**Java语言的重要特点**

1. **Java语言是面向对象的oop**
2. **Java语言是健壮的。Java的强类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。**
3. **Java语言是跨平台性的**
4. **Java语言是解释型的**

**解释性语言：JavaScript、PHP、java 编译性语言：c/c++**

**区别是：解释性语言，编译后的代码，不能直接被机器执行，需要解释器来执行，编译性语言，可以直接被机器执行，c/c++**

**Java运行机制及运行过程**

**Java核心机制：java虚拟机【JVM】**

1. **JVM是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器，包含在JDK中。**
2. **对于不同的平台，有不同的虚拟机。**
3. **Java虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处运行”**

**Jdk基本介绍**

1. **jdk的全称 Java development kit java开发工具包**

**jdk = jre+Java开发工具【Java、javac、javadoc、Javap等】**

1. **jdk是提供给Java开发人员使用的，其中包含了java的开发工具，也包括了jre。**

**Jre基本介绍**

1. **jre Java runtime environment Java运行环境**

**jre = jvm +Java的核心库【类】**

1. **包括java虚拟机和java程序所需的核心库等，如果想运行一个开发好的Java程序，计算机中只需要安装jre即可。**

**Jdk、jre和JVM的包含关系**

1. **JDK = JRE + 开发工具集（例如javac、java编译工具等）**
2. **JRE = JVM + Java SE标准类库（java核心类库）**
3. **JDK = JVM + Java SE标准类库 + 开发工具集**
4. **如果想要运行开发好的.class文件 只需要JRE**

**什么是运行**

1. **有了可执行的java程序（hello.class）**
2. **通过运行工具java.exe 对字节码文件进行执行，本质就是.class 装载到jvm机执行**

**Java程序开发注意事项**

**对修改后的Hello.java源文件需要重新编译，生成新的class文件后，再进行执行，才能生效。**

**Java开发注意事项和细节说明**

1. **Java源程序以.java为扩展名。源文件的基本组成部分是类（class），如本类中的hello类。**
2. **Java应用程序的执行入口是main( )方法。它有固定的书写编写方式。**
3. **Java语言严格区分大小写**
4. **Java方法由一条条语句构成，每个语句以“；”结束。**
5. **大括号都是成对出现的，缺一不可 [习惯，先写{} 再写代码]**
6. **一个源文件中最多只能有一个public类。其它类的个数不限。**
7. **如果源文件中包含一个public类，则文件名必须按改类名命名！**
8. **一个源文件中最能只能有一个public类。其它类的个数不限，也可以将main方法写在非public类中，然后指定运行非public类，这样入口方法就是非public 的 main 方法。**

