



Java性能调优实战
刘超
金山软件西山居技术经理
[查看详情](#)

5786 人已学习

03 | 字符串性能优化不容小觑，百M内存轻松存储几十G数据

刘超 2019-05-25

64

76

52

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95



03 | 字符串性能优化不容小觑，百M内存轻松存储几十G数据

04 | 慎重使用正则表达式

05 | ArrayList还是LinkedList？使用不当性能差千倍

加餐 | 推荐几款常用的性能测试工具

你好，我是刘超。

从第二个模块开始，我将带你学习 Java 编程的性能优化。今天我们就从最基础的 String 字符串优化讲起。

String 对象是我们使用最频繁的一个对象类型，但它的性能问题却是最容易被忽略的。String 对象作为 Java 语言中重要的数据类型，是内存中占据空间最大的一个对象。**高效地使用字符串，可以提升系统的整体性能。**

接下来我们就从 String 对象的实现、特性以及实际使用中的优化这三个方面入手，深入了解。

在开始之前，我想先问你一个小问题，也是我在招聘时，经常会问到面试者的一道题。虽是老生常谈了，但错误率依然很高，当然也有一些面试者答对了，但能解释清楚答案背后原理的人少之又少。问题如下：

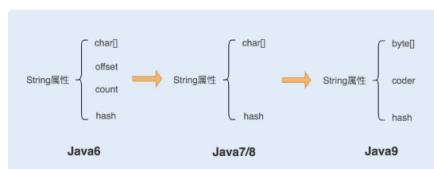
通过三种不同的方式创建了三个对象，再依次两两匹配，每组被匹配的两个对象是否相等？代码如下：

```
1 String str1 = "abc";
2 String str2 = new String("abc");
3 String str3= str2.intern();
4 assertEquals(str1==str2);
5 assertEquals(str2==str3);
6 assertEquals(str1==str3)
7
```

你可以先想想答案，以及这样回答的原因。希望通过今天的学习，你能拿到满分。

String 对象是如何实现的？

在 Java 语言中，Sun 公司的工程师们对 String 对象做了大量的优化，来节约内存空间，提升 String 对象在系统中的性能。一起来看看优化过程，如下图所示：



1. 在 Java6 以及之前的版本中，String 对象是对 char 数组进行了封装实现的对象，主要有四个成员变量：char 数组、偏移量 offset、字符数量 count、哈希值 hash。

String 对象是通过 offset 和 count 两个属性来定位 char[] 数组，获取字符串。这么做可以高效、快速地共享数组对象，同时节省内存空间，但这种方式很有可能会导致内存泄漏。

2. 从 Java7 版本开始到 Java8 版本，Java 对 String 做了一些改变。String 类中不再有 offset 和 count 两个变量了。这样的好处是 String 对象占用的内存稍微少了些，同时，String.substring 方法也不再共享 char[]，从而解决了使用该方法可能导致的内存泄漏问题。

3. 从 Java9 版本开始，工程师将 char[] 字段改为了 byte[] 字段，又维护了一个新的属性 coder，它是一个编码格式的标识。

工程师为什么这样修改呢？

我们知道一个 char 字符占 16 位，2 个字节。这个情况下，存储单字符编码的字符（占一个字节的字符）就显得非常浪费。JDK1.9 的 String 类为了节约内存空间，于是使用了占 8 位，1 个字节的 byte 数组来存放字符串。

而前面对 coder 的UTF用法，在字符串中使用或者使用 INOEXUT () 函数时，我们需要根据这个字段，判断如何计算字符串长度。coder 属性默认有 0 和 1 两个值，0 代表 Latin-1（单字节编码），1 代表 UTF-16。如果 String 判断字符串只包含了 Latin-1，则 coder 属性值为 0，否则为 1。

String 对象的不可变性

了解了 String 对象的实现后，你有没有发现在实现代码中 String 类被 final 关键字修饰了，而且变量 char 数组也被 final 修饰了。

我们知道类被 final 修饰代表该类不可继承，而 char[] 被 final+private 修饰，代表了 String 对象不可被更改。Java 实现的这个特性叫作 String 对象的不可变性，即 String 对象一旦创建成功，就不能再对它进行改变。

