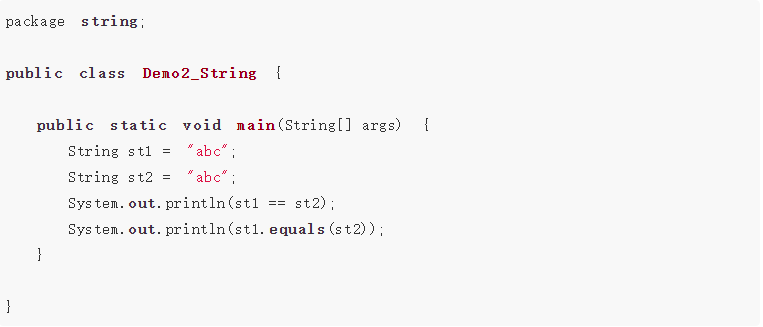
<https://mp.weixin.qq.com/s/VRMU2jtjNn1AD7JLPkBFDA>

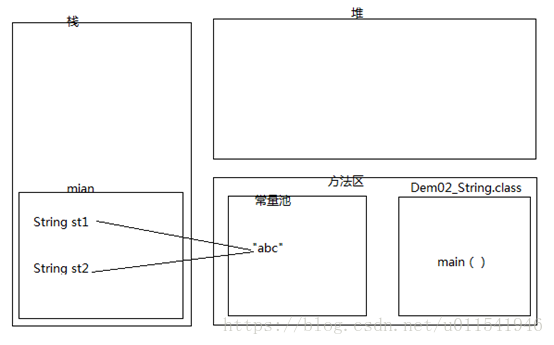
# String 相关面试题

1. **判定定义为String类型的st1和st2是否相等，为什么？**



输出结果：第一行：true第二行：true。分析：先看第一个打印语句，在Java中==这个符号是比较运算符，它可以基本数据类型和引用数据类型是否相等，如果是基本数据类型，==比较的是值是否相等，如果是引用数据类型，==比较的是两个对象的内存地址是否相等。字符串不属于8中基本数据类型，字符串对象属于引用数据类型，在上面把“abc”同时赋值给了st1和st2两个字符串对象，指向的都是同一个地址，所以第一个打印语句中的==比较输出结果是 true。

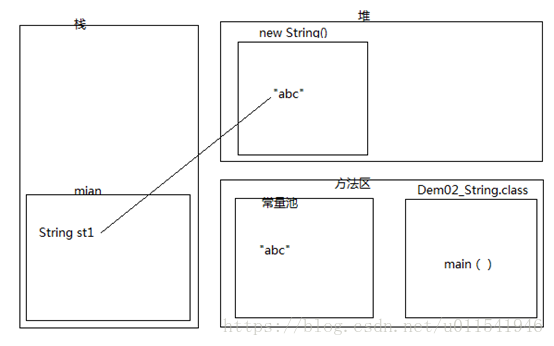
然后我们看第二个打印语句中的equals的比较，我们知道，equals是Object这个父类的方法，在String类中重写了这个equals方法，在JDK API 1.6文档中找到String类下的equals方法，点击进去可以看大这么一句话“将此字符串与指定的对象比较。当且仅当该参数不为null，并且是与此对象表示相同字符序列的 String 对象时，结果才为 true。” 注意这个相同字符序列，在后面介绍的比较两个数组，列表，字典是否相等，都是这个逻辑去写代码实现。由于st1和st2的值都是“abc”，两者指向同一个对象，当前字符序列相同，所以第二行打印结果也为true。下面我们来画一个内存图来表示上面的代码，看起来更加有说服力。



