**Day01**

**1.Linux:**

**1)开源的操作系统、免费**

**主要是服务器端操作系统，java主要是服务器开发**

**2)与Windows目录结构的差别:**

**2.1)文件系统不同:**

**Windows:盘符 Linux:目录(只有一个根目录/)**

**2.2)外部设备映射不同:**

**Windows:盘符 Linux:挂载点**

**2.3)安全级别不同:**

**Windows:低 Linux:高(/home/soft01称为主目录)**

**3)Linux是基于命令操作的:**

**3.1)pwd:显示当前工作目录**

**3.2)ls:查看当前工作目录的内容**

**3.3)cd:改变当前工作目录**

**绝对路径:相对于根目录的位置，以/开头**

**eg: cd /home/soft01**

**相对路径:相对于当前工作目录的位置**

**eg: cd soft01/workspace**

**.:当前目录**

**..:上一级目录**

**2.Java开发环境:**

**1)java编译运行过程:常见面试题**

**编译期:.java源文件，经过编译，生成.class字节码文件**

**运行期:JVM加载.class并运行.class**

**跨平台、一次编程到处使用**

**2)名词解释:常见面试题**

**2.1)JVM:java虚拟机**

**加载.class并运行.class**

**2.2)JRE:java运行环境**

**除了包含JVM以外，还包含了运行java程序所必须的环境**

**JRE=JVM+系统类库**

**2.3)JDK:java开发工具包**

**除了包含JRE以外，还包含了开发java程序所必须的命令工具**

**JDK=JRE+编译、运行等命令工具**

**运行java程序的最小环境为JRE**

**开发java程序的最小环境为JDK**

**3)配置环境变量: 了解**

**JAVA\_HOME:指向jdk安装目录**

**CLASSPATH:表示类的搜索路径，一般简写为.**

**PATH:指向jdk下的bin目录**

**3.eclipse:**

**IBM、开源的、免费的、解压即可使用**

**Day02**

**1.变量:存数的，代词，指代的就是它所存的那个数**

**1)声明:**

**int a; //声明一个整型变量，名为a**

**int b,c,d; //声明三个整型变量，名为b,c,d**

**2)命名:**

**2.1)只能包含字母、数字、\_和$符，**

**并且不能以数字开头**

**2.2)严格区分大小写**

**2.3)不能使用关键字**

**2.4)可以中文命名，但不建议**

**建议:见名知意、驼峰命名法**

**3)初始化:第一次赋值**

**3.1)声明的同时初始化:**

**int a=250; //声明整型变量a并赋值为250**

**3.2)先声明后初始化:**

**int a; //声明整型变量a**

**a=250; //给变量a赋值给250**

**4)使用:**

**4.1)必须与数据类型匹配**

**int a=3.14; //编译错误，类型不匹配**

**4.2)对变量的操作就是对它所存的那个数的操作**

**int a=5;**

**int b=a+10; //取出a的值5，加10后，再赋值给整型变量b**

**a =a+10; //取出a的值5，加10后，再赋值给a**

**//在a本身基础之上增10**

**System.out.println("a"); //a，双引号中的原样输出**

**System.out.println(a); //输出a的值15**

**4.3)变量在用之前必须声明并初始化**

**System.out.println(m); //编译错误，m未声明**

**int m;**

**System.out.println(m); //编译错误，m未初始化**

**2.基本数据类型:**

**1)int:整型，4个字节，-21个多亿到21个多亿**

**1.1)整数直接量默认为int型，但不能超范围，**

**超范围则编译错误**

**1.2)两个整数相除，结果还是整数，小数位无条件舍弃**

**1.3)运算时超范围会发生溢出，溢出是需要避免的**

**2)long:长整型，8个字节，很大很大很大**

**2.1)long型直接量需要数字后加L或l**

**2.2)运算时为了避免溢出建议在第1个数字后加L**

**2.3)System.currentTimeMillis()**

**获取自1970.1.1零时到此时此刻的毫秒数**

**3)double:浮点型，8个字节，很大很大很大**

**3.1)浮点型直接量默认为double型，**

**若表示float型需在数字后加f或F**

**3.2)double运算时有舍入误差，所以精确运算场合不适合**

**4)boolean:布尔型，1个字节**

**4.1)只能取值为true和false**

**5)char:字符型，2个字节**

**5.1)java中的char采用Unicode编码格式，**

**每个字符(char)对应一个码(int)**

**char字符是表现形式，实质上是int码(0到65535之间)**

**5.2)字符必须放在单引号中，有且仅有一个**

**3.