# 面试题中常见的String类问题

问题1:  
  
String s1 = "abc";  
String s2 = "abc";  
System.out.println(s1 == s2);  
  
这里的结果是true. 由于字符串是常量(内存中创建对象后不能修改), 而且字符串在程序中经常使用. 所以Java对其提供了缓冲区. 缓冲区内的字符串会被共享. 使用双引号的形式定义字符串常量就是存储在缓冲区中的. 使用"abc"时会先在缓冲区中查找是否存在此字符串, 没有就创建一个, 有则直接使用. 第一次使用"abc"时会在缓冲区中创建, 第二次则是直接引用之前创建好的了.  
  
  
问题2:  
  
String s1 = new String("abc");  
String s2 = new String("abc");  
System.out.println(s1 == s2);  
  
这里的结果是false. 使用构造函数创建的字符串都是在缓冲区外面创建一个新的. 所以只要是使用构造函数创建的字符串, 和任何字符串都不会相等.  
  
  
问题3:  
  
String s1 = "abc";  
String s2 = "a";  
String s3 = "bc";  
String s4 = s2 + s3;  
System.out.println(s1 == s4);  
  
这里的结果是false. Java中字符串的相加其内部是使用StringBuilder类的append()方法和toString()方法来实现的. 而StringBuilder类toString()方法返回的字符串是通过构造函数创建的.  
  
  
问题4:

String s1 = "abc";  
String s2 = "a" + "bc";  
System.out.println(s1 == s2);  
  
这里的结果是true. 其实这里的s2并没有进行字符串相加, 两个双引号形式的字符串常量相加, 在编译的时候直接会被转为一个字符串"abc".  
  
  
问题5:  
  
String str = "abc";  
str.substring(3);  
str.concat("123");  
System.out.println(str);             
  
这里结果是"abc". 由于字符串是常量(内存中创建对象后不能修改), 该类中所有方法都不会改变字符串的值. 如果希望使用一个可变的字符串, 可以使用StringBuilder或StringBuffer类.

# [String详解和字符串缓冲区](http://blog.csdn.net/zhang__jin8809/article/details/16907533)

## 一.String类及常用方法

1.概述：

public final class **String 它是一个final类，所以不可以被继承。**

String 类代表字符串。Java 程序中的所有字符串字面值（如 "abc" ）都作为此类的实例实现。

字符串是常量；它们的值在创建之后不能更改。字符串缓冲区支持可变的字符串。

String 类包括的方法可用于检查序列的单个字符、比较字符串、搜索字符串、提取子字符串、

创建字符串副本并将所有字符全部转换为大写或小写。

    除非另行说明，否则将 null 参数传递给此类中的构造方法或方法将抛出 [NullPointerException](http://blog.csdn.net/zhang__jin8809/article/details/16907533)。

    字符串最大特性：一旦被初始化，就不可以被改变。

2.字符串创建：（字符串常量池和构造方法的参数类型）

    （1）.构造方法：根据参数类型不同分为以下6种：

               空参String()、byte[]类型String(byte[] bytes)、char[]类型String(char[] value)、

              String类型String(String str)、StringBuffer类型String(StringBuffer buffer)、StringBuilder类型String(StringBuilder builder)

              注意：其中接收byte[]类型和char[]类型参数的构造方法中,可以对数组指定offset和length，且byte[]中还可以指定编码集。

     （2）.直接创建字符串常量：位于常量池中，常量字符串相加依然位于常量池中，含引用的相加则会产生位于堆内存的新字符串

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)

1. **public** **static** **void** constantPool(){
2. String str1 = "hey";
3. String str2 = **new** String("hey");
5. String str3 = str1+"world";
6. String str4 = "hey"+"world";
7. String str5 = "heyworld";
9. //常量与new出来的相比为false
10. System.out.println(str1==str2);//false
11. //含引用的字符串与常量比较为false，因为内部会使用字符串缓冲区创建新的对象然后赋值
12. System.out.println(str3==str4);//false
13. //同上
14. System.out.println(str3==str5);//false
15. //常量与常量比较为true
16. System.out.println(str4==str5);//true
17. }

