

携程一面：String 类型的变量和常量做"+"运算时发生了什么？

Original Guide哥 JavaGuide Yesterday

收录于话题

#Java 2 #大厂面试 31

大家好，我是 Guide！分享一篇今天起早写的原创。

相关阅读：String s="a"+"b"+"c"，到底创建了几个对象？

前言

看到了一个球友分享的面试题，一定要分享一下。

这个面试题不论是面试还是笔试中都是非常常见的，搞懂原理非常重要！

球友的描述如下：



包子

2021年08月16日 18:08

笔试遇到的一道题，虽说蒙对了吧，但是看不懂，希望哪位大佬能详细的讲一下，谢谢！

```
String a = "a";
String b = "b";
String c = "a" + "b";
String d = a + b;
System.out.println(c == d); false
```

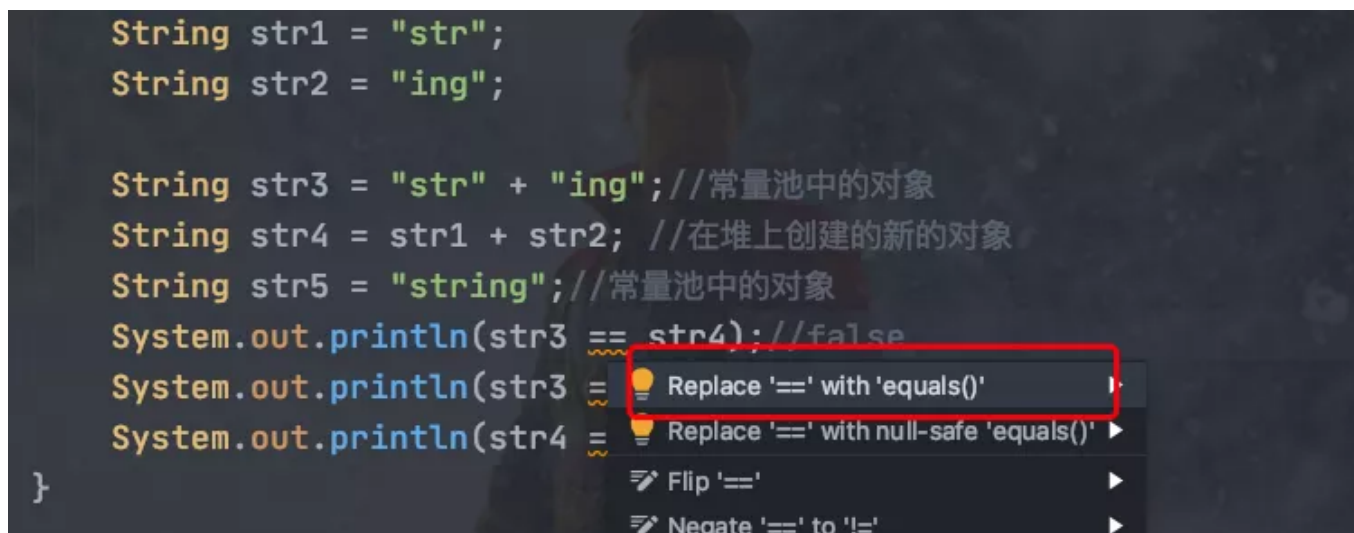
```
final String m = "m";
final String n = "n";
String p = "m" + "n";
String q = m + n;
System.out.println(p == q); true
```

不过，这个问题我们在日常开发中不会遇到。

因为，比较 String 字符串的值是否相等，可以使用 `equals()` 方法。String 中的 `equals` 方法是被重写过的。Object 的 `equals` 方法是比较的对象的内存地址，而 String 的 `equals` 方法比较的是字符串的值是否相等。

```
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
}
```



不过，这个面试题会涉及到很多 Java 基础以及 JVM 相关的知识点。还是非常有必要搞懂的！

问题解答&原理分析

我对问题进行了完善修改，我们先来看字符串不加 `final` 关键字拼接的情况。完善后的代码如下（JDK1.8）：

```
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
```

对于基本数据类型来说，`==` 比较的是值。对于引用数据类型来说，`==`比较的是对象的内存地址。

对于编译期可以确定值的字符串，也就是常量字符串，jvm 会将其存入字符串常量池。

字符串常量池 是 JVM 为了提升性能和减少内存消耗针对字符串（String 类）专门开辟的一块区域，主要目的是为了避免字符串的重复创建。

```
String aa = "ab"; // 放在常量池中

String bb = "ab"; // 从常量池中查找

System.out.println("aa==bb");// true
```

JDK1.7 之前运行时常量池逻辑包含字符串常量池存放在方法区。JDK1.7 的时候，字符串常量池被从方法区拿到了堆中。

并且，字符串常量拼接得到的字符串常量在编译阶段就已经被存放字符串常量池，这个得益于编译器的优化。

在编译过程中，Javac 编译器（下文中统称为编译器）会进行一个叫做 **常量折叠 (Constant Folding)** 的代码优化。《深入理解 Java 虚拟机》中是也有介绍到：