**Java转义字符**

**1、\t 一个制表位，实现对齐的功能**

**2、\n 换行符**

**3、\\ 一个\**

**4、\” 一个”**

**5、\’ 一个’**

**6、\r 一个回车 System.out.println(“Angelczack\r上海”)；**

**初学Java易犯错误**

1. **找不到文件**

**解决方法：源文件名不存在或者写错，或者当前路径错误**

1. **主类名和文件名不一致**

**解决方法：声明为public的主类应与文件名一致，否则编译失败**

1. **缺少分号**

**解决方法：编译失败，注意错误出现的行数，再到源代码中指定位置改错。**

**注释（comment）**

**介绍：**

**用于注解说明结束程序的文字就是注释，注释提高了代码的阅读性；注释是一个程序员必须具有的良好编程习惯。将自己的思想通过注释先整理出来，再用代码去体现。**

**Java中的注释类型**

**1、单行注释**

**基本格式 //注释文字**

**2、多行注释**

**基本格式 /\*注释文字\*/**

**3、文档注释**

**注释内容可以被JDK提供的工具javadoc所解析，生成一套以网页为文件形式体现的该程序的说明文档，一般写在类。**

**基本格式 Javadoc -d 文件夹名（目录） -xx -yy Demo.java**

**Javadoc 标签**

**@author 标识一个类的作者**

**@version 指定类的版本**

**@return 说明返回的类型**

**@param 说明一个方法的参数**

**@see 说明一个到另一个主题的链接**

**使用细节**

**1）被注释的文字，不会被JVM解释执行**

**2）多行注释里面不允许有多行注释嵌套**

**Java代码规范**

**1、类、方法的注释，要以javadoc的当时来写**

**2、非Java Doc的数值，往往是给代码的维护者看的，着重告诉读者为什么这样写，如何修改，注意什么问题等**

**3、使用tab操作，实现缩进，默认整体向右边移动，时候用shift+tab整体向左移**

**4、运算符和 = 两边习惯性各加一个空格。 比如：2 + 4 \* 5 + 345 -89**

**5、源文件使用utf-8编码**

**6、行宽度不要超过80字符**

**7、代码编写次行风格和行尾风格**

**DOS命令**

**DOS介绍：Disk Operating System 磁盘操作系统，简单说一下windows的目录结构**

**常用的dos命令**

**1、查看当前的目录有什么**

**dir dir d:\abc2\test200**

**2、切换到其他盘下**

**Cd /D c:**

**3、切换到当前盘的其他目录下(相对路径和绝对路径的展示方法)**

**1) cd d:\abc2\test200 cd ..\..\abc2\test200**

**4、切换到上一级**

**cd ..**

**5、切换到根目录**

**cd \**

**变量原理**

**为什么需要变量：变量是程序的基本组成单位**

**变量的介绍**

**变量相当于内存中一个数据存储空间的表示，你可以把变量看作是一个房间的门牌号，通过门牌号我们可以找到房间，而通过变量名可以访问到变量值**

**变量使用的基本步骤**

**1）声明变量**

**int a；**

**2) 赋值**

**a = 60; //应该这么说：把60赋给a**

**3) 使用System.out.println(a);**

**//也可以一步到位[int a =60; 通常我们是一步完成]**

**变量使用注意事项**

**1、变量表示内存中的一个存储区域 [ 不同的变量，类型不同，占用的空间大小不同，比如：int 4个字节，double 就是 8个字节]**

**2、该区域有自己的名称 [变量名] 和类型 [数据类型]**

**3、变量必须先声明，后使用，即有顺序**

**4、该区域的数据可以在同一类型范围内不断变化**

**5、变量在同一个作用域内不能重名**

**6、变量 = 变量名 + 值 + 数据类型，这是变量的三要素**

**Java程序中 +号的使用**

**1、当左右两边都是数值型时，则做加法运算**

**2、当左右两边有一方为字符串，则做拼接运算**

**数据类型**

**每一种数据都定义了明确的数据类型，在内存中分配了不同大小的内存空间(字节)**

**一、基本数据类型**

**1、数值型**

**1）整数类型，存放整数(byte[1],short[2],int[4],long[8])**

**2）浮点（小数）类型(float[4],double[8])**

**2、字符型(char[2]),存放单个字符 ‘a’**

**3、布尔型(boolean[1]),存放true，false**

**二、引用数据类型**

**1、类 class**

**2、接口 interface**

**3、数组 [ ]**

**整数类型**

**整形的使用细节 inDetail.java**

1. **Java各整数类型有固定的范围和字段长度，不受具体OS【操作系统】 的影响，以保证java程序的可移植性。**
2. **Java的整型常量声明为int型，声明long型常量需后加‘l’或者‘L’**
3. **Java程序中变量常声明为int型，除非不足以表示大数，才使用long**
4. **bit:计算机中最小的存储单位 byte：计算机中最基本的存储单元 1byte =8bit**

**浮点类型**

**Java的浮点类型可以用来表示一个小数**

**浮点型的分类**

**单精度型 4个字节**

**双精度型 8个字节**

**注：浮点数在机器中存放形式的简单说明，浮点数 = 符号位 + 指数位 + 尾数位，尾数部分可能丢失，造成精度损失（小数都是近似值）**

**浮点类型使用细节 floatDetail.java**

1. **与整数类型相比，Java浮点类型也有固定的范围和字段长度，不受具体OS的影响**
2. **Java的浮点型常量 默认为double型，声明float型常量，须后加’f’或者‘F’**
3. **浮点型常量有两种表示形式**

**十进制的形式：如 5.12 512.0f .512(必须有小数点)**

**科学计数法形式： 5.12e2 [ ] 5.12E-2 [ ]**

1. **通常情况下，应该使用double型，因为它比float型更精确**

**Java API文档**

1. **api，是Java提供的基本编程接口 （java提供的类还有相关的方法）。**

**中文在线文档：** [**https://www.matools.com**](https://www.matools.com)

1. **java语言提供了大量的基础类，因此 Oracle公司也为这些基础提供了响应的API文档，用于告诉开发者如何使用这些类，以及这些类里包含的方法。**
2. **java类组织形式**

**字符类型**

**字符类型可以表示单个字符，字符类型是char，char是两个字节(可以存放汉字)，多个字符我们用字符串String**

**案例 char01.java**

**说明：字符类型可以直接存放一个数字**

**字符型使用细节**

1. **字符常量是用单引号(‘ ’)括起来的单个字符**
2. **Java中还允许使用转义字符， ‘\t’**
3. **在java中，char的本质是一个整数，在输出时，是unicode码对应的字符。**

[**http://tool.chinaz.com/Tools/Unicode.aspx**](http://tool.chinaz.com/Tools/Unicode.aspx)

1. **可以直接给char赋一个整数，然后输出时，会按照对应的unicode字符输出**
2. **Char类型是可以进行运算的，相当于一个整数，因为它都对应有Unicode码。**

**字符类型的本质探讨**

1. **字符型 存储到计算机中 ， 需要将字符对应的码值 （整数） 找出来，比如‘a’**

**存储：‘a’==>码值97 ==>二进制 ==> 存储**

**读取：二进制 ==>97 ==> ‘a’==> 显示**

1. **字符和码值的对应关系是通过字符编码表决定的**

**介绍字符编码表**

**ASCII（一个字节表示，一个128个字符，实际上一个字节可以表示256个字符，只用128个）**

**Unicode （固定大小的编码 使用两个字节来表示字符 字母和汉字统一都是占用两个字节 这样浪费空间）**

**Utf-8 (大小可变的编码 字母使用1个字节 汉字使用3个字节)**

**Gbk （可以表示汉字，而且范围广，字母使用1个字节 汉字使用2个字节）**

**Gb2312 （可以表示汉字 gb2312 < gbk）**

**Big5 (繁体中文 ，台湾 香港)**

**布尔类型**

1. **布尔类型也叫做Boolean类型，Boolean类型数据只允许取值true或者false，无null**
2. **Boolean类型占用1个字节**
3. **Boolean类型适用于逻辑运算，一般用于流程控制**