基本类型间的转换:**

**类型从小到大依次为:**

**byte-short-int-long-float-double byte -128---127**

**char-- short -32768---32768**

**1)两种形式:**

**1.1)自动类型转换:从小类型到大类型 long l=1024L;**

**1.2)强制类型转换:从大类型到小类型 int i=(int)l;**

**语法:(要转换成的类型)变量**

**强转有可能会溢出或精度的丢失**

**2)两点规则:**

**2.1)整数直接量可以直接赋值给byte,short,char，**

**但不能超范围**

**2.2)byte,short,char型变量参与运算时，**

**先一律转为int再运算**

**4.Scanner的用法:**

**1)在package的下面:**

**import java.util.Scanner;**

**2)在main()方法中:**

**Scanner scan = new Scanner(System.in);**

**3)在第2步之下:**

**System.out.println("请输入年龄:");**

**int age = scan.nextInt();**

**System.out.println("请输入价格:");**

**double price = scan.nextDouble();**

**Day03**

**1.运算符:**

**1)算术:+,-,\*,/,%,++,--**

**2)关系:>,<,>=,<=,==,!=**

**3)逻辑:&&,||,!**

**4)赋值:=,+=,-=,\*=,/=,%=**

**5)字符串连接:+**

**6)三目运算符:boolean?数1(T):数2(F)**

**2.分支结构:有条件的执行某语句，并非每句必走**

**1)if结构:1条路**

**2)if...else结构:2条路**

**3)if...else if结构:多条路**

**4)switch...case结构:多条路**

**优点:结构清晰、效率高**

**缺点:整数、相等**

**break:跳出switch**

**Day04**

**1.循环:反复执行一段相同或相似的代码**

**2.循环三要素:**

**1)循环变量的初始化**

**2)循环的条件(以循环变量为基础)**

**3)循环变量的改变(向着循环的结束变)**

**循环变量:在循环过程中所改变的那个量**

**3.循环结构:**

**1)while:先判断后执行，有可能一次都不执行**

**2)do...while:先执行再判断，至少执行一次**

**第1要素与第3要素相同时，首选**

**3)for:应用率最高，固定次数循环**

**4.break:跳出循环**

**continue:跳过循环体中剩余语句而进入下一次循环**

**随机数代码**

**int num=(int)(Math.*random*()\*1000+1);**

**Math.*random 0.0000000001~0.99999999999***

**Day5**

**1.三种循环结构的更佳适用情况:**

**1)while: "当..."循环**

**2)do...while: "直到..."循环**

**第1要素与第3要素相同时首选**

**3)for: 固定次数循环**

**2.嵌套循环:**

**1)循环中套循环，多行多列时使用，外层控制行，内层控制列**

**2)执行规则: 外层循环走一次，内层循环走所有次**

**3)建议:嵌套层数越少越好，能用一层就不用两层，能用两层就不用三层**

**必须用三层以上才能解决，说明设计有问题**

**4)break只能跳出一层循环**

**3.程序=算法+数据结构**

**1)算法:解决问题的步骤/流程/方法(顺序、分支、循环)**

**数据结构:将数据按照某种特定的结构来保存**

**数怎么存**

**2)设计良好的数据结构会导致好的算法**

**4.数组:**

**1)相同数据类型元素的集合**

**2)数组是一种数据类型(引用类型)**

**3)数组的定义:**

**int[] arr = new int[4];**

**4)数组的初始化:**

**int[] arr = new int[4]; //0,0,0,0**

**int[] arr = {1,3,5,7}; //1,3,5,7**

**int[] arr = new int[]{1,3,5,7}; //1,3,5,7**

**int[] arr;**

**arr = {1,3,5,7}; //编译错误，此方式必须声明同时初始化**

**arr = new int[]{1,3,5,7}; //正确**

**5)数组的访问:**

**5.1)通过数组名.length来获取数组的长度**

**int[] arr = new int[3];**