Java 这样做的好处在哪里呢？

第一，保证 String 对象的安全性。假设 String 对象是可变的，那么 String 对象将可能被恶意修改。

第二，保证 hash 属性值不会频繁变更，确保了唯一性，使得类似 HashMap 容器才能实现相应的 key-value 缓存功能。

第三，可以实现字符串常量池。在 Java 中，通常有两种创建字符串对象的方式，一种是通过字符串常量的方式创建，如 String str="abc"；另一种是字符串变量通过 new 形式的创建，如 String str = new String("abc")。

当代码中使用第一种方式创建字符串对象时，JVM 首先会检查该对象是否在字符串常量池中，如果在，就返回该对象引用，否则新的字符串将在常量池中被创建。这种方式可以减少同一个值的字符串对象的重复创建，节约内存。

String str = new String("abc") 这种方式，首先在编译类文件时，“abc”常量字符串将会放入到常量结构中，在类加载时，“abc”将会在常量池中创建；其次，在调用 new 时，JVM 命令将会调用 String 的构造函数，同时引用常量池中的“abc”字符串，在堆内存中创建一个 String 对象；最后，str 将引用 String 对象。

这里附上一个你可能会想到的经典反例。

平常编程时，对一个 String 对象 str 赋值“hello”，然后又让 str 值为“world”，这个时候 str 的值变成了“world”。那么 str 值确实改变了，为什么我还说 String 对象不可变呢？

首先，我来解释下什么是对象和对象引用。Java 初学者往往对此存在误区，特别是一些从 PHP 转 Java 的同学。在 Java 中要比较两个对象是否相等，往往是用 ==，而要判断两个对象的值是否相等，则需要用 equals 方法来判断。

这是因为 str 只是 String 对象的引用，并不是对象本身。对象在内存中是一块内存地址，str 则是一个指向该内存地址的引用。所以在刚刚我们说的这个例子中，第一次赋值的时候，创建了一个“hello”对象，str 引用指向“hello”地址；第二次赋值的时候，又重新创建了一个对象“world”，str 引用指向了“world”，但“hello”对象依然存在于内存中。

也就是说 str 并不是对象，而只是一个对象引用。真正的对象依然还在内存中，没有被改变。

String 对象的优化

了解了 String 对象的实现原理和特性，接下来我们就结合实际场景，看看如何优化 String 对象的使用，优化的过程中又有哪些需要注意的地方。

1. 如何构建超大字符串

编程过程中，字符串的拼接很常见。前面我讲过 String 对象是不可变的，如果我们使用 String 对象相加，拼接我们想要的字符串，是不是就会产生多个对象呢？例如以下代码：

```
1 String str= "ab" + "cd" + "ef";  
2
```

分析代码可知：首先会生成 ab 对象，再生成 abcd 对象，最后生成 abcdef 对象，从理论上来说，这段代码是低效的。

但实际运行中，我们发现只有一个对象生成，这是为什么呢？难道我们的理论判断错了？我们再来看编译后的代码，你会发现编译器自动优化了这行代码，如下：

```
1 String str= "abcdef";  
2
```

上面我介绍的是字符串常量的累计，我们再来看看字符串变量的累计又是怎样的呢？

```
1 String str = "abcdef";  
2  
3 for(int i=0; i<1000; i++) {  
4     str = str + i;  
5 }  
6
```

上面的代码编译后，你可以看到编译器同样对这段代码进行了优化。不难发现，Java 在进行字符串的拼接时，偏向使用 StringBuilder，这样可以提高程序的效率。

```
1
2 String str = "abcdef";
3
4 for(int i=0; i<1000; i++) {
5     str = (new StringBuilder(String.valueOf(str))).append(i).toString();
6 }
7
```

综上已知：即使使用 + 号作为字符串的拼接，也一样可以被编译器优化成 StringBuilder 的方式。但再细致些，你会发现在编译器优化的代码中，每次循环都会生成一个新的 StringBuilder 实例，同样也会降低系统的性能。

所以平时做字符串拼接的时候，我建议你还是要显示地使用 StringBuilder 来提升系统性能。

如果在多线程编程中，String 对象的拼接涉及到线程安全，你可以使用 StringBuffer。但是要注意，由于 StringBuffer 是线程安全的，涉及到锁竞争，所以从性能上来说，要比 StringBuilder 差一些。

