### 1、判断定义为String类型的st1和st2是否相等

public static void main(String[] args) {

String s1 = "abc";  
 String s2 = "abc";  
 System.*out*.println(s1 == s2);  
 System.*out*.println(s1.equals(s2));  
}

输出结果：

true

true

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。



内存过程大致如下：

* 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
* 第二步，main方法压入栈内存
* 常量池创建一个“abc”对象，产生一个内存地址
* 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”。
* 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
* 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st2
* 最后st1和st2都指向了内存中同一个地址。

### 2、下面这句话在内存中创建了几个对象

String st1 = new String(“abc”);

创建了两个对象，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。



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。

### 3、判定以下定义为String类型的s3和s4是否相等

String s3 = "a" + "b" + "c";

String s4 = "abc";  
System.*out*.println(s3 == s4);  
System.*out*.println(s3.equals(s4));

输出结果：

true

true

“a”“b”“c”三个字符串常量，进行+符号拼接之后编程了“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，再进行s4=“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。

### 4、判断以下st2和st3是否相等

String st1 = "ab";

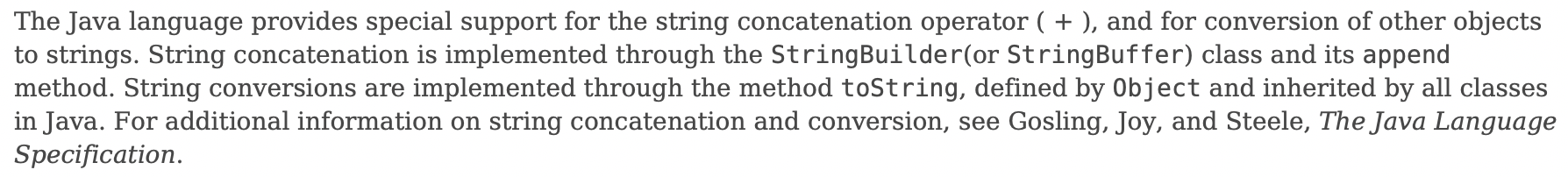
String st2 = "abc";  
String st3 = st1 + "c";  
System.*out*.println(st2 == st3);  
System.*out*.println(st2.equals(st3));

输出结果：

false

true

分析：



Java语言为字符串连接运算符（+）以及将其他对象转换为字符串提供了特殊的支持。 字符串连接是通过StringBuilder（或StringBuffer）类及其append方法实现的。字符串转换是通过toString方法实现的，该方法由Object定义并由Java中的所有类继承。有关字符串连接和转换的其他信息，请参见Java语言规范，Gosling，Joy和Steele。



大致内存过程：

* 常量池创建“ab”对象，并赋值给st1，所以st1指向了“ab”
* 常量池创建“abc”对象，并赋值给st2，所以st2指向了“abc”
* 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。
* 调用了Object的toString方法把StringBuffer对象装换成了String对象。
* 把String对象（0x0022）赋值给st3