**[java中的String类常量池详解](http://www.cnblogs.com/brokencode/archive/2011/08/24/2151450.html)**

从一个博客上看到的6个题，先看看吧，如果都会了，这部分的知识就掌握的不错啦！输出结果在代码注释后面：  
  
  
test1：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test1 {  
  
 /\*\*  
 \* @param args  
 \*/  
 public static void main(String[] args){  
 String a = "a1";  
 String b = "a"+ 1;  
 System.out.println(a==b);  
 }//true  
  
}

[复制代码](javascript:void(0);)

test2：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test2 {  
  
 /\*\*  
 \* @param args  
 \*/  
 public static void main(String[] args){  
 String a = "ab";  
 String bb = "b";  
 String b = "a"+ bb; //编译器不能确定为常量  
 System.out.println(a==b);  
 }//false  
  
}

[复制代码](javascript:void(0);)

test3：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test3 {  
  
 /\*\*  
 \* @param args  
 \*/  
 public static void main(String[] args){  
 String a = "ab";  
 final String bb = "b";  
 String b = "a"+ bb; //bb加final后是常量，可以在编译器确定b  
 System.out.println(a==b);  
 }//true  
  
}

[复制代码](javascript:void(0);)

test4：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test4 {  
  
 /\*\*  
 \* @param args  
 \*/  
 public static void main(String[] args){  
 String a = "ab";  
 final String bb = getBB();  
 String b = "a"+ bb;//bb是通过函数返回的，虽然知道它是final的，但不知道具体是啥，要到运行期才知道bb的值  
 System.out.println(a==b);  
 }//false  
 private static String getBB(){ return "b"; }  
  
}

[复制代码](javascript:void(0);)

test5：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test5 {  
  
 /\*\*  
 \* @param args  
 \*/  
 private static String a = "ab";  
 public static void main(String[] args){  
 String s1 = "a";  
 String s2 = "b";  
 String s = s1 + s2;//+的用法  
 System.out.println(s == a);  
 System.out.println(s.intern() == a);//intern的含义  
 }//flase true  
  
}

[复制代码](javascript:void(0);)

test6：

[复制代码](javascript:void(0);)

package StringTest;  
  
public class test6 {  
  
 /\*\*  
 \* @param args  
 \*/  
 private static String a = new String("ab");  
 public static void main(String[] args){  
 String s1 = "a";  
 String s2 = "b";  
 String s = s1 + s2;  
 System.out.println(s == a);  
 System.out.println(s.intern() == a);  
 System.out.println(s.intern() == a.intern());  
 }//flase false true  
}

[复制代码](javascript:void(0);)

-------------------------------------------------------------------------------------------------------------------------------------------------  
  
  
String常量池详解：  
  
　　1.String使用private final char value[]来实现字符串的存储,也就是说String对象创建之后,就不能再修改此对象中存储的字符串内容,就是因为如此,才说String类型是不 可变的(immutable)。String类有一个特殊的创建方法,就是使用""双引号来创建.例如new String("i am")实际创建了2个  
　　String对象,一个是"i am"通过""双引号创建的,另一个是通过new创建的.只不过他们创建的时期不同,  
　　一个是编译期,一个是运行期!java对String类型重载了+操作符,可以直接使用+对两个字符串进行连接。运行期调用String类的intern()方法可以向String Pool中动态添加对象。  
　　  
　　例1  
　　String s1 = "sss111";  
　　//此语句同上  
　　String s2 = "sss111";  
　　System.out.println(s1 == s2); //结果为true  
　　例2  
　　String s1 = new String("sss111");  
　　String s2 = "sss111";  
　　System.out.println(s1 == s2); //结果为false  
　　例3  
　　String s1 = new String("sss111");  
　　s1 = s1.intern();  
　　String s2 = "sss111";  
　　System.out.println(s1 == s2);//结果为true  
　　例4  
　　String s1 = new String("111");  
　　String s2 = "sss111";  
　　String s3 = "sss" + "111";  
　　String s4 = "sss" + s1;  
　　System.out.println(s2 == s3); //true  
　　System.out.println(s2 == s4); //false  
　　System.out.println(s2 == s4.intern()); //true  
　　  
  
　　**结果上面分析,总结如下:**  
  
　 　**1.单独使用""引号创建的字符串都是常量,编译期就已经确定存储到String Pool中；**  
  
**2，使用new String("")创建的对象会存储到heap中,是运行期新创建的；**  
  
**3，使用只包含常量的字符串连接符如"aa" + "aa"创建的也是常量,编译期就能确定,已经确定存储到String Pool中；**  
  
**4，使用包含变量的字符串连接符如"aa" + s1创建的对象是运行期才创建的,存储在heap中；**  
  
　　还有几个经常考的面试题:  
　　  
　　String s1 = new String("s1") ;  
　　String s2 = new String("s1") ;  
　　上面创建了几个String对象?  
　　答案:3个 ,编译期Constant Pool中创建1个,运行期heap中创建2个.（用new创建的每new一次就在堆上创建一个对象，用引号创建的如果在常量池中已有就直接指向，不用创建）  
  
　　String s1 = "s1";  
　　String s2 = s1;  
　　s2 = "s2";  
　　s1指向的对象中的字符串是什么?  
　　答案: "s1"。（永远不要忘了String不可变的，s2 = "s2";实际上s2的指向就变了，因为你不可以去改变一个String，）  
  
--------------------------------------------------------------------------------------------------------------------------------------------------  
  
String是一个特殊的包装类数据。可以用：   
String str = new String("abc");   
String str = "abc";   
两种的形式来创建，第一种是用new()来新建对象的，它会在存放于堆中。每调用一次就会创建一个新的对象。   
而第二种是先在栈中创建一个对String类的对象引用变量str，然后通过符号引用去字符串常量池里找有没有"abc",如果没有，则将"abc"存放进字符串常量池，并令str指向”abc”，如果已经有”abc” 则直接令str指向“abc”。   
  
