  第一行是替换字符串中一些字符后的结果，但这次操作产生了一个新的字符串对象，所以原来的字符串 s1 并没有发生变化。  StringBuffer 类是变量类型，在对 StringBuffer 类的对象进行操作时，是直接在原有字符串的内容上进行更改，看看下列语句：  StringBuffer s2 = new StringBuffer("Hello, world!");  System.out.println(s2.append("end"));  System.out.println(s2);  程序的输出结果是：  Hello, world!end  Hello, world!end  这里可以明显地看出来，在对 s2 进行操作时，实际上是在 s2 所在的末尾增加了一些字符，所以在操作完成后 s2 对象的内容发生了变化。  综上所述，String 对象在进行操作时，生成了一个新的 String 对象，原有的字符串不会发生改变；而 StringBuffer 对象在进行操作时，是在原有的字符串上进行操作，不会产生新对象，而是改变原有的字符串的内容。这是 String 类和 StringBuffer 类的最大也是最根本的区别。  **任务实施**  (1)/\* 源程序文件名：Ex0209.java\*/  public class Ex0209  {  public static void main(String args[])  {  String s1 = "Hello, world!";  String s2 = new String("Hello, world!");  s2 = " "+s2+" ";  String s3 = s1.replace('w','W');  if(s3.equals(s1))  System.out.println("s3 等于 s1。");  else  System.out.println("s3 不等于 s1。");  if(s3.equalsIgnoreCase(s1))  System.out.println("s3 等于 s1( 忽略大小写 )。");  else  System.out.println("s3 不等于 s1( 忽略大小写 )。");  System.out.println("s2 去除首尾空格后："+s2.trim());  System.out.println("s1 中第一个 o 字母出现在第 "+s1.indexOf('o')+" 位置。");  System.out.println("s1 中第一个 e 字母和后一个 l 字母之间的字符串是："+  s1.substring(s1.indexOf('e'), s1.lastIndexOf('l'));  }  }  在这个任务中我们可以看到，由于 Java 将字符串视为对象，所以可以使用一些很便捷的方法对其进行较为复杂的操作，观察下面这些语句：  s2 = " "+s2+" " // 将三个字符串连接起来  s3 = s1.replace('w','W') // 将 s1 中的 w 替换成 W  s3.equals(s1) // 比较 s1 和 s3  s3.equalsIgnoreCase(s1) // 比较 s1 和 s3, 忽略大小写  s2.trim() // 去掉 s2 首尾的空格  s1.indexOf('o') // 查找 s1 中第一次出现字母 o 的位置  s1.substring(s1.indexOf('e'),s1.lastIndexOf('l'))  // 取出 s1 中从第一个 e 字母开始 , 到后一个 l 字母结束的子串简单的几条语句，完成了字符串的连接、替换、比较、去首尾空格、查找、取子串等在其他语言中可能需要使用到循环、数组、判断等复杂语句才能完成的功能。这也从一个方面体现了面向对象程序设计的优越性。  (2)/\* 源程序文件名：Ex0210.java\*/  public class Ex0210  {  public static void main(String args[])  {  boolean b = true;  int i = 321;  long l = 123456;  String s1 = new String("Hello, World!");  StringBuffer s2 = new StringBuffer(s1);  System.out.println("s2 的内容是："+s2);  System.out.println("s2 内字符个数为："+s2.length());  System.out.println("s2 缓冲区长度为："+ s2.capacity());  System.out.println("s2 插入数据后："+ s2.append(b).insert(0,i).insert(3,l));  System.out.println("s2 内字符个数为："+s2.length());  System.out.println("s2 缓冲区长度为："+ s2.capacity());  s2.setLength(15);  System.out.println("s2 重设长度后："+s2);  }  }  **【学生】**思考、讨论。 | **通过教师讲解，了解字符串的基本理论知识。** |