**System.out.println(arr.length); //3**

**5.2)通过下标/索引来访问数组中的元素**

**下标从0开始，最大到(数组名.length-1)**

**int[] arr = new int[3];**

**arr[0] = 100;**

**arr[1] = 200;**

**arr[2] = 300;**

**arr[3] = 400; //数组下标越界异常**

**System.out.println(arr[0]); //输出第1个元素的值**

**System.out.println(arr[arr.length-1]); //输出最后一个元素的值**

**6)数组的遍历:**

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

**for(int i=0;i<arr.length;i++){**

**arr[i] = 100;**

**}**

**for(int i=0;i<arr.length;i++){**

**System.out.println(arr[i]);**

**}**

**for(int i=arr.length-1;i>=0;i--){**

**System.out.println(arr[i]);**

**}**

**7)数组的复制:**

**7.1)System.arraycopy(a,1,a1,0,4);---效率高、灵活**

**7.2)int[] a1 = Arrays.copyOf(a,6);**

**a = Arrays.copyOf(a,a.length+1); //扩容**

**8)数组的排序:**

**8.1)Array.sort(arr); //升序排列，首选**

**8.2)冒泡工作原理:**

**8.2.1)四个数冒三轮**

**8.2.2)每一轮都是从第1个元素开始冒**

**每一次都是和它的下一个元素比**

**8.2.3)冒出来了就不带它玩了**

**冒泡算法的原理如下：**

**在冒泡排序算法中，需要重复的走访要排序的数列，一次比较两个元素，如果它们的顺序错误就把它们交换过来。走访数列的工作是重复地进行直到没有再需要交换的元素，也就是说该数列已经排序完成。**

**Day06**

**1.方法:函数、过程**

**1)作用:封装一段特定的逻辑功能**

**2)方法尽可能的独立，一个方法只做一件事**

**3)方法可以被调用多次，有效的避免了代码的重复**

**4)方法有利于团队协作开发**

**2.方法的定义:**

**修饰词 返回值类型 方法名(参数列表){**

**方法体**

**}**

**方法可以有返回值，也可以没有返回值:**

**无返回值，返回值类型设置为void**

**有返回值，返回值类型设置为具体的类型**

**方法可以有参，也可以无参，有参可以使方法更灵活**

**3.方法的调用:**

**1)无返回值方法:**

**方法名(有参传参);**

**2)有返回值方法:**

**数据类型 变量 = 方法名(有参传参);**

**4.return:**

**1)return 值; //1.结束方法的执行 2.返回结果给调用方**

**2)return; //结束方法的执行，只能用在无返回值的方法中**

**Oop day01**

**1.什么是类？什么是对象？**

**1)现实生活是由很多很多对象组成的**

**基于对象抽出了类**

**2)对象:真实存在的单个个体**

**类:类型/类别，代表一类个体**

**3)类中可以包含:**

**3.1)所有对象所共有的特征/属性-------变量**

**3.2)所有对象所共有的行为------------方法**

**4)一个类可以创建多个对象，**

**同一类型的多个对象，结构相同，数据不同**

**5)类是对象的模板，对象是类的具体的实例**

**2.如何创建类？如何创建对象？**

**3.引用类型之间画等号:指向了同一个对象**

**1)对其中一个引用的修改影响另外一个引用**

**2)eg: 房子钥匙**

**基本类型之间画等号:赋值**

**1)对其中一个变量的修改不会影响另外一个变量**

**2)eg: 身份证复印件**

**4.null:空，没有指向任何对象**

**若引用的值为null，则不能再进行任何操作了**

**若操作则空指针异常NullPointerException**

**Ooday02**

**1.方法的签名:方法名+参数列表**

**2.方法的重载(overload):**

**1)在同一个类中，方法名称相同，参数列表不同**

**2)编译器在编译时会自动根据签名来绑定调用不同的方法**

**3.构造方法:构造器，构建器，构造函数**

**1)常常用于给成员变量赋初值**

**2)与类同名，没有返回值类型**

**3)在创建(new)对象时被自动调用**

**4)若自己不写构造，则默认一个无参的空构造，**

**若自己写了构造，则不再默认提供**

**5)构造方法可以重载**

**4.this:指代当前对象，哪个对象调指的就是哪个对象**

**方法中访问成员变量之前默认有个this.