内存过程大致如下：1）运行先编译，然后当前类Demo2\_String.class文件加载进入内存的方法区2）第二步，main方法压入栈内存3）常量池创建一个“abc”对象，产生一个内存地址4）然后把“abc”内存地址赋值给main方法里的成员变量st1，这个时候st1根据内存地址，指向了常量池中的“abc”。5）前面一篇提到，常量池有这个特点，如果发现已经存在，就不在创建重复的对象6）运行到代码 Stringst2 =”abc”, 由于常量池存在“abc”，所以不会再创建，直接把“abc”内存地址赋值给了st27）最后st1和st2都指向了内存中同一个地址，所以两者是完全相同的。

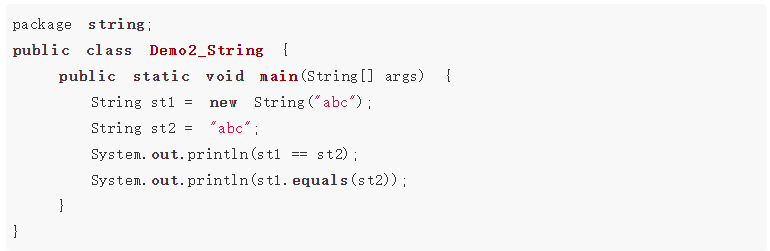
1. **下面这句话在内存中创建了几个对象**

答案是：在内存中创建两个对象，一个在堆内存，一个在常量池，堆内存对象是常量池对象的一个拷贝副本。分析：我们下面直接来一个内存图。

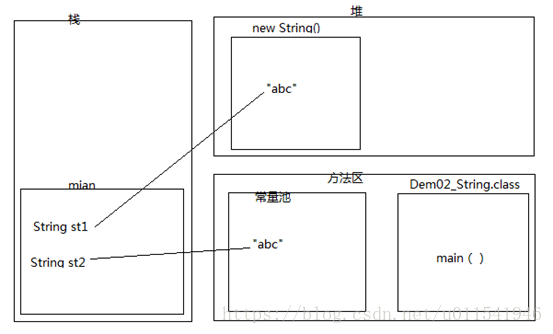


当我们看到了new这个关键字，就要想到，new出来的对象都是存储在堆内存。然后我们来解释堆中对象为什么是常量池的对象的拷贝副本。“abc”属于字符串，字符串属于常量，所以应该在常量池中创建，所以第一个创建的对象就是在常量池里的“abc”。第二个对象在堆内存为啥是一个拷贝的副本呢，这个就需要在JDK API 1.6找到String(String original)这个构造方法的注释：初始化一个新创建的 String 对象，使其表示一个与参数相同的字符序列；换句话说，新创建的字符串是该参数字符串的副本。所以，答案就出来了，两个对象。

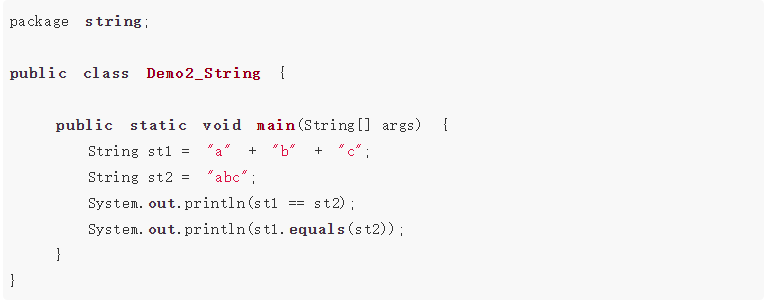
1. **判定以下定义为String类型的st1和st2是否相等**



答案：false 和 true由于有前面两道提内存分析的经验和理论，所以，我能快速得出上面的答案。==比较的st1和st2对象的内存地址，由于st1指向的是堆内存的地址，st2看到“abc”已经在常量池存在，就不会再新建，所以st2指向了常量池的内存地址，所以==判断结果输出false，两者不相等。第二个equals比较，比较是两个字符串序列是否相等，由于就一个“abc”，所以完全相等。内存图如下