第四部分 程序编译与代码优化

第10章 前端编译与优化

10.1 概述

10.2 Javac编译器

10.2.1 Javac的源码与调试

10.2.2 解析与填充符号表

10.2.3 注解处理器

10.2.4 语义分析与字节码生成

10.3 Java语法的味道

10.4 实战：插入式注解处理器

10.5 本章小结

第11章 后端编译与优化

第五部分 高效并发

附录 A 在 Windows 系统下编译 OpenJDK 6

```
int d = a + c;
int d = b + c;
char d = a + c;
```

后续代码中如果出现了如上3种赋值运算的话，那它们都能构成结构正确的抽象语法树，但是只有第一种写法在语义上是没有错误的，能够通过检查和编译。其余两种在Java语言中是不合逻辑的，无法编译（是否合乎语义逻辑必须限定在具体的语言与具体的上下文环境之中才有意义。如在C语言中，a、b、c的上下文定义不变，第二、三种写法都是可以正确编译的）。我们编码时经常能在IDE中看到由红线标注的错误提示，其中绝大部分都是来源于语义分析阶段的检查结果。

1.标注检查

Javac在编译过程中，语义分析过程可分为标注检查和数据及控制流分析两个步骤，分别由图10-5的attribute()和flow()方法（分别对应图10-5中的过程3.1和过程3.2）完成。

标注检查步骤要检查的内容包括诸如变量使用前是否已被声明、变量与赋值之间的数据类型是否能够匹配，等等，刚才3个变量定义的例子就属于标注检查的处理范畴。在标注检查中，还会顺便进行一个称为**常量折叠 (Constant Folding)**的代码优化，这是Javac编译器会对源代码做的极少量优化措施之一（代码优化几乎都在即时编译器中进行）。如果我们在Java代码中写下如下所示的变量定义：

常量折叠会把常量表达式的值求出来作为常量嵌在最终生成的代码中，这是 Javac 编译器会对源代码做的极少量优化措施之一(代码优化几乎都在即时编译器中进行)。

对于 `String str3 = "str" + "ing";` 编译器会给你优化成 `String str3 = "string";` 。

并不是所有的常量都会进行折叠，只有编译器在程序编译期就可以确定值的常量才可以：

1. 基本数据类型(byte、boolean、short、char、int、float、long、double) 以及字符串常量
2. `final` 修饰的基本数据类型和字符串变量
3. 字符串通过 “+” 拼接得到的字符串、基本数据类型之间算数运算（加减乘除）、基本数据类型的位运算（<<、>>、>>>）

因此，`str1`、`str2`、`str3` 都属于字符串常量池中的对象。

引用的值在程序编译期是无法确定的，编译器无法对其进行优化。

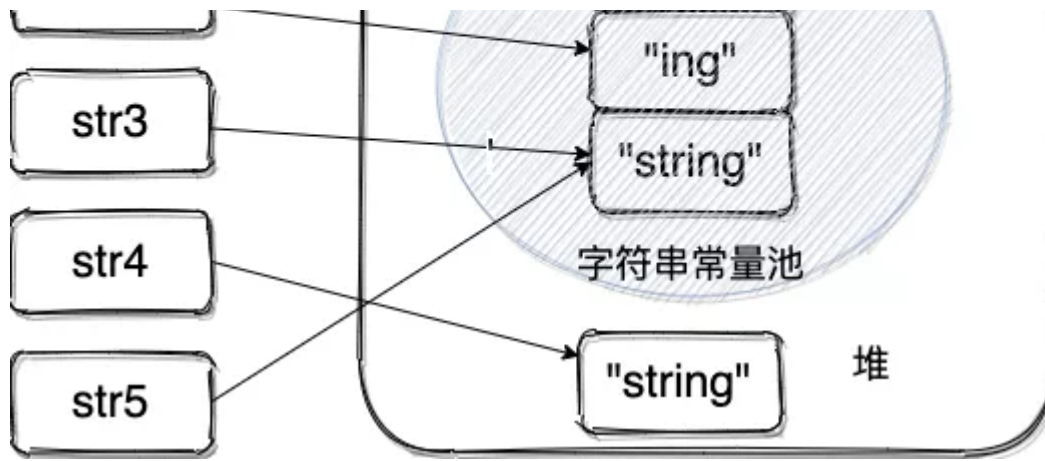
对象引用和“+”的字符串拼接方式，实际上是通过 `StringBuilder` 调用 `append()` 方法实现的，拼接完成之后调用 `toString()` 得到一个 `String` 对象。

```
String str4 = new StringBuilder().append(str1).append(str2).toString();
```

因此，`str4` 并不是字符串常量池中存在的对象，属于堆上的新对象。

我画了一个图帮助理解：





我们在平时写代码的时候，尽量避免多个字符串对象拼接，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`。

不过，字符串使用 `final` 关键字声明之后，可以让编译器当做常量来处理。

```
final String str1 = "str";
final String str2 = "ing";
// 下面两个表达式其实是等价的
String c = "str" + "str2"; // 常量池中的对象
String d = str1 + str2; // 常量池中的对象
System.out.println(c == d); // true
```