2. 如何使用 String.intern 节省内存？

讲完了构建字符串，我们再来讨论下 String 对象的存储问题。先看一个案例。

Twitter 每次发布消息状态的时候，都会产生一个地址信息，以当时 Twitter 用户的规模预估，服务器需要 32G 的内存来存储地址信息。

```
1 public class Location {
2     private String city;
3     private String region;
4     private String countryCode;
5     private double longitude;
6     private double latitude;
7 }
8
```

考虑到其中有很多用户在地址信息上是有重合的，比如，国家、省份、城市等，这时就可以将这部分信息单独列出一个类，以减少重复，代码如下：

```
1
2 public class SharedLocation {
3
4     private String city;
5     private String region;
6     private String countryCode;
7 }
8
9 public class Location {
10
11     private SharedLocation sharedLocation;
12     double longitude;
13     double latitude;
14 }
15
```

通过优化，数据存储大小减到了 20G 左右。但对于内存存储这个数据来说，依然很大，怎么办呢？

这个案例来自一位 Twitter 工程师在 QCon 全球软件开发大会上的演讲，他们想到的解决方法，就是使用 String.intern 来节省内存空间，从而优化 String 对象的存储。

具体做法就是，在每次赋值的时候使用 String 的 intern 方法，如果常量池中有相同值，就会重复使用该对象，返回对象引用，这样一开始的对象就可以被回收掉。这种方式可以使重复性非常高的地址信息存储大小从 20G 降到几百兆。

```
1 SharedLocation sharedLocation = new SharedLocation();
2
3 sharedLocation.setCity(messageInfo.getCity().intern());           sharedLocation.setRegion(messageInfo.getRegion().intern());
4 sharedLocation.setCountryCode(messageInfo.getCountryCode().intern());
5
6 Location location = new Location();
7 location.set(sharedLocation);
8 location.set(messageInfo.getLongitude());
9 location.set(messageInfo.getLatitude());
10
```

为了更好地理解，我们再来通过一个简单的例子，回顾下其中的原理：

```
1 String a = new String("abc").intern();
2 String b = new String("abc").intern();
3
4 if(a==b) {
5     System.out.print("a==b");
6 }
7
```

输出结果：

```
1 a==b
2
```

在字符串常量中，默认会将对象放入常量池；在字符串变量中，对象是会创建在堆内存中，同时

如果调用 intern 方法，会去查看字符串常量池中是否有等于该对象的字符串，如果没有，在常量池中新增该对象，并返回对该对象引用；如果有，就返回常量池中的字符串引用。堆内存中原有的对象由于没有引用指向它，将会通过垃圾回收器回收。

了解了原理，我们再一起看看上边的例子。

在一开始创建 a 变量时，会在堆内存中创建一个对象，同时会在加载类时，在常量池中创建一个字符串对象，在调用 intern 方法之后，会去常量池中查找是否有等于该字符串的对象，有就返回引用。

在创建 b 字符串变量时，也会在堆中创建一个对象，此时常量池中有该字符串对象，就不再创建。调用 intern 方法则会去常量池中判断是否有等于该字符串的对象，发现有等于“abc”字符串的对象，就直接返回引用。而在堆内存中的对象，由于没有引用指向它，将会被垃圾回收。所以 a 和 b 引用的是同一个对象。

下面我用一张图来总结下 String 字符串的创建分配内存地址情况：



使用 intern 方法需要注意的一点是，一定要结合实际场景。因为常量池的实现是类似于一个 HashTable 的实现方式，HashTable 存储的数据越大，遍历的时间复杂度就会增加。如果数据过大，会增加整个字符串常量池的负担。

3. 如何使用字符串的分割方法？

最后我想跟你聊聊字符串的分割，这种方法在编码中也很常见。Split() 方法使用了正则表达式实现了其强大的分割功能，而正则表达式的性能是非常不稳定的，使用不恰当会引起回溯问题，很可能导致 CPU 居高不下。

所以我们应该慎重使用 Split() 方法，我们可以用 String.indexOf() 方法代替 Split() 方法完成字符串的分割。如果实在无法满足需求，你就在使用 Split() 方法时，对回溯问题加以重视就可以了。

