java认证:Java编译器中对String对象的优化Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/582/2021_2022_java_E8_AE_A4_E8_AF_81_c104_582212.htm 首先把问题摆出来，先看这个代码： String a = "ab". String b = "a" "b". System.out.println((a == b)). 打印结果会是什么？类似这样的问题，有人考过我，我也拿来考过别人(蛮好玩的，大家也可以拿来问人玩)，一般答案会是以下几种： 1、true "a" "b" 的结果就是“ ab ”，这样a，b都是“ ab ”了，内容一样所以“ 相等 ”，结果true。 一般Java新人如是答。 2、false "a" "a"会生成新的对象“ aa ”，但是这个对象和String a = "ab".不同，(a == b)是比较对象引用，因此不相等，结果false。 对Java的String有一定了解的通常这样回答。 3、true String a = "ab".创建了新的对象“ ab ”；再执行String b = "a" "b".结果b="ab",这里没有创建新的对象，而是从JVM字符串常量池中获取之前已经存在的“ ab ”对象。因此a，b具有对同一个string对象的引用，两个引用相等，结果true。 能回答出这个答案的，基本已经是高手了，对Java中的string机制比较了解。 很遗憾，这个答案是不够准确的。或者说，根本没有运行时计算b = "a" "b".这个操作。实际上运行时只有String b = "ab".。 3的观点适合解释以下情况： String a = "ab". String b = "ab". System.out.println((a == b)). 如果String b = "a" "b".是在运行期执行，则3的观点是无法解释的。运行期的两个string相加，会产生新的对象的。(本文后面对此有解释) 4、true 下面是我的回答：编译优化 3的处理方式 = 最后的true String b = "a" "b".编译器将这个"a" "b"作为常量表达式，在编译时进行优化，直接取结果"ab"，这样这个问题退化。 String a =

"ab". String b = "ab". System.out.println((a == b)). 然后根据3的解释，得到结果true。 这里有一个疑问就是String不是基本类型，像 int secondsOfDay = 24 * 60 * 60. 这样的表达式是常量表达式,编译器在编译时直接计算容易理解，而"a" "b" 这样的表达式，string是对象不是基本类型,编译器会把它当成常量表达式来优化吗? 下面简单证明我的推断，首先编译这个类： public class Test { private String a = "aa". } 复制class文件备用，然后修改为： public class Test { private String a = "a" "a". } 再次编译，用ue之类的文本编辑器打开，察看二进制内容，可以发现，两个class文件完全一致，连一个字节都不差。 ok，真相大白了。根本不存在运行期的处理String b = "a" "b".这样的代码的问题，编译时就直接优化掉了。 下面进一步探讨，什么样的string 表达式会被编译器当成常量表达式？ String b = "a" "b". 这个String String被正式是ok的，那么string 基本类型呢？ String a = "a1". String b = "a" 1. System.out.println((a == b)). //result = trueString a = "atrue". String b = "a" true. System.out.println((a == b)). //result = trueString a = "a3.4". String b = "a" 3.4. System.out.println((a == b)). //result = true 可见编译器对string 基本类型是当成常量表达式直接求值来优化的。 再注意看这里的string都是"**"这样的，我们换成变量来试试： String a = "ab". String bb = "b". String b = "a" bb. System.out.println((a == b)). //result = false 这个好理解，"a" bb中的bb是变量，不能进行优化。这里很很好的解释了为什么3的观点不正确，如果String String的操作是在运行时进行的，则会产生新的对象，而不是直接从jvm的string池中获取。 再修改一下，把bb作为常量变量： String a = "ab". final String bb = "b". String b = "a" bb.

System.out.println((a == b)). //result = true 竟然又是true，编译器的优化好厉害啊！呵呵！考虑下面这种情况： String a = "ab". final String bb = getBB(). String b = "a" bb. System.out.println((a == b)). //result = false private static String getBB() { return "b". } 看来Java(包括编译器和jvm)对string的优化，真的是到了极点了，string这个所谓的“ 对象 ”，完全不可以看成一般的对象，Java对string的处理近乎于基本类型，最大限度的优化了几乎能优化的地方。 另外感叹一下，string的号处理，算是Java语言里面唯一的一个“ 运算符重载 ”(接触过c 的人对这个不会陌生)吧？