**

**this用法:**

**1)this.成员变量名--------访问成员变量**

**2)this.方法名()----------调用方法**

**3)this()-----------------调用构造方法**

**5.引用类型数组:**

**1)Cell[] cells = new Cell[4];**

**cells[0] = new Cell(2,5);**

**cells[1] = new Cell(2,6);**

**cells[2] = new Cell(2,7);**

**cells[3] = new Cell(3,6);**

**2)Cell[] cells = new Cell[]{**

**new Cell(2,5),**

**new Cell(2,6),**

**new Cell(2,7),**

**new Cell(3,6)**

**};**

**3)int[][] arr = new int[3][];**

**arr[0] = new int[2];**

**arr[1] = new int[3];**

**arr[2] = new int[2];**

**4)int[][] arr = new int[3][4];**

**for(int i=0;i<arr.length;i++){**

**for(int j=0;j<arr[i].length;j++){**

**arr[i][j] = 100;**

**}**

**}**

**Oop03**

**1.内存管理:由JVM来分配管理----了解**

**1)堆:**

**1.1)存储所有new出来的对象(包括成员变量)**

**1.2)成员变量的生命周期:**

**创建对象时在堆中存在，垃圾回收器回收对象时一并消失**

**1.3)垃圾回收器(GC)，不定时到内存中清理垃圾**

**没有任何引用指向的对象就是垃圾**

**回收过程是透明的，System.gc()建议快一些回收**

**1.4)内存泄露:不再使用的内存没有被及时的回收**

**建议:不再使用的对象及时将引用设置为null**

**2)栈:**

**2.1)存储正在执行的方法中的所有局部变量**

**2.2)JVM为每一个正在调用中的方法在栈中**

**分配一块对应的栈桢，**

**栈桢中包含方法中的所有局部变量，**

**方法执行完以后，栈桢被清除，局部变量一并消失**

**2.3)局部变量生命周期:**

**调用方法时存在栈中，方法结束栈桢被清除时消失**

**3)方法区:**

**3.1)用于存储字节码文件(.class)以及方法**

**3.2)方法只有一份， 通过this来区分具体的对象**

**2.继承:**

**1)作用:代码复用，避免代码的重复**

**2)通过extends实现继承**

**3)父类:所有子类所共有的属性和行为**

**子类:子类所特有的属性和行为**

**4)子类继承父类后，子类具有: 父类的+子类的**

**5)一个父类可以有多个子类**

**一个子类只能有一个父类----单一继承**

**6)继承具有传递性**

**7)java规定:构造子类之前必须先构造父类**

**在子类构造方法中若没有通过super调用父类的构造方法，**

**则默认super()调用父类的无参构造，**

**若子类构造方法中自己调用了，则不再默认提供**

**super调父类构造方法必须位于子类构造方法的第1句**

**3.super:指代当前对象的父类对象**

**super用法:**

**super.成员变量名------访问父类的成员变量**

**super.方法名()--------调用父类的方法**

**super()---------------调用父类的构造方法**

**4.向上造型:**

**1)父类型的引用指向子类的对象 重写看对象**

**2)能点出来什么，看引用的类型 重载看引用**

**Ooday04**

**1.方法的重写(override):覆盖、重新写**

**1)发生在父子类中，方法签名相同，方法体不同**

**2)重写方法被调用时，看对象的类型**

**2.重写与重载的区别:常见面试题**

**1)重写(override):**

**1.1)发生在父子类中，方法名相同，参数列表相同，方法体不同**

**1.2)遵循"运行期"绑定，根据对象的类型调用方法**

**2)重载(overload):**

**2.1)发生在一个类中，方法名相同，参数列表不同，方法体不同**

**2.2)遵循"编译期"绑定，根据引用的类型绑定调用方法**

**3.package:**

**1)作用:避免类名的冲突**

**2)包名常常有层次结构，建议:包名所有字母都小写**

**3)类的全称: 包名.类名**

**4)同包中的类不能同名**

**import:**

**1)同包中的类可以直接访问，**

**不同包中的类不能直接访问，有如下两种方式:**

**1.1)先import声明/引入，后使用----建议**

**1.2)类的全称----太长了，不建议**

**4.