3.String类常用方法：

  （1）.判断：   
      |-- boolean equals(String str) 比较字符串的内容是否相等  
      |-- boolean  equalsIgnoreCase(String str) 忽略大小写进行比较  
      |-- boolean contains(String s) 判断一个字符串是否包含另一个字符串  
      |-- boolean startsWith(String) 判断一个字符串是否以另一个字符串开头  
      |-- boolean endsWith(String) 判断一个字符串是否以另一个字符串结尾  
      |-- boolean isEmpty()判断一个字符串是否有内容

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)[在CODE上查看代码片](https://code.csdn.net/snippets/81603)

1. **public** **static** **void** estimateMethod(){
2. //1。equals方法，比较字符串内容是否相同，==则是比较引用地址是否相同
3. String str1 = "hello";
4. String str2 = **new** String("hello");
5. //地址不同但内容相同，所以equals为true
6. System.out.println(str1.equals(str2));
8. //2。startsWith方法，是否以指定字符串开始
9. String str3 = "我是黑马";
10. System.out.println(str3.startsWith("我是"));
12. //3。endsWith方法，是否以指定字符串结尾
13. System.out.println(str3.endsWith("黑马"));
15. //4。contains方法，是否包含某个字符串
16. System.out.println(str3.contains("是黑"));
18. //5。isEmpty方法，字符串是否为空
19. System.out.println(str3.isEmpty());
20. }

         （2）.获取

      char charAt(int index索引) 返回指定索引位置上的字符  
      String substring(int start,int end)获取字符串中的一部分，包含头不包含尾，返回一个新字符串  
      String substring(int start) 获取从指定位置开始的字符串  
      int length();字符串长度

      int indexOf(char c)查找指定字符在字符串中第一次出现的索引  
      int indexOf(char c,int fromIndex)指定从哪个位置开始查找  
      int indexOf(String s)   
      int indexOf(String s,int fromIndex)

      int lastIndexOf(char c) 和indexOf功能一致，但是反向搜索，返回的下标不能反着数  
      int lastIndexOf(char c,int fromIndex)  
      int lastIndexOf(String s)  
      int lastIndexOf(String s,int fromIndex)

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)[在CODE上查看代码片](https://code.csdn.net/snippets/81603)

1. **public** **static** **void** getMethod(){
2. String str = "我是获取方法中的字符串";
4. //1。charAt方法，获取指定位置的字符
5. System.out.println(str.charAt(0));
7. //2.substring方法，获取指定位置的字符串,含头不含尾,注意end取值范围str.length()
8. System.out.println(str.substring(0));
9. System.out.println(str.substring(0,str.length()));
11. //3。indexOf和lastIndexOf方法，获取指定字符或者字符串的索引，还可以指定索引位置fromIndex
12. System.out.println(str.indexOf('串'));
13. System.out.println(str.lastIndexOf("串"));
14. }

  （3）.转换  
      byte[]b getBytes()字符串转成字节数组  
      char[] ch toCharArray()字符串转成字符数组  
      String toUpperCase()将字符串转成大写  
      String toLowerCase()将字符串转成小写

      static String valueOf() 任意类型转成字符串

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)[在CODE上查看代码片](https://code.csdn.net/snippets/81603)

1. **public** **static** **void** convertMethod(){
2. String str = "1314521";
3. //1。getBytes方法，返回字节数组
4. **byte**[] bytes = str.getBytes();
5. **for**(**byte** i : bytes){
6. System.out.println(**(char)i**);//注意此时的字节数组已经代表为编码集中的标号
7. }
9. String str1 = "Make A Promise";
10. //2。toCharArray()方法，返回字符数组
11. **char**[] chars = str1.toCharArray();
12. **for**(**char** ch : chars){
13. System.out.println(ch);
14. }
16. //3。toUpperCase方法，返回全大写
17. System.out.println(str1.toUpperCase());
19. //4。toLowerCase方法，返回全小写
20. System.out.println(str1.toLowerCase());
22. //5。valueOf方法，将任意类型转为字符串
23. **boolean** flag = **true**;
24. System.out.println(flag);
25. }

