Java代码

1. String s = **new** String("abc");
2. String s1 = "abc";
3. String s2 = **new** String("abc");
5. System.out.println(s == s1);   **false**
6. System.out.println(s == s2);   **false**
7. System.out.println(s1 == s2);  **false**   
   String s = **new** String("abc");  执行后内存中有两个对象，而不是一个。

一个由new String("abc")中的"abc"在String Pool里生成一个值为"abc"的对象；

第二个由new在堆里产生一个值为"abc"的对象，该对象完全是String Pool里的"abc"的一个拷贝。变量s最后指向堆中产生的"abc"对象;

String s1 = "abc";   执行时，s1先去String Pool找是否有值为"abc"的对象，很显然在上一步中java已经在String Pool里生成一个"abc"对象了，所以s1直接指向String Pool中的这个"abc"；

String s2 = **new** String("abc");又有一个new，在java中凡遇到new时，都会在堆里产生一个新的对象。因此，该句执行后堆里又多了一个"abc"对象，这与执行第一句后生成的"abc"是不同的两个对象，s2最后指向这个新生成的对象。   
因此，执行后面的打印语句的结果是三个**false**   
  
问题2:   
Java代码

1. String s = **new** String("abc");
2. String s1 = "abc";
3. String s2 = **new** String("abc");
4. System.out.println(s == s.intern());   **false**
5. System.out.println(s1 == s1.intern()); **true**
6. System.out.println(s1.intern() == s2.intern()); **true**

设 s为String类型的变量，当执行s.intern()时，java先在String Pool里找与字符串变量s相等(用equals()方法)的字符串，若有则将其引用返回；**若没有则在String Pool里创建一个与s的值相等的字符串对象，并将其引用返回。从中我们可以总结出intern()方法无论如何都将返回String Pool里的字符串对象的引用。**  
  
PS:设s和t为两个字符串变量，若有s.equals(t)，必有s.intern() == t.intern()；   
PS:**"=="永远比较的是两边对象的地址是否相等**。   
  
问题3：   
Java代码

1. String hello = "hello";
2. String hel = "hel";
3. String lo = "lo";
4. System.out.println(hello == "hel" + "lo");
5. System.out.println(hello == "hel" + lo);

String hello = "hello";String hel = "hel";String lo = "lo";System.out.println(hello == "hel" + "lo");System.out.println(hello == "hel" + lo);  
请问以上程序执行结果是什么？   
  
前三句在String Pool里分别产生“hello”、“hel”、“lo”三个常量字符串对象   
当做第一个加法连接时，+号两边都是常量字符串，java就会将两者拼起来后到String Pool里找与之相等（用equals）的字符串，若存在则将其地址返回；不存在则在String Pool里新建一个常量对象，其值等于拼接后的字符串，并将其地址返回。   
第二个+号两边有一个是变量，此时，java会在堆里新建一个对象，其值是两字符串拼接后的值，此时返回的地址是堆中新对象的地址。   
所以，第一句做+连接后返回String Pool中“hello”的地址，显然与变量hello的地址相等；   
第二句返回的是堆中地址，显然与变量hello的地址不等；