访问控制修饰符:**

**1)public:公共的，任何类**

**2)private:私有的，本类**

**3)protected:受保护的，本类、子类、同包类**

**4)默认的:什么也不写，本类、同包类**

**类只能用public和默认的来修饰**

**类的成员如上4种都可以**

**5.static:静态的**

**1)静态变量:**

**1.1)由static修饰**

**1.2)属于类的，存在方法区中，只有一份**

**1.3)常常通过类名.来访问**

**1.4)何时用:所有对象数据都一样时使用**

**2)静态方法:**

**2.1)由static修饰**

**2.2)存在方法区中，只有一份**

**2.3)常常通过类名.来访问**

**2.4)没有隐式this传递，静态方法中不能直接访问实例成员**

**2.5)何时用:方法的操作仅与参数相关而与对象无关时使用**

**3)静态块:**

**3.1)由static修饰**

**3.2)类被加载期间自动执行，**

**因为类只被加载一次，所以静态块也只执行一次**

**3.3)何时用:常常用于初始化静态资源(图片、音频、视频...)**

**6.final:最终的**

**1)修饰变量:变量不能被改变**

**2)修饰方法:方法不能被重写**

**3)修改类: 类不能被继承**

**7.static final常量:**

**Oo.day05**

**1.static final常量:**

**1)必须声明的同时初始化**

**2)通过类名.来访问，不能被改变**

**3)建议:常量名所有字母都大写**

**4)编译器在编译时被自动替换为具体的值，效率高**

**2.抽象方法:**

**1)由abstract修饰**

**2)抽象方法只有方法的定义，没有具体的实现(连大括号都不写)**

**3.抽象类:**

**1)由abstract修饰**

**2)包含抽象方法的类必须是抽象类**

**不包含抽象方法的类也可以声明为抽象类---我乐意**

**3)抽象类不能被实例化**

**4)抽象类是需要被继承的，子类:**

**4.1)重写父类的所有抽象方法--建议**

**4.2)也可以声明为抽象类------不建议**

**5)抽象类的意义:**

**5.1)封装子类共有的属性和行为，被子类共用**

**5.2)为所有子类提供了公共的类型---向上造型**

**5.3)可以包含抽象方法，为所有子类提供一个统一的入口**

**4.接口:**

**1)是一个标准、规范**

**遵守这个标准、规范，就能够去干某件事---API之后**

**2)由interface定义**

**3)只能包含常量和抽象方法**

**4)接口不能被实例化**

**5)接口是需要被实现/继承的，实现类/子类:**

**必须重写接口中的所有抽象方法**

**6)一个类可以实现多个接口，用逗号隔开**

**若既继承又实现时，应先继承后实现**

**7)接口可以继承接口**

**Ooday06**

**1.多态:**

**1)多态的意义:**

**1.1)同一类型的引用指向不同的对象时，有不同的实现**

**----行为的多态: cut(), run()**

**1.2)同一个对象被造型成不同的类型时，有不同的功能**

**----对象的多态: 我，你，水**

**2)向上造型:**

**2.1)父类的引用指向子类的对象**

**2.2)能造型成的类型包括:**

**父类+所实现的接口**

**2.3)能点出来什么，看引用的类型**

**3)强制类型转换，成功的有条件:**

**3.1)引用所指向的对象，就是该类型**

**3.2)引用所指向的对象，实现了该接口**

**4)若不符合如上两个条件，则发生ClassCastException类型转换异常，**

**强转之前建议通过instanceof判断引用指向的对象是否是该类型**

**2.成员内部类: 应用率比较低**

**1)类中套类，外面的称为Outer外部类，里面的称为Inner内部类**

**2)内部类通常只服务于外部类，对外不具备可见性**

**3)内部类对象通常在外部类中创建**

**4)内部类中可以直接访问外部类的成员(包括私有的)**

**内部类隐式的引用指向了创建它的外部类对象**

**3.匿名内部类:较常用**

**1)想创建一个类的对象，并且对象只创建一次，**

**该类不必命名，称之为匿名内部类**

**前提: 匿名内部类一定得是子类或实现类**

**2)匿名内部类中想访问外部的变量，该变量必须是final的**

**4.面向对象三大特征:**

**1)封装:**

**1.1)类:封装的是对象的属性和行为**

**1.2)方法:封装的是具体的逻辑功能**

**1.3)访问控制修饰符:封装的访问的权限**

**2)继承:**

**2.1)代码的复用**

**2.2)父类/基类:共有的属性和行为**

**子类/派生类:特有的属性和行为**

**2.3)子类继承父类后，子类具有:父+子**

**2.4)单一继承、传递性**

**3)多态:**

**3.1)意义: 行为的多态、对象的多态**

**3.2)向上造型、强制类型转换、instanceof**

**3.3)多态的表现形式: 重写、重载**