 （4）.其他方法  
      String[] split() 切割字符串  
      boolean matches()  字符串匹配  
      String replace(老的,新的)替换字符或者字符串  
      String replaceAll();替换字符串  
      String trim() 去掉字符串两端空格  
      String intern()  优先返回常量池中的字符串，没有才创建

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)[在CODE上查看代码片](https://code.csdn.net/snippets/81603)

1. **public** **static** **void** otherMethod(){
2. String str = " hello world  ";
3. //1。split方法，返回String[]
4. String[] strArr = str.split(" ");//注意空格的位置前后也有影响，在前面则数组长度多1，后面则不
5. System.out.println(strArr.length);
6. **for**(String strEle : strArr){
7. System.out.println(strEle);
8. }
10. //2。matches方法，匹配字符串
11. System.out.println(str.matches(" hello world  "));
13. //3。replace和replaceAll方法
14. System.out.println(str.replace('h','w'));
16. //4。trim方法，取出两端空格
17. System.out.println(str);
18. System.out.println(str.trim());
20. //5。intern方法，优先获取常量池中的字符串，没有则创建
21. String str2 = str.intern();
22. System.out.println(str==str2);
23. }

## 二.StringBuffer和StringBuilder

     1.比较：

      相同处：二者方法都是一样的，都可以创建可变字符串，底层结构都是可变字符数组。  
      区别处：StringBuffer是线程安全的，StringBuilder是线程不安全的，但StringBuilder的效率比StringBuffer高  
                     不涉及安全的时候，建议使用StringBuilder。

      2.特有方法：

         |-- append(任意类型) 向缓冲区追加数据 增  
         |-- insert(int offset,任意类型) 向缓冲区中指定的位置上，插入数据 增  
         |-- delete(int start,int end) 删除缓冲区中的数据 ， 删除  
         |-- setCharAt(int offest,char c) 修改指定位置上的字符，修改  
         |-- reverse()方法，字符序列进行翻转

**[java]** [view plaincopy](http://blog.csdn.net/zhang__jin8809/article/details/16907533)[在CODE上查看代码片](https://code.csdn.net/snippets/81603)

1. **public** **static** **void** main(String[] args)
2. {
3. StringBuffer strBuf = **new** StringBuffer();
4. //1。append方法，追加数据
5. strBuf.append("hello");
6. **boolean** flag = **false**;
7. strBuf.append(flag);
8. **int** number = 12;
9. strBuf.append(number);
11. System.out.println(strBuf);
13. //2。insert方法
14. strBuf.insert(5," world ");
15. System.out.println(strBuf);
17. //3。delete方法
18. strBuf.delete(5,12);
19. System.out.println(strBuf);
21. //4、setCharAt方法
22. strBuf.setCharAt(5,' ');
23. System.out.println(strBuf);
25. //5.reverse方法
26. strBuf.reverse();
27. System.out.println(strBuf);
28. }

## 三. 字符串总结

      字符串的方法比较多，最重要的是要举一反三，而不要思维僵化，当String类本身的方法不足以解决问题时，

      可以考虑转为数组或者转为缓冲区对象，进而可使用他们的方法来解决问题，选择最为合适的方法来提高效率。

      如以下练习：

      1.如何操作一个字符串使其反转，按自然顺序进行排序，大小写互转等等诸如此类操作。

      2.模拟trim方法，去除字符串两端空格

      3.获取一个子串在另一个字符串中出现的次数

      4.获取两个字符串中最大的相同子串