总结

这一讲中，我们认识到做好 String 字符串性能优化，可以提高系统的整体性能。在这个理论基础上，Java 版本在迭代中通过不断地更改成员变量，节约内存空间，对 String 对象进行优化。

我们还特别提到了 String 对象的不可变性，正是这个特性实现了字符串常量池，通过减少同一个值的字符串对象的重复创建，进一步节约内存。

但也是因为这个特性，我们在做长字符串拼接时，需要显示使用 StringBuilder，以提高字符串的拼接性能。最后，在优化方面，我们还可以使用 intern 方法，让变量字符串对象重复使用常量池中相同值的对象，进而节约内存。

最后再分享一个个人观点。那就是千里之堤，溃于蚁穴。日常编程中，我们往往可能就是对一个小小的字符串了解不够深入，使用不够恰当，从而引发线上事故。

比如，在我之前的工作经历中，就曾因为使用正则表达式对字符串进行匹配，导致并发瓶颈，这里也可以将其归纳为字符串使用的性能问题。具体实战分析，我将在 04 讲中为你详解。

思考题

通过今天的学习，你知道文章开头那道面试题的答案了吗？背后的原理是什么？

互动时刻

今天除了思考题，我还想和你做一个简短的交流。

上两讲中，我收到了很多留言，在此非常感谢你的支持。由于前两讲是概述内容，主要是帮你建立对性能调优的整体认识，所以相对来说重理论、偏基础。但我发现，很多同学都有这样迫切的愿望，那就是赶紧学会使用排查工具，监测分析性能，解决当下的一些问题。

我这里特别想分享一点，其实性能调优不仅仅是学会使用排查监测工具，更重要的是掌握背后的调优原理，这样你不仅能够独立解决同一类的性能问题，还能写出高性能代码，所以我希望给你的学习路径是：夯实基础 - 结合实战 - 实现进阶。

最后，欢迎你积极发言，讨论思考题或是你遇到的性能问题都可以，我会知无不尽。也欢迎你点击“请朋友读”，把今天的内容分享给身边的朋友，邀请他一起讨论。



胡晓

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Command + Enter 发表

0/2000字

提交留言

64

76

精选留言(64)

KL3

老师，能解释下，
“String.substring 方法也不再共享 char[]，从而解决了使用该方法可能导致的内存泄漏问题。”

共享char数组可能导致内存泄露问题？

作者回复: 你好 KL3，在Java6中substring方法会调用new String构造函数，此时会复用原来的char数组，而如果我们仅仅是用substring获取一小段字符，而原本String字符串非常大的情况下，substring的对象如果一直被引用，由于substring的里面的char数组仍然指向原字符串，此时String字符串也无法回收，从而导致内存泄露。

试想下，如果有大量这种通过substring获取超大字符串中一小段字符串的操作，会因为内存泄露而导致内存溢出。

2019-05-25

28

扫地僧

答案是false,false,true。背后的原理是：

1. String str1 = "abc"通过字面量的方式创建，abc存储于字符串常量池中；
2. String str2 = new String("abc");通过new对象的方式创建字符串对象，引用地址存放在堆内存中，abc则存放在字符串常量池中；所以下str1 == str2?显然是false
3. String str3 = str2.intern();由于str2调用了intern()方法，会返回常量池中的数据，地址直接指向常量池，所以str1 == str3；而str2和str3地址值不等所以也是false (str2指向堆空间，str3直接指向字符串常量池)。不知道这样理解有木有问题

作者回复: 答案非常正确，理解了这个题目基本理解了String的特性了。

2019-05-25

14

失火的夏天

开头题目答案是false false true
str1是建立在常量池中的"abc"，str2是new出来，在堆内存里的，所以str1!=str2，
str3是通过str2.intern()出来的，str1在常量池中已经建立了"abc"，这个时候str3是从常量池里取出来的，和str1指向的是同一个对象，自然也就有了str1==str3, str3!=str2了

作者回复: 这里我纠正下，str3是intern返回的引用，intern而不是创建出来的。

你的答案是正确的！

2019-05-25

13

快乐的五五开

自学一年居然不知道有String.intern()这个方法😭😭
不过从Java8开始（大概）String.split()传入长度为1字符串的时候并不会使用正则，这种情况还是可以用