**If条件控制语句**

**While循环控制语句**

**Do-while循环控制语句**

**For循环控制语句**

**案例 boolean01.java**

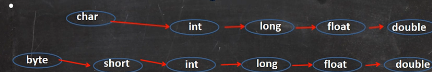
**说明：不可以用0或非0的整数代替false或true，这点和c语言不同**

**自动类型转化（基本数据类型转换）**

**介绍**

**当java程序在进行赋值或者运算时，精度小的类型自动转换为精度大的数据类型，这个就是自动类型转换**

**数据类型按精度（容量）大小排序为**



**案例 autoConvert.java**

**自动类型转换注意和细节**

1. **有多种类型的数据混合运算时，系统首先自动将所有数据转换为容量大的那种数据类型，然后进行运算。**
2. **当我们把精度（容量大）的数据类型赋值给精度（容量）小的数据类型时，就会报错，反之就会进行自动类型转换**
3. **（byte，short） 和char之间不会相互自动转换**
4. **Byte，short，char 他们三者可以计算，在计算时先转换为int类型**
5. **Boolean不参与转换**
6. **自动提升原则：表达式结果的类型自动提升为 操作数中最大的类型**

**案例：autoconvertDetail**

**强制类型转换**

**介绍**

**自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转换符()，但可能造成精度降低或溢出，格外要注意。**

**案例：forceConvert.java**

**强制类型转化细节说明**

**案例 forceconvertDetail.java**

1. **当进行数据的大小从 大 --- 小，就需要使用强制转换**
2. **强制符号只针对于最近的操作数有效，往往会使用小括号提升优先级**
3. **Char类型可以保存int常量值，但不能保存int变量值，需要转换**
4. **Byte和short类型给在进行运算时，当作int类型处理**

**基本数据类型和String类型的转换**

**介绍**

**在程序开发中，我们经常需要将基本数据类型转成String类型。或者将String类型转成基本数据类型**

**基本类型转String类型**

**语法：将基本类型的值 + “”即可**

**案例:StringTobasic.java**

**String类型转基本数据类型**

**语法：通过基本类型的包装类调用parseXX方法即可**

**案例：StringTobasic.java**

**基本数据类型和String类型的转换**

**案例：StringTobasicDetail.java**

1. **在将String 类型转换为 基本数据类型时，要确保String类型能过成有效的数据，比如 我们可以把 “123”，转成一个数整数，但是不饿能把“hello” 转成一个整数**
2. **如果格式不正确，就会抛出异常，程序就会终止，这个问题在异常处理章节中，会处理。**

**算数运算符**

**算术运算符是对数据类型的变量进行运算的，在Java程序中使用的非常多**

**案例：ArithmeticOperator.java**

1. **取模（求余） 本质公式 a%b 等价于 a – a/b \*b**
2. **自增：作为独立语句使用，前++和后++都完全等价于 i=i+1；作为表达式使用，++i先自增后赋值，i++先赋值后自增**

**关系运算符(比较运算符)**

**介绍**

1. **关系运算符的结果都是boolean型，也就是要么是true，要么是false**
2. **关系表达式 经常用在 if结构的条件中或循环结构的条件中**

**案例 RelationalOperator.java**

**细节说明**

1. **关系运算符的结果都是boolean型**
2. **关系运算符组成的表达式，我们称为关系表达式**
3. **比较运算符 == 不能误写为 =**

**逻辑运算符**

**用于连接多个条件（多个关系表达式），最终的结果也是一个boolean值**

1. **短路与 && 短路或 || 取反 ！**
2. **逻辑与 & 逻辑或 | 逻辑异或^**

**运算规则**

1. **逻辑与 a&b :当a与b同时为true时，结果为true，否则为false**
2. **短路与 a&&b ：当a和b同时为true时，结果为true，否则为false**
3. **逻辑或 a|b ：当a和b，有一个为true，结果为true，否则为false**
4. **短路或 a || b ：当a和b，有一个为true，结果为true，否则为false**
5. **取反 !a ：当a为true时，结果为false，当a为false时，结果为true**
6. **逻辑异或 a^b : 当a和b不同时，结果为true，否则为false**

**案例 logicoperator01.java**

**&& 与 & 的区别**

1. **短路与 && ：如果第一个条件为false，则第二个条件不会判断，最终结果为false**
2. **逻辑与 & ：不管第一个条件为false，第二个条件都要判断，效率低**
3. **开发中，我们使用的基本都是短路与 &&，效率高**

**|| 与 | 的区别**

1. **短路或 || ：如果一个条件为true，则第二个条件不会判断，最终结果为true，效率高**
2. **逻辑或 | ：不管第一个条件是否为true，第二个条件都要判断，效率低**
3. **开发中，我们基本使用 ||**