比较类里面的数值是否相等时，用equals()方法；当测试两个包装类的引用是否指向同一个对象时，用==，下面用例子说明上面的理论。   
String str1 = "abc";   
String str2 = "abc";   
System.out.println(str1==str2); //true   
可以看出str1和str2是指向同一个对象的。   
  
String str1 =new String ("abc");   
String str2 =new String ("abc");   
System.out.println(str1==str2); // false   
用new的方式是生成不同的对象。每一次生成一个。   
  
因 此用第二种方式创建多个”abc”字符串,在内存中其实只存在一个对象而已. 这种写法有利与节省内存空间. 同时它可以在一定程度上提高程序的运行速度，因为JVM会自动根据栈中数据的实际情况来决定是否有必要创建新对象。而对于String str = new String("abc")；的代码，则一概在堆中创建新对象，而不管其字符串值是否相等，是否有必要创建新对象，从而加重了程序的负担。   
  
另 一方面, 要注意: 我们在使用诸如String str = "abc"；的格式定义类时，总是想当然地认为，创建了String类的对象str。担心陷阱！对象可能并没有被创建！而可能只是指向一个先前已经创建的 对象。只有通过new()方法才能保证每次都创建一个新的对象。   
**由于String类的immutable性质，当String变量需要经常变换其值时，应该考虑使用StringBuffer类，以提高程序效率。**   
1. 首先String不属于8种基本数据类型，String是一个对象。   
因为对象的默认值是null，所以String的默认值也是null；但它又是一种特殊的对象，有其它对象没有的一些特性。   
  
2. new String()和new String(”")都是申明一个新的空字符串，是空串不是null；   
  
3. String str=”kvill”；String str=new String (”kvill”)的区别  
  
看例1：   
  
String s0="kvill";   
String s1="kvill";   
String s2="kv" + "ill";   
System.out.println( s0==s1 );   
System.out.println( s0==s2 );   
结果为：   
true   
true   
  
首先，我们要知结果为道Java会确保一个字符串常量只有一个拷贝。   
因 为例子中的s0和s1中的”kvill”都是字符串常量，它们在编译期就被确定了，所以s0==s1为true；而”kv”和”ill”也都是字符串常 量，当一个字符串由多个字符串常量连接而成时，它自己肯定也是字符串常量，所以s2也同样在编译期就被解析为一个字符串常量，所以s2也是常量池中” kvill”的一个引用。所以我们得出s0==s1==s2；用new String() 创建的字符串不是常量，不能在编译期就确定，所以new String() 创建的字符串不放入常量池中，它们有自己的地址空间。   
  
看例2：   
String s0="kvill";   
String s1=new String("kvill");   
String s2="kv" + new String("ill");   
System.out.println( s0==s1 );   
System.out.println( s0==s2 );   
System.out.println( s1==s2 );   
结果为：   
false   
false   
false   
  
例 2中s0还是常量池中"kvill”的应用，s1因为无法在编译期确定，所以是运行时创建的新对象”kvill”的引用，s2因为有后半部分 new String(”ill”)所以也无法在编译期确定，所以也是一个新创建对象”kvill”的应用;明白了这些也就知道为何得出此结果了。   
  
4. String.intern()：   
再补充介绍一点：存在于.class文件中的常量池，在运行期被JVM装载，并且可以扩充。String的intern()方法就是扩充常量池的 一个方法；当一个String实例str调用intern()方法时，Java查找常量池中是否有相同Unicode的字符串常量，如果有，则返回其的引用，如果没有，则在常量池中增加一个Unicode等于str的字符串并返回它的引用；看例3就清楚了   
  
例3：   
String s0= "kvill";   
String s1=new String("kvill");   
String s2=new String("kvill");   
System.out.println( s0==s1 );   
System.out.println( "\*\*\*\*\*\*\*\*\*\*" );   
s1.intern();   
s2=s2.intern(); //把常量池中"kvill"的引用赋给s2   
System.out.println( s0==s1);   
System.out.println( s0==s1.intern() );   
System.out.println( s0==s2 );   
结果为：   
false   
\*\*\*\*\*\*\*\*\*\*   
false //虽然执行了s1.intern(),但它的返回值没有赋给s1   
true //说明s1.intern()返回的是常量池中"kvill"的引用   
true   
  
最 后我再破除一个错误的理解：有人说，“使用 String.intern() 方法则可以将一个 String 类的保存到一个全局 String 表中 ，如果具有相同值的 Unicode 字符串已经在这个表中，那么该方法返回表中已有字符串的地址，如果在表中没有相同值的字符串，则将自己的地址注册到表中”如果我把他说的这个全局的 String 表理解为常量池的话，他的最后一句话，”如果在表中没有相同值的字符串，则将自己的地址注册到表中”是错的：   
  
看例4：   
String s1=new String("kvill");   
String s2=s1.intern();   
System.out.println( s1==s1.intern() );   
System.out.println( s1+" "+s2 );   
System.out.println( s2==s1.intern() );   
结果：   
false   
kvill kvill   
true   
  
在这个类中我们没有声名一个”kvill”常量，所以常量池中一开始是没有”kvill”的，当我们调用s1.intern()后就在常量池中新添加了一个”kvill”常量，原来的不在常量池中的”kvill”仍然存在，也就不是“将自己的地址注册到常量池中”了。   
s1==s1.intern()为false说明原来的”kvill”仍然存在；s2现在为常量池中”kvill”的地址，所以有s2==s1.intern()为true。 