作者回复: 非常感谢Geek的补充，我在这里也再补充一个小点，split有两种情况不会使用正则表达式：

第一种为传入的参数长度为1，且不包含".\\$|[({^?*+\\")regex元字符的情况下，不会使用正则表达式；

第二种为传入的参数长度为2，第一个字符是反斜杠，并且第二个字符不是ASCII数字或ASCII字母的情况下，不会使用正则表达式。

2019-05-25

10

风朝

使用 intern 方法需要注意的一点是，一定要结合实际场景。因为常量池的实现是类似于一个 HashTable 的实现方式，HashTable 存储的数据越大，遍历的时间复杂度就会增加。如果数据过大，会增加整个字符串常量池的负担。
像国家地区是有边界的。像其他情况，怎么把握这个度呢？

作者回复: 如果对空间要求高于时间要求，且存在大量重复字符串时，可以考虑使用常量池存储。

如果对查询速度要求很高，且存储字符串数量很大，重复率很低的情况下，不建议存储在常量池中。

具体可以通过模拟测试自己的场景，对比两种存储方式的性能，通过数据来给自己答案。

2019-05-25

Eric

对于您文中“一开始创建 a 变量时，会在堆内存中创建一个对象，同时在常量池中创建一个字符串对象”这句话 我认为前部分没有问题 分歧点在后面那部分 我觉得abc常量早就在运行时常量池就存在了 可以理解使用这个类之前 就已经构造好了运行时常量池 而运行时常量池中就包括“abc”常量 至于使用 new String("abc") 我觉得它应该只会在堆中创建String对象 并将运行时常量池中已经存在的“abc”常量的引用作为构造函数的参数而已

作者回复：你理解的分歧点是对的，这个构造是在加载类时，就已经在常量池中构造好常量。

2019-05-25

Zend

“在字符串变量中，对象是会创建在堆内存中，同时也会在常量池中创建一个字符串对象，复制到堆内存对象中，并返回堆内存对象引用。”
比如：

是从常量池中复制到堆内存，这时常量池中字符串与堆内存字符串是完全独立的，内部也不存在引用关系？

作者回复：你好 Zend，具体的复制过程是先将常量池中的字符串压入栈中，在使用String的构造方法时，会拿到栈中的字符串作为构造方法的参数。这里我纠正一点，今天我查看了下这个构造函数，String的构造函数是一个char数组赋值过程，不是new char[]重新创建，所以是引用了常量池中的字符串对象，存在引用关系。

2019-05-26

Eric

我在《Java虚拟机规范》里面看到一句话 这句话是当类或接口创建时，它的二进制表示中的常量池表被用来构造运行时常量池 我理解的意思是 类或接口 创建时就根据.class文件的常量池表生成了运行时常量池 执行new String("abc")这行代码应该只会生成一个String对象 并且调用它的构造函数 参数是运行时常量池里面“abc”字符串常量的Reference类型的数据（可以理解为指针吧）怎么会在运行代码执行的时候才会在运行时常量池生成“abc”对象呢？

作者回复：如果是需要按照创建顺序来讲，常量“abc”，则会在加载编译时构造常量池时在常量池中创建“abc”字符串对象，而new对象的构造函数是在运行时创建并复制常量池中的“abc”。还有一个运行时常量池。也就是说，在运行时创建的字符串对象，通过intern方法会在运行时常量池中创建字符串对象。

2019-05-25

Eric

String s1 = new String("abc").intern()

Code:

```
0: new #2 // class java/lang/String
3: dup
4: ldc #3 // String abc
6: invokespecial #4 // Method java/lang/String.<init>:(Ljava/lang/String;)V
9: invokevirtual #5 // Method java/lang/String.intern():Ljava/lang/String;
12: astore_1
13: return
```

9:invokevirtual的时候 常量池里面应该早就有了“abc”这个字符串常量了吧 为什么文中说的是先去堆中创建一个String对象 然后再去常量池创建一个字符串常量？我理解错误了吗？

作者回复：我们可以看到0 new，即是生成了一个对象，这个对象是在堆内存中创建的，之后4 ldc则是将常量池中创建的字符串abc压入栈中，invokespecial调用构造方法复制abc字符串到对象中，invokevirtual调用intern本地方法，