**赋值运算符**

**介绍**

**赋值运算符就是将某个运算后的值 ,赋给指定的变量**

**分类**

**基本赋值运算符 int a= 10;**

**复合赋值运算符 += -= \*= /= %=**

**案例 AssignOperator.java**

**赋值运算符的特点**

1. **运算顺序从右往左 int num = a + b +c；**
2. **赋值运算符的左边 只能是变量 右边可以是变量、表达式、常量值**
3. **复合赋值运算符等价于下面的效果**

**a+=3 a =a+3**

1. **复合赋值运算符会进行类型转换**

**byte b =2; b+=3; b++;**

**三元运算符**

**条件表达式 ？表达式1 ： 表达式2；**

1. **如果条件表达式为true，运算后的结果是表达式1**
2. **如果条件表达式为false，运算后的结果是表达式2**

**案例 TernaryOperator.java**

**使用细节**

**案例 TernaryoperatorDetail.java**

1. **表达式1和表达式2要为可以赋给接收变量的类型（或可以自动转换）**
2. **三元运算符可以转成 if ---else语句**

**运算符的优先级**

**1.运算符有不同的优先级，所谓优先级就是表达式运算中的运算顺序，如右表，上一行运算符总优先于下一行。**

**2.只有单目运算符、赋值运算符是从右往左运算的。**

**概括种类**

**1）（）{}等**

**2）单目运行 ++ --**

**3）算术运算符**

**4）位移运算符**

**5）比较运算符**

**6）逻辑运算符**

**7）三元运算符**

**8）赋值运算符**

**标识符的命名规则和规范**

**标识符的概念**

**1.Java对各种变量、方法和类等命名规则使用的字符序列称为标识符**

**2.凡是自己可以起名字的东方都叫做标识符 int num = 90；**

**标识符的命名规则（必须遵守）**

**1.由26个英文字大小写，0-9，\_或 $组成**

**2.数字不可以开头**

**3.不可以使用关键字和保留字，但能包含关键字和保留字**

**4.Java中严格区分大小写，长度无限制。**

**5.标识符不能包含空格。**

**标识符的命名规范（更加专业）**

**1.包名：对单词组成时所有字母都小写 ：aaa.bbb.ccc //比如com.hsp.crm**

**2.类名、接口名 ： 多单词组成时，所有单词的首字母大写，XxxYyyZzz（大驼峰法）**

**3.变量名、方法名：多单词组成时，第一个单首字母大写，第二个单词开始每个单词首字母大写：xxxYyyZzz 比如：tankShotGame（小驼峰法 简称驼峰法）**

**4.常量名：所有字母都大写，多单词时每个单词用下划线连接**

**XXX\_YYY\_ZZZ 比如：TAX\_RATE**

**5.后面的类、包、接口、等时，命名规范要这样遵守**

**关键字的定义和特点**

**定义：被Java语言赋予了特殊含义，用作专门用途的字符串**

**特点：关键字中所有字母都为小写**

游戏的屏幕

描述已自动生成

**保留字**

**介绍**

**Java保留字：现有java版本尚未使用，但以后的二版本可能会作为关键字使用。自己命名标识符时要避免使用这些保留字**

**byValue、cast、future、generic、inner、operator、outer、rest、var、goto、const**

**键盘输入语句**

**介绍**

**在编程中，需要接收用户输入的数据，就可以使用键盘输入语句来获取。**

**Input.java ， 需要一个扫描器（对象），就是Scanner**

**步骤：**

**1）导入该类所在包，java.unil.\***

**2)创建该类对象（声明变量）**

**3）调用里面的功能**

**进制**

**对于整数，有四种表示方式**

**1.二进制 0-1**

**2.十进制 0-9**

**3.八进制 0-7**

**4.十六进制 0-9 A-F(10-15)**

**案例：BinaryTest.java**

**二进制转换为十进制**

**规则：从最低位（右边）开始，将每一位上的数据取出来，乘以2的（位数-1）次方，然后求和。**

**八进制转换为十进制**

**规则：从最低位开始，将每个位上的数提取出来，乘以8的(位数-1)次方，然后求和。**

**十六进制转换为十进制**

**规则：从最低位（右边）开始，将每个位的数提取出来，乘以16的（位数-1）次方，然后求和。**

**十进制转换为二进制**

**规则：将该数不断除2，直到商为0为止，然后将每步得到的余数倒过来，就是对应的二进制。**

**十进制转换为八进制**

**规则：将该数不断除以8，直到商为0位为止，然偶将每步得到的余数倒过来，就是对应的八进制。**

**十进制转换为十六进制**

**规则：将该数不断除以16，直到商为0为止，然后将每步得到的余数倒过来，就是对应的十六进制。**

**二进制转换为八进制**

**规则：从低位开始，将二进制数每三位一组，转成对于八进制数即可。**

**二进制转换为十六进制**

**规则：从低位开始，将二进制数每四位一组，转成对应的十六进制数即可。**

**八进制转换为二进制**

**规则：将八进制数每1位，转成对应的3位二进制数即可**

**十六进制转换为二进制**

**规则：将十六进制数每1位，转成对应的4位二进制数即可。**

**二进制在运算中的说明**

**1.二进制是逢2进位的进位制，0、1是基本算符。**

**2、现代的电子计算机技术全部采用二进制，因为它只有0、1两个数字符号，非常简单方便，易于用电子方式实现。计算机内部处理的信息，都是采用二进制数来表示的。二进制数用0和1两个数字及其组合来表示任何数。仅为规则是“逢2进1“，数字1在不同的位商代表不同的值，按从右往左的次序，这个值以二倍递增。**

