### 4.1String类和常量池

1. String对象的两种创建方式：

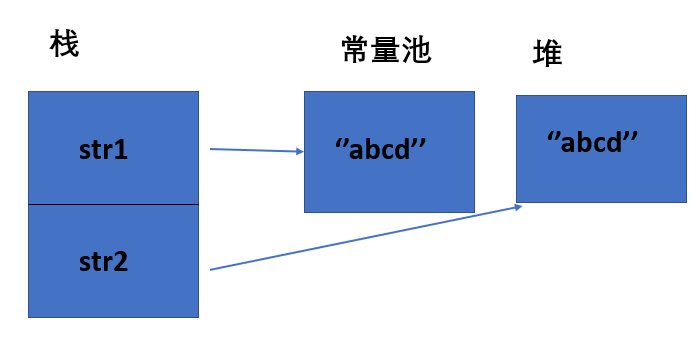
String str1 = “abcd”;

String str2 = new String(“abcd”);

System.out.println(“str1==str2”); //false

这两种不同的创建方法是有差别的，第一种方式是在常量池中拿对象，第二种方式是直接在堆内存空间创建一个新的对象。

注意：只要使用new方法，便需要创建新的对象。



1. String类型的常量池比较特殊。它的主要使用方法有两种：
   1. 直接使用双引号声明出来的String对象会直接存储在常量池中。
   2. 如果不使用双引号声明的String对象，可以使用String提供的intern方法。String.intern()是一个Native方法，它的作用是：如果运行时常量池中已经包含一个等于此String对象内容的字符串，则返回常量池中该字符串的引用；如果没有，则在常量池中创建与此String内容相同的字符串，并返回常量池中创建的字符串的引用。

String s1 = new String("计算机");

String s2 = s1.intern();

String s3 = "计算机";

System.out.println(s2);//计算机

System.out.println(s1 == s2);//false，因为一个是堆内存中的String对象一个是常量池中的String对象，

System.out.println(s3 == s2);//true，因为两个都是常量池中的String对象

1. String字符串拼接

String str1 = "str";

String str2 = "ing";

String str3 = "str" + "ing";//常量池中的对象

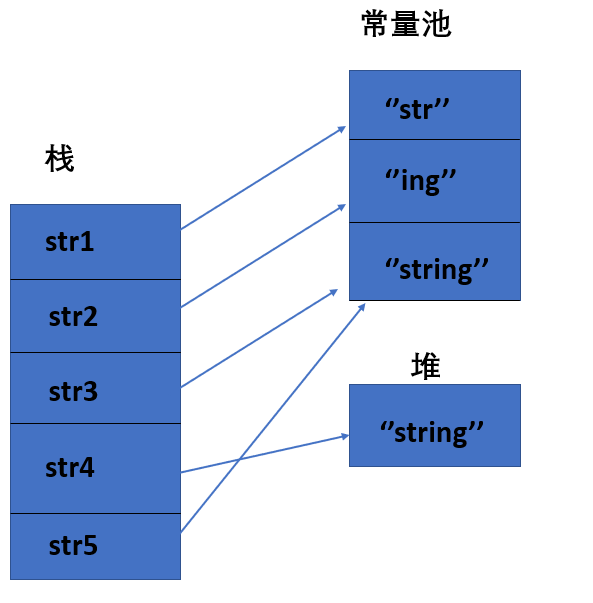
String str4 = str1 + str2; //在堆上创建的新的对象

String str5 = "string";//常量池中的对象

System.out.println(str3 == str4);//false

System.out.println(str3 == str5);//true

System.out.println(str4 == str5);//false



尽量避免多个字符串拼接，因为这样会重新创建对象。如果需要改变字符串的话，可以使用StringBuilder或者StringBuffer。

### 4.2String s1 = new String(“abc”);这句话创建了几个对象？

答：2个

String s1 = new String("abc");// 堆内存的地址值

String s2 = "abc";

System.out.println(s1 == s2);// 输出false,因为一个是堆内存，一个是常量池的内存，故两者是不同的。

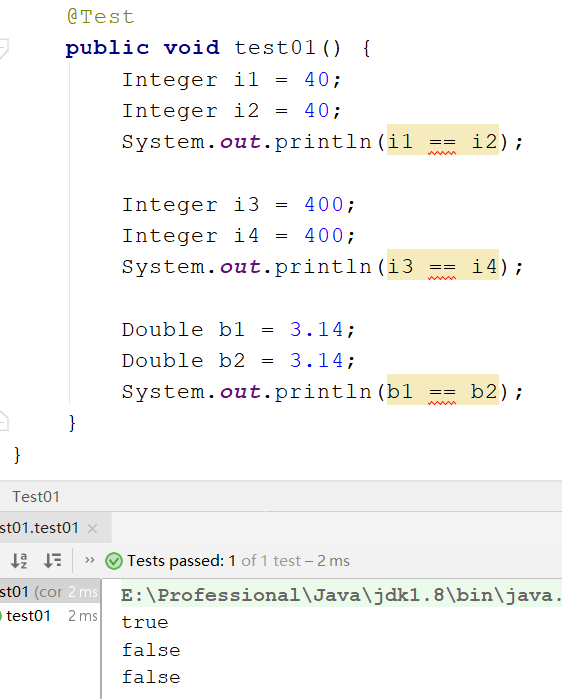
System.out.println(s1.equals(s2));// 输出true

解释：先有字符串”abc”放入常量池，然后new了一份字符串”abc”放入Java堆（字符串常量”abc”在编译期就已经确定放入常量池，而Java堆上的”abc”是在运行期初始化阶段才确定），然后Java栈的str1指向Java堆上的”abc”。

### 4.38种基本类型的包装类和常量池

Java基本类型的包装类的大部分都实现了常量池技术，即Byte，Short、Integer、Long、Character、Boolean，这6种包装类默认创建了数值[-128,127]的相应类型的缓存数据，但是超出此范围仍然会去创建新的对象。

两种浮点数类型的包装类Float，Double并没有实现常量池技术。



Integer缓存源代码：

/\*\*

\*此方法将始终缓存-128到127（包括端点）范围内的值，并可以缓存此范围之外的其他值。

\*/

public static Integer valueOf(int i) {

if (i >= IntegerCache.low && i <= IntegerCache.high) {

return IntegerCache.cache[i + (-IntegerCache.low)];

}

return new Integer(i);

}

应用场景：

1. Integeri1 = 40;Java在编译的时候会直接将代码封装成Integer i1 = Integer.valueOf(40);，从而使用常量池中的对象。
2. Integer i1 = new Integer(40);这种情况下会创建新的对象。

Integer i1 = 40;

Integer i2 = new Integer(40);

System.out.println(i1==i2);//输出false

Integer比较丰富的例子：

Integer i1 = 40;

Integer i2 = 40;

Integer i3 = 0;

Integer i4 = new Integer(40);

Integer i5 = new Integer(40);

Integer i6 = new Integer(0);

System.out.println("i1=i2 " + (i1 == i2));

System.out.println("i1=i2+i3 " + (i1 == i2 + i3));

System.out.println("i1=i4 " + (i1 == i4));

System.out.println("i4=i5 " + (i4 == i5));

System.out.println("i4=i5+i6 " + (i4 == i5 + i6));

System.out.println("40=i5+i6 " + (40 == i5 + i6));

解释：

语句i4==i5+i6，因为+这个操作符不适用于Integer对象，首先i5和i6进行自动拆箱操作，进行数值相加，即i4==40。然后Integer对象无法与数值进行直接比较，所以i4自动拆箱转为int值40，最终这条语句转为40==40进行数值比较。