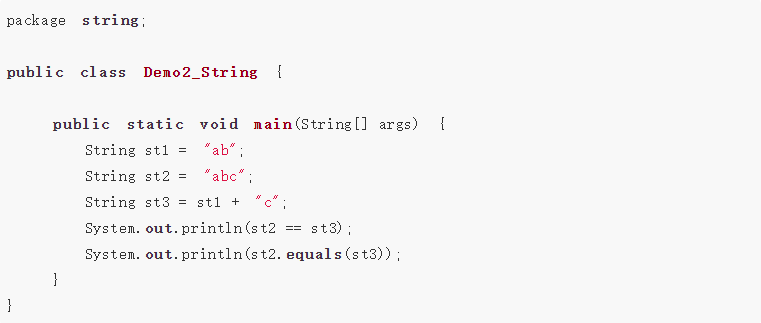


1. **判定以下定义为String类型的st1和st2是否相等**

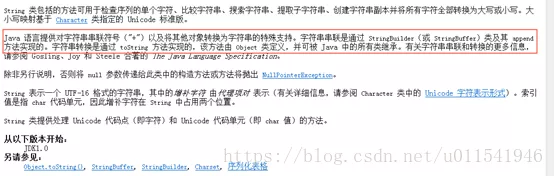


答案是：true 和 true分析：“a”,”b”,”c”三个本来就是字符串常量，进行+符号拼接之后变成了“abc”，“abc”本身就是字符串常量（Java中有常量优化机制），所以常量池立马会创建一个“abc”的字符串常量对象，在进行st2=”abc”,这个时候，常量池存在“abc”，所以不再创建。所以，不管比较内存地址还是比较字符串序列，都相等。

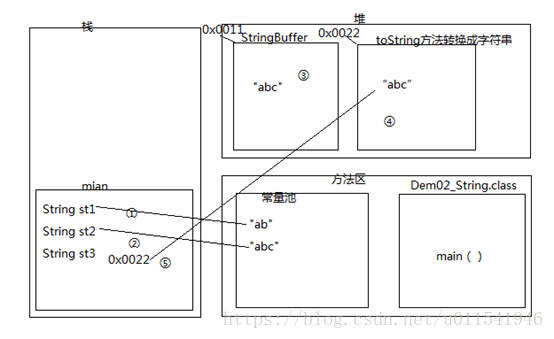
1. **判断以下st2和st3是否相等**



答案：false 和 true分析：上面的答案第一个是false，第二个是true，第二个是true我们很好理解，因为比较一个是“abc”，另外一个是拼接得到的“abc”，所以equals比较，这个是输出true，我们很好理解。那么第一个判断为什么是false，我们很疑惑。同样，下面我们用API的注释说明和内存图来解释这个为什么不相等。首先，打开JDK API 1.6中String的介绍，找到下面图片这句话。



关键点就在红圈这句话，我们知道任何数据和字符串进行加号（+）运算，最终得到是一个拼接的新的字符串。上面注释说明了这个拼接的原理是由StringBuilder或者StringBuffer类和里面的append方法实现拼接，然后调用toString（）把拼接的对象转换成字符串对象，最后把得到字符串对象的地址赋值给变量。结合这个理解，我们下面画一个内存图来分析。



大致内存过程  
1）常量池创建“ab”对象，并赋值给st1，所以st1指向了“ab”2）常量池创建“abc”对象，并赋值给st2，所以st2指向了“abc”3）由于这里走的+的拼接方法，所以第三步是使用StringBuffer类的append方法，得到了“abc”，这个时候内存0x0011表示的是一个StringBuffer对象，注意不是String对象。4）调用了Object的toString方法把StringBuffer对象装换成了String对象。5）把String对象（0x0022）赋值给st3所以，st3和st2进行==判断结果是不相等，因为两个对象内存地址不同。

**总结：**

这篇的面试题，完全就是要求掌握JDK API中一些注解和原理，以及内存图分析，才能得到正确的结果。