**原码、反码、补码**

**对于有符号的而言：**

**1.二进制的最高位是符号位，0表示整正数，1表示负数**

**2.正数的原码、反码、补码都一样**

**3.负数的反码 = 它的原码符号位不变，其他位按位取反**

**4.负数的补码 = 它的反码 +1 负数的反码 = 负数的补码 – 1**

**5.0的反码、补码都是0**

**6.Java没有无符号数，换言之，Java中的数都是有符号的**

**7.在计算机运算的时候，都是以补码的方式来运算的**

**8.当我们看运算结果的时候，要看他的原码**

**位运算符**

**Java中有7个位运算  
& | ^ ~ >> << >>>**

**分别是 按位与 按位或 按位异或 按位取反**

**按位与：两位全为1，结果为1，否则为0**

**按位或：两位有一位为1，结果为1，否则为0**

**按位异或：两位一个为0，一个为1，结果为1，否则为0**

**按位取反：0->1 1->0**

**案例：BitOperator.java**

**顺序控制**

**程序控制介绍**

**程序从上到下逐行地执行，中间没有任何判断和跳转。**

**顺序控制执行的举例和注意事项**

**Java中定义变量时采用合法的前向引用。如**

**Int num1 = 12;**

**Int num2 = num + 1;**

**错误形式：**

**Int num2 = num1 + 1;**

**Int num1 = 12;**

**分支控制if-else**

**让程序有选择的执行，分支控制有三种**

**1）单分支**

**2）双分支**

**3）多分枝**

**单分支**

**基本语法**

**If(条件表达式){**

**执行代码块；（可以有多条）**

**说明：当条件表达式为true时，就会执行 { }的代码。如果为false，就不执行，特别说明，如果{ }中只有一条语句，则可以暴不用 { } ，建议写上。**

**案例：If01.java**

**双分支**

**基本语法**

**If(条件表达式){**

**执行代码块1;**

**}**

**Else{**

**执行代码块2；**

**}**

**说明：当条件表达式成立，及执行代码块1，否则执行代码块2.如果执行代码块 只有一条语句，则 { } 可以省略，否则不可以。**

**案例：If02.java**

**多分支**

**基本语法**

**If(执行表达式1){**

**执行代码块1；**

**}**

**Else if(执行表达式2) {**

**执行代码块2；**

**}**

**…**

**else{**

**执行代码块n；**

**}**

**说明：**

**1.当条件表达式1成立时，即执行代码块1**

**2.如果表达式1不成立，才去判断表达式2是否成立**

**3.如果表达式2成立，就执行代码块2**

**4.以此类推，如果所有的表达式都不成立**

**5.则执行else的代码块，注意，只能有一个执行入口**

**特别说明：**

**1）多分支 可以没有else，如果所有的条件表达式都不成立，则一个执行入口都没有**

**2）如果有else，如果所有的条件表达式都不成立，则默认执行else语块。**

**多分支案例：If03.java**

**嵌套分支**

**基本介绍**

**在一个分支结构中又完整的嵌套了另一个完整的分支结构，里面的分支的结构称为内层分支外面的分支结构称为外层分支。规范：不要超过3层。**

**案例：NestedIf.java**

**Switch分支结构**

**基本语法**

黑板上的文字

描述已自动生成

**说明：**

**1.switch关键字，表示switch分支**

**2.表达式对应一个值**

**3.case 常量1：当表达式1的值等于常量1，就执行 语句块1**

**4.break：表示退出switch**

**5.如果和case 常量1匹配，就执行语句块1，如果没有匹配，就继续匹配常量2**

**6.如果都没有匹配上，执行default**

**案例：Switch01.java**

**Switch细节讨论**

**1.表达式数据类型，应和case后的常量类型一致，或者是可以自动转成可以相互比较的类型，比如输入的是字符，二常量值是int**

**2.switch（表达式）中表达式的返回值必须是 ：（byte，short，int，char，enum，String）**

文本

描述已自动生成

**3.case子句中的值必须是常量，而不能是变量**

**4.default子句是可选的，当没有匹配的case时，执行default**

**5.break语句用来在执行完一个case分支后使程序跳出switch语句块，如果没有写break，程序会顺序执行到switch结尾。**

**案例：SwitchDetail.Java**

**练习：SwitchExercise.java.**

**Switch和if的比较**

1. **如果判断的具体数值不多，而且符合byte、short、int、char、enum、String这六种类型，虽然两个语句都可以使用，建议使用switch语句。**
2. **其他情况：对区间判断，对结果为Boolean类型判断，使用if，if的使用范围更广。**

**For循环控制**

**基本介绍：听其名而知其意，就是让你的代码可以循环的执行。**

**案例：For01.java**

**基本语法**

文本

描述已自动生成

**说明：**

1. **for关键字，表示循环控制**
2. **2.for有四要素：（1）循环变量初始化（2）循环条件（3）循环操作（4）循环变量迭代**
3. **循环操作，这里可以有多条语句，也就是我们要循环执行的代码**
4. **如果 循环操作（语句）只有一条语句，可以省略 { }，建议不要省略**

