每个Java初学者都应该搞懂的六个问题 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/253/2021_2022__E6_AF_8F_E4_B8_AAJava_c104_253555.htm 问题一：我声明了什么！String s = "Hello world!". 许多人都做过这样的事情，但是，我们到底声明了什么？回答通常是：一个String，内容是“ Hello world! ”。这样模糊的回答通常是概念不清的根源。如果要准确的回答，一半的人大概会回答错误。这个语句声明的是一个指向对象的引用，名为“ s ”，可以指向类型为String的任何对象，目前指向"Hello world!"这个String类型的对象。这就是真正发生的事情。我们并没有声明一个String对象，我们只是声明了一个只能指向String对象的引用变量。所以，如果在刚才那句语句后面，如果再运行一句：String string = s. 我们是声明了另外一个只能指向String对象的引用，名为string，并没有第二个对象产生，string还是指向原来那个对象，也就是，和s指向同一个对象。 问题二："=="和equals方法究竟有什么区别？ ==操作符专门用来比较变量的值是否相等。比较好理解的一点是：int a=10.int b=10. 则a==b将是true。 但不好理解的地方是：String a=new String("foo").String b=new String("foo"). 则a==b将返回false。 根据前一帖说过，对象变量其实是一个引用，它们的值是指向对象所在的内存地址，而不是对象本身。a和b都使用了new操作符，意味着将在内存中产生两个内容为"foo"的字符串，既然是“ 两个 ”，它们自然位于不同的内存地址。a和b的值其实是两个不同的内存地址的值，所以使用"=="操作符，结果会是false。诚然，a和b所指的对象，它们的内容都是"foo"，应该是“ 相等 ”，但

是==操作符并不涉及到对象内容的比较。 对象内容的比较，正是equals方法做的事。 看一下Object对象的equals方法是如何实现的：boolean equals(Object o){return this==o.} Object对象默认使用了==操作符。所以如果你自创的类没有覆盖equals方法，那你的类使用equals和使用==会得到同样的结果。同样也可以看出，Object的equals方法没有达到equals方法应该达到的目标：比较两个对象内容是否相等。因为答案应该由类的创建者决定，所以Object把这个任务留给了类的创建者。 看一下一个极端的类：Class Monster{private String content....boolean equals(Object another){ return true.}} 覆盖了equals方法。这个实现会导致无论Monster实例内容如何，它们之间的比较永远返回true。 所以当你是用equals方法判断对象的内容是否相等，请不要想当然。因为可能你认为相等，而这个类的作者不这样认为，而类的equals方法的实现是由他掌握的。如果你需要使用equals方法，或者使用任何基于散列码的集合（HashSet,HashMap,HashTable），请察看一下java doc以确认这个类的equals逻辑是如何实现的。 问题五：到底要怎么样初始化！ 本问题讨论变量的初始化，所以先来看一下Java中有哪些种类的变量。1. 类的属性，或者叫值域2. 方法里的局部变量3. 方法的参数 对于第一种变量，Java虚拟机会自动进行初始化。如果给出了初始值，则初始化为该初始值。如果没有给出，则把它初始化为该类型变量的默认初始值。int类型变量默认初始值为0float类型变量默认初始值为0.0fdouble类型变量默认初始值为0.0boolean类型变量默认初始值为falsechar类型变量默认初始值为0(ASCII码)long类型变量默认初始值为0 所有对象引用类型变量默认初始值为null，即不指向任何

对象。注意数组本身也是对象，所以没有初始化的数组引用在自动初始化后其值也是null。 对于两种不同的类属性，static属性与instance属性，初始化的时机是不同的。instance属性在创建实例的时候初始化，static属性在类加载，也就是第一次用到这个类的时候初始化，对于后来的实例的创建，不再次进行初始化。这个问题会在以后的系列中进行详细讨论。 对于第二种变量，必须明确地进行初始化。如果再没有初始化之前就试图使用它，编译器会抗议。如果初始化的语句在try块中或if块中，也必须要让它在第一次使用前一定能够得到赋值。也就是说，把初始化语句放在只有if块的条件判断语句中编译器也会抗议，因为执行的时候可能不符合if后面的判断条件，如此一来初始化语句就不会被执行了，这就违反了局部变量使用前必须初始化的规定。但如果在else块中也有初始化语句，就可以通过编译，因为无论如何，总有至少一条初始化语句会被执行，不会发生使用前未被初始化的事情。对于try-catch也是一样，如果只有在try块里才有初始化语句，编译部通过。如果在catch或finally里也有，则可以通过编译。总之，要保证局部变量在使用之前一定被初始化了。所以，一个好的做法是在声明他们的时候就初始化他们，如果不知道要出事化成什么值好，就用上面的默认值吧！ 其实第三种变量和第二种本质上是一样的，都是方法中的局部变量。只不过作为参数，肯定是被初始化过的，传入的值就是初始值，所以不需要初始化。 问题六：instanceof是什么东东？ instanceof是Java的一个二元操作符，和==，>，String s = "I AM an Object!".boolean isObject = s instanceof Object. 我们声明了一个String对象引用，指向一个String对象，然后用instancof

来测试它所指向的对象是否是Object类的一个实例，显然，这是真的，所以返回true，也就是isObject的值为True。
instanceof有一些用处。比如我们写了一个处理账单的系统，其中有这样三个类：public class Bill {//省略细节}public class PhoneBill extends Bill {//省略细节}public class GasBill extends Bill {//省略细节} 在处理程序里有一个方法，接受一个Bill类型的对象，计算金额。假设两种账单计算方法不同，而传入的Bill对象可能是两种中的任何一种，所以要用instanceof来判断：public double calculate(Bill bill) {if (bill instanceof PhoneBill) {//计算电话账单}if (bill instanceof GasBill) {//计算燃气账单}...} 这样就可以用一个方法处理两种子类。 然而，这种做法通常被认为是没有好好利用面向对象中的多态性。其实上面的功能要求用方法重载完全可以实现，这是面向对象变成应有的做法，避免回到结构化编程模式。只要提供两个名字和返回值都相同，接受参数类型不同的方法就可以了：public double calculate(PhoneBill bill) {//计算电话账单}public double calculate(GasBill bill) {//计算燃气账单} 所以，使用instanceof在绝大多数情况下并不是推荐的做法，应当好好利用多态。