| **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字符串，让学生知道既能理解 String 的“墨守成规”，又能理解 StringBuffer 的“灵活多变”，即而创新驱动发展。** | 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String 类对象的创建。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示思考与练习  **一、简答题**  （1）Java 中有哪些基本数据类型？怎样根据实际需要选择适当的数据类型？  （2）Java 中的表达式和数学上的表达式有哪些区别？  （3）什么是强制类型转换？在何时需要使用强制类型转换？使用强制类型转换时应  注意些什么？  （4）画出自动类型转换关系图。  （5）Java 语言中的三种循环控制语句是否可以在任何情况下互换？  （6）试区分 continue、break 和 return 语句的适用范围。  （7）Java 中数组的灵活性体现在哪些方面？  **二、选择题**  （1）下列程序错误的是（　　）。  int i = 100;  byte b = 10;  char c = 'a';  A. b=i B. c=b C. c=i D. i=b  （2）为一个 boolean 类型变量赋值时，可以用（　　）方式。  A. boolean=1; B. boolean a=（9>=10）;  C. boolean a=" 真 "; D. boolean a=False;  （3）以下变量定义中正确的是（　　）。  A. int I=123a B. float f=7.8f  C. char c='abc' D. String str='d'  （4）下列不是 Java 语言关键字的是（　　）。  A. extends B. abstract C. Class D. protected  （5）下面标识符中合法的是（　　）。  A. per%12 B. ab$ C. #hello D. \_1AB  （6）在 Java 语言中，1 个字符占（　　）个字节。  A. 2 B. 3 C. 4 D. 无数  （7）在 Java 语言中，逻辑常量只有（　　）个值。  A. 2 B. 3 C. 4 D. 无数  （8）变量定义如下：  long L = 15; float f = 9.8f; double d = 1.2; String s = "123";  int i = 18;  以下赋值语句不正确的是（　　）。  A. s=s+L; B. f=f+I; C. s=s+i+f+d; D. L=f+i;  （9）以下选项中变量均已正确定义，合法的赋值语句是（　　）。  A. a+b=2 B. a= =5 C. i− − D. y=int（i）  （10）下列说法正确的是（　　）。  A. 变量由字母、下划线、数字随意组成  B. 变量不能以数字开头  C. A 和 a 在 java 中是同一个变量  D. 不同类型的变量，可以起相同的名字  （11）以下语句的输出结果是（　　）。  String str = "12";  int x = 3,y = 4;  str = x+y+str;  System.out.println(str)  A. 712 B. 3412  C. 会产生编译错误 D. 3+4+12  （12）下列关于字符串的使用中，错误的是（　　）。  A. String str=new String（“java”）;  B. String str= “String”+100;  C. String str=100;  D. String str=null;  （13）关于下面程序片段，（　　）的结论是正确的。  String a = "java";  String b = "java";  String x = "ja";  String y = "va";  String c = x+y;  A. a 和 b 指向同一个实例对象，a 和 c 指向同一个实例对象  B. a 和 b 指向同一个实例对象，a 和 c 不指向同一个实例对象  C. a 和 b 不指向同一个实例对象，a 和 c 指向同一个实例对象  D. a 和 b 不指向同一个实例对象，a 和 c 不指向同一个实例对象  （14）下列值不为 true 的表达式有（　　）。  A. "john"= ="john"  B. "john".equals("john")  C. "john"="john"  D. "john".equals(new String("john"))  **【学生】**思考、讨论。 | **通过教师讲解，了解思考与练习的基本理论知识。** |
| **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思考与练习，让学生能够更加熟悉String 类。** | 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  String 对象和 StringBuffer 对象在使用时应如何选择？ | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示实训任务  （1）编写程序：分别为一个逻辑变量、字符变量、整型变量、单精度浮点型变量、双精度浮点型变量赋值，并显示变量的内容。  （2）编写程序：将字符串“abc”转换为大写“ABC”[ 不使用 String 类的 toUpperCase( ) 方法 ]。  （3）编写程序：将给定的摄氏温度转换成华氏温度。公式为：f = 1.8×c + 32。  （4）编写程序：根据给定的身高和体重计算肥胖指数 BMI，并输出相应的健康标准。  BMI 计算公式为：BMI = 体重（kg）÷[ 身高（m）]2。  根据亚洲人健康标准，BMI 值在 18.5 以下为体重不足，18.5～23 为健康，23～25 为超重，25～30 为肥胖，30 以上则为严重肥胖。  （5）编写程序：求 10 行 10 列的杨辉三角，如图 2−6 所示。    **【学生】**思考、讨论。 | **通过教师讲解，了解实训任务的基本理论知识。** |
| **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实训任务，让学生更加熟悉所学的实际操作。** | 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  编写程序：计算一句英文句子里面有多少个单词，输出单词个数和每一个单词。 | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 兴趣是学习最好老师，因此在教学中如何创设良好的教学情景，使学生乐学，好学是我们信息学教师要不断探索的方面。 | |