**注意事项和细节说明**

**案例：ForDetail.java**

1. **循环条件是返回一个布尔值的表达式**
2. **For(;循环判断条件;) 中的初始化和变量迭代可以写到其它地方，但是两边的分号不能省略。**
3. **循环初始值可以有多条初始化语句，但要求类那个一样，并且中间用逗号隔开，循环变量迭代也可以有多条变量迭代语句，中间用逗号隔开。**

**While循环控制**

**基本语法**

**循环变量初始化；**

**While（循环条件） {**

**循环体（语句）；**

**循环变量迭代；**

**}**

**说明：**

1. **while循环也有四要素**
2. **只是四要素放的位置，不一样**

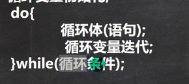
**案例：While01.java**

**注意事项和使用细节**

1. **循环条件是返回一个布尔值的表达式**
2. **While循环是先判断再执行的语句**

**Do..while循环控制**

**循环变量初始化：**



**说明：**

1. **do while 是关键字**
2. **也有循环四要素，只是位置不一样**
3. **先执行，再判断，也就是说，一定会执行一次**
4. **最后有一个分号；**
5. **While和do...while 的区别**

**案例：DoWhile01.java**

**注意事项和细节说明**

1. **循环条件时返回一个布尔值的表达式**
2. **Do..while循环是先执行，再判断，因此它至少执行一次**

**案例：DoWhileExercise01.java DoWhileExercise02.java**

**多重循环控制（重点）**

**介绍**

1. **将一个循环放到另一个循环体内，就形成了嵌套循环。其中for、while、do.while均可以作为外层循环和内层循环，建议一般使用2层，最多不要超过3层。**
2. **实质上，嵌套循环就是把内层循环当成外层循环的循环体。当只有内层循环的循环条件为false时，才会完全跳出内层循环，才可以结束外层的当次循环，开始下一次的循环**
3. **设外层循环次数为m次，内层为n次，则内层循环体实际需要执行m\*n次**

**练习：MulForExercise01.java**

**空心金字塔练习：Stars.java**

**跳转控制语句 break**

**基本介绍**

**Break语句用于终止某个语句块的执行，一般使用在switch或者循环[for,while,do..while]中。**

**基本语法**

**{**

**…**

**Break;**

**…**

**}**

**案例：Break01.java**

**注意事项和细节说明：**

1. **break语句出现在多层嵌套的语句块中时，可以通过标签指明要终止的是哪一层语句块 BreakDetail.java**
2. **标签的基本使用**
3. **break 语句可以指定退出哪一层**
4. **label1 是标签 由程序员指定**
5. **break后指定到哪一个label 就退出到哪里**
6. **在实际的开发中，尽量不要使用标签**
7. **如果没有指定break，默认退出最近的循环体**

**练习：BreakExercise.java**

**跳转控制语句 continue**

**基本介绍**

1. **continue语句用于结束本次循环 继续执行下一次循环**
2. **continue语句出现在多层嵌套的循环语句体中，可以通过标签指明要跳出的是哪一层循环，这个和前面的标签的使用的规则一样。**
3. **基本语法**
4. **{**
5. **…**
6. **Continue;**
7. **…**
8. **}**

**案例：Continue01.java ContinueDetail.java**

**跳转控制语句 return**

**介绍**

**Return使用在方法，表示跳出所在的方法**

**注意：如果return 写在 main方法 退出程序…**

**案例：Return01.java**

**数组介绍**

**数组可以存放多个同类型的数据。数组也是一种数据类型，是引用类型。**

**即：数组就是一组数据**

**数组快速入门**

**Array01.java**

**数组的使用**

**使用方法1-动态初始化**

**数组的定义**

**数据类型 数组名[] = new 数据类型[大小]**

**Int a[] = new int[5]; //创建了一个数组，名字为a，存放5个int**

**说明：这是定义数组的一种方法。**

**案例：Array02.java**

**使用方法2-动态初始化**

**先声明数组**

**语法：数据类型 数组名[]; 也可以 数据类型[] 数组名;**

**Int a[]; 或者 int[] a;**

**创建数组**

**语法：数组名 = new 数据类型[大小]；**

**a = new int[10];**

**案例：Array03.java**

**使用方式3-静态初始化**

**初始化数组**

**语法：数据类型 数组名[] = {元素值，元素值…}**

**Int a[] = {2,5,6,7,8,9}; 如果知道数组有多少元素，具体值上面的用法相当于**

**Int a[] = new int[9];**

**案例：Array04.java**

**数组的注意事项和细节说明**

1. **数组是多个相同类型数据的集合，实现对这些数据的统一管理**
2. **数组种的元素可以是任何数据类型，包括基本数据类型和引用类型，但是不能混用。**
3. **数组创建后，如果没有赋值，有默认值**

**int 0 short 0 byte 0 long 0 float 0.0 double 0.0 char \u0000 boolean false String nyll**

1. **使用数组的步骤：1）声明数组并开辟空间 2）给数组各个元素赋值 3）使用数组**
2. **数组的下标是从0开始的**
3. **数组下标必须在指定范围内使用，否则报：下标越界异常，比如：**