被 `final` 关键字修改之后的 `String` 会被编译器当做常量来处理，编译器在程序编译期就可以确定它的值，其效果就想到于访问常量。

如果，编译器在运行时才能知道其确切值的话，就无法对其优化。

示例代码如下（`str2` 在运行时才能确定其值）：

```
final String str1 = "str";
final String str2 = getStr();
String c = "str" + "str2"; // 常量池中的对象
```

```
String c = str1 + str2; // 常量池中的对象  
String d = str1 + str2; // 常量池中的对象  
System.out.println(c == d); // false  
public static String getStr() {  
    return "ing";  
}
```

我们再来看一个类似的问题！

```
String str1 = "abcd";  
String str2 = new String("abcd");  
String str3 = new String("abcd");  
System.out.println(str1==str2);  
System.out.println(str2==str3);
```

上面的代码运行之后会输出什么呢？

答案是：

```
false  
false
```

这是为什么呢？

我们先来看下面这种创建字符串对象的方式：

```
// 从字符串常量池中拿对象  
String str1 = "abcd";
```

这种情况下，jvm 会先检查字符串常量池中有没有"abcd"，如果字符串常量池中
没有，则创建一个，然后 str1 指向字符串常量池中的对象，如果有，则直接将
str1 指向"abcd"；

因此，str1 指向的是字符串常量池的对象。

我们再来看下面这种创建字符串对象的方式：

```
// 直接在堆内存空间创建一个新的对象。  
String str2 = new String("abcd");  
String str3 = new String("abcd");
```

只要使用 new 的方式创建对象，便需要创建新的对象。

使用 new 的方式创建对象的方式如下，可以简单概括为 3 步：

1. 在堆中创建一个字符串对象
2. 检查字符串常量池中是否有和 new 的字符串值相等的字符串常量
3. 如果没有的话需要在字符串常量池中也创建一个值相等的字符串常量，如果有的话，就直接返回堆中的字符串实例对象地址。

因此，str2 和 str3 都是在堆中新创建的对象。

字符串常量池比较特殊，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. 如果不是用双引号声明的 String 对象，使用 String 提供的 intern() 方法也有同样的效果。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，JDK1.7 之前（不包含 1.7）的处理方式是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，JDK1.7 以及之后的处理方式是在常量池中记录此字符串的引用，并返回该引用。

示例代码如下：

```
String s1 = "计算机";  
String s2 = s1.intern();  
System.out.println(s1 == s2);
```

JDK1.7 之前的输出（不包含 1.7）：

```
false
```

JDK1.7 以及之后版本的输出（包含 1.7）：

```
true
```

推荐阅读

- R 大 (RednaxelaFX) 关于常量折叠的回答：
<https://www.zhihu.com/question/55976094/answer/147302764>
- 《深入理解 Java 虚拟机》第 10 章程序编译与代码优化

总结

1. 对于基本数据类型来说，`==`比较的是值。对于引用数据类型来说，`==`比较的是对象的内存地址。
2. 在编译过程中，Javac 编译器（下文中统称为编译器）会进行一个叫做 **常量折叠 (Constant Folding)** 的代码优化。常量折叠会把常量表达式的值求出来作为常量嵌在最终生成的代码中，这是 Javac 编译器会对源代码做的极少量优化措施之一(代码优化几乎都在即时编译器中进行)。

3. 一般来说，我们要尽量避免通过 `new` 的方式创建字符串。使用双引号声明的 `String` 对象（`String s1 = "java"`）更利于让编译器有机会优化我们的代码，同时也更易于阅读。
4. 被 `final` 关键字修改之后的 `String` 会被编译器当做常量来处理，编译器程序编译期就可以确定它的值，其效果就想到于访问常量。

< END >

也许你还想看

- | 我在 B 站淘了 2 个 Java 实战项目！小破站，YYDS！
- | 我常用的20+个学习编程的网站！芜湖起飞！
- | 1w+字的 Dubbo 面试题/知识点总结！（2021 最新版）
- | 7年前，24岁，出版了一本 Redis 神书
- | 京东二面：为什么需要分布式ID？你项目中是怎么做的？
- | 和 Github 在一起 5 年多了.....
- | 一键生成数据库文档，堪称数据库界的Swagger
- | 来看看这个超好用的项目脚手架吧！5分钟搭建一个Spring Boot 前后端分离系统！
- | Java 集合使用不当，Code Review 被 diss了！

我是 Guide哥，一个工作2年有余，接触编程已经6年有余的程序员。大三开源 JavaGuide，目前已经 100k+ Star。未来几年，希望持续完善 JavaGuide，争取能够帮助更多学习 Java 的小伙伴！共勉！淦！[点击即可了解我的个人经历](#)。

欢迎点赞分享。咱们下期再会！

收录于话题 #大厂面试·31个

下一篇 · 1w+字的 Dubbo 面试题/知识点总结！（2021 最新版）

People who liked this content also liked

sizeof(变量)可不可以秀一点操作？