**int [] arr = new int[5]; 则有效下标为0-4**

1. **数组属引用类型，数组型数据是对象（object）**

**案例：ArrayExercise01.java ArrayExercise02.java**

**数组赋值机制**

1. **基本数据类型赋值，这个值就是具体的数据，而且相互不受影响**

**int n1 =2; int n2 = n1;**

1. **数组在默认情况下是引用传递，赋的值是地址**

**案例：ArrayAssign.java**

**数组拷贝**

**案例：ArrayCopy.java**

**数组反转**

**案例：ArrayReverse.java**

**数组添加**

**要求：实现动态的给数组添加元素效果，实现对数组扩容。 ArrayAdd.java**

1. **原始数组使用静态分配 int[] arr = {1,2,3}**
2. **增加的元素，直接放在数组的最后 arr = {1,2,3,4}**

**arrNew = {1,2,3,4}**

**3）用户可以通过如下方法来决定是否继续添加，添加成功，是否继续 y/n**

**数组缩减**

**案例：ArrayReduce.java**

**排序的介绍**

**排序是将一群数据，依指定的顺序进行排列的过程。**

**排序的分类：**

**1、内部排序**

**指将需要处理的所有数据都加载到内部储存器中进行排序。包括（交换式排序法、选择式排序法和插入式排序法）；**

**2、外部排序法**

**数据量过大，无法全部加载到内存中，需要借助外部存储进行排序，包括（合并排序法和直接合并排序法）**

**冒泡排序法**

**冒泡排序（Bubble Sorting）的基本思想是：通过对待排序序列从后往前（从下标较大的元素开始），依次比较相邻元素的值，若发现逆序则交换，使值较大的元素逐渐从前向后移，就像水底的气泡一样逐渐向上冒。**

**案例：BubbleSort.java**

**查找**

**介绍**

**在Java中，我们常用的查找有两种**

**1、顺序查找 SeqSearch.java**

**2、二分查找**

**二维数组**

**案例：TwoDimensionalArray01.java**

**使用方法1：动态初始化 TwoDimensionalArray02.java**

**1) 语法：类型[][] 数组名 = new 类型[大小] [大小]**

**2) 例：int a[][] = new int[2][3]**

**使用方法2：动态初始化**

**1）先声明： 类型 数组名[][]; TwoDimensionalArray02.java**

**2) 再定义（开辟空间） 数组名 = new 类型[大小][大小]**

**3）赋值（有默认值）， 比如int 类型就是0**

**使用方法3-动态初始化-列数不确定 TwoDimensionalArray03.java**

**使用方法4-静态初始化 TwoDimensionalArray04.java**

**1、定义 类型 数组名[][] = {{v1,v2},{v1,v2},{v1,v2},{v1,v2}}**

**2、使用即可 [固定方式访问]**

**应用案例**

**杨辉三角形 YangHui.java**

**二维数组使用细节和注意事项**

1. **一维数组的声明方式有：**

**int[] x 或者 int x[]**

1. **二维数组的声明方式有：**

**int[][] y 或者 int y[] 或者 int y[][]**

1. **二维数组实际上是由多个一维数组组成的，它的各个一维数组的长度可以相同，也可以不同**

**类与对象**

**使用现有的技术解决 Object01.java**

1. **单独的定义变量解决**
2. **使用数组解决**

**类和对象的区别和联系**

1. **类是抽象的，概念的，代表一类事物，即它是数据类型**
2. **对象是具体的，实际的，代表一个具体事物，即实例**
3. **类是对象的模板，对象是类的一个个体，对应一个实例**

**属性和成员变量**

**基本介绍**

1. **从概念或者叫法上看：成员变量 = 属性 = field 即成员变量是用来表示属性的**
2. **属性是类的一个组成部分，一般是基本数据类型，也可以是引用类型（对象、数组）**

**注意事项和细节说明**

1. **属性的定义语法同变量，示例：访问修饰符 属性类型 属性名；**

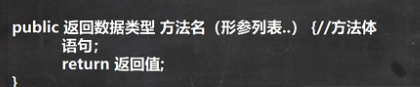
**有四种访问修饰符：public，protected，默认，private**

1. **属性的定义类型可以为任意类型，包含基本类型或引用类型**
2. **属性如果不赋值，有默认值，规则和数组一致。**

**案例：PropertiesDetail.java**

**类和对象的内存分配机制 Object03.java**

**成员方法的定义**



1. **形参列表：表示成员方法输入cal(int n), getSum(int num1 , int num2)**
2. **数据类型（返回类型）：表示成员方法输出，void 表示没有返回值**
3. **方法主体：表示为了实现某一功能代码块**
4. **return 语句不是必须的**

**注意事项和使用细节 MethodDetail.java**

**返回类型**

1. **一个方法最多有一个返回值**
2. **返回类型可以为任意类型，包含基本类型或引用类型（数组、对象）**
3. **如果方法要求返回数据类型，则方法体种最后的执行语句必须为return值；而且要求返回值类型必须和return 值的类型一致或兼容。**
4. **如果方法是void，则方法体中可以没有return语句，或者只写return；**

**方法递归调用**

**基本介绍**

**简单来说：递归就是方法自己调用自己，每次调用时传入不同的变量，递归有助于编程者解决复杂问题，同时可以让代码变得简洁。**

**方法递归重要规则**

1. **执行一个方法时，就创建一个新的受保护的独立空间(栈空间)**
2. **方法的局部变量时独立的，不会相互影响，比如n变量**
3. **如果方法中使用的时引用类型变量（比如数组），就会共享该引用类型的数据。**
4. **递归必须向退出递归的条件逼近，否则就是无限递归。**
5. **当一个方法执行完毕，同时当方法执行完毕或者返回时，该方法也就执行完毕**

**方法重载**

**基本介绍**

**java中允许同一个类中，多个同名方法的存在，但要求 形参列表不一致！**

**比如：System.out.println(); out是PrintStream类型**

**重载的好处**

1. **减轻了起名的麻烦**
2. **减轻了记名的麻烦**

**注意事项和使用细节**

1. **方法名：必须相同**
2. **参数列表：必须相同（参数类型或个数或顺序，至少有一样不同，参数名无要求）**
3. **返回类型：无要求**

**案例： OverLoadExercise.java OverLoadExercise02.java**

**可变参数**

**基本概念**

**java允许同一个类中多个同名同功能但参数个数不同的方法，封装成一个方法**

**基本语法**

**访问修饰符 返回类型 方法名（数据类型 … 形参名）**

**案例：VarParameter01.java**

**注意事项和使用细节**

**VarParameterDetail.java**

1. **可变参数的实参可以为0或者任意多个**
2. **可变参数的实参可以为数组**
3. **可变参数的本质就是数组**
4. **可变参数可以和普通类型的参数一起放在形参列表，但必须保证可变参数在最后**
5. **一个参数列表中只能出现一个可变参数**

**案例：VarParameterExercise.java**

**作用域**

**基本使用**

**面向对象中，变量作用域是非常重要知识点 案例：VarScope.java**

1. **在java编程中，主要的变量就是属性（成员变量）和局部变量**
2. **局部变量一般是指在成员方法中定义的变量**
3. **java中作用域的分类**

**全局变量：也就是属性，作用域为整个类体**

**局部变量：也就是除了属性之外的其他变量，作用域为定义它的代码块中。**

1. **全局变量可以不赋值，直接使用，因为它有默认值，局部变量必须赋值后，才能使用，因为它没有默认值。**

**注意事项和使用细节**

**VarScopeDetail.java**

1. **属性和局部变量可以重名，访问时遵循就近原则**
2. **在同一个作用域中，比如在同一个成员方法中，两个局部变量，不能重名**
3. **属性生命周期较长，伴随着对象的创建而创建，伴随着对象的死亡而死亡。局部变量，生命周期较短，伴随着它的代码块而创建，伴随着代码块的结束而死亡。即在一次方法调用过程中。**
4. **作用域范围不同**

**全局变量：可以被本类使用，或其他类使用（通过对象调用）**

**局部变量：只能在本类中对应的方法中使用**

1. **修饰符不同**

**全局变量/属性可以加修饰符**

**局部变量不可以加修饰符**

**构造方法/构造器**

**基本语法**

**[修饰符] 方法名(形参列表) {**

**方法体;**

**}**

1. **构造器的修饰符可以默认，也可以是public protected private**
2. **构造器没有返回值**
3. **方法名和类名字必须一样**
4. **参数列表 和 成员方法一样的规则**
5. **构造器的调用系统完成**

**基本介绍**

**构造方法又叫构造器（constructor），是类的一种特殊的方法，它的主要作用是完成对新对象的初始化，它有几个特点：**

1. **方法名和类名相同**
2. **没有返回值**
3. **创建对象时，系统会自动的调用该类的构造器完成对对象的初始化**

**注：**

1. **构造器没有返回值，也不能写void**
2. **构造器的名称和类Person一样**
3. **（String pName , int pAge）是构造器形参列表，规则和成员方法一样**

**注意事项和使用细节**

**ConstructorDetail.java**

1. **一个类可以定义多个不同的构造器，即构造器重载**

**比如：我们可以再给Person类定义一个构造器，用来创建对象的时候，只指定人名。不需要指定年龄。**

1. **构造器名和类名要相同**
2. **构造器没有返回值**
3. **构造器是完成对象的初始化，并不是创建对象**
4. **在创建对象时，系统自动的调用该类的构造方法**
5. **如果程序员没有定义构造器，系统会自动给类生成一个默认无参数的构造器（也叫默认构造器），比如Person() {},使用javap指令 反编译看看**
6. **一旦定义了自己的构造器，默认构造器就被覆盖了，就不能使用默认的无参构造器，除非显式的定义一下，即Person() {}**

**练习：ConstructorExercise.java**

**this关键字**

**java虚拟机会给每个对象分配this，代表当前对象。**

**this的注意事项和使用细节**

**ThisDetail.java**

1. **this关键字可以用来访问本类的属性、方法、构造器**
2. **this用于区分当前；类的属性和局部变量**
3. **访问成员方法的语法：this.方法名(参数列表);**
4. **访问构造器语法：this(参数列表)；注意只能在构造器中使用（即只能在构造器中访问另外一个构造器，必须放在第一条语句）**
5. **this不能在类定义的外部使用，只能在类定义的方法中使用**

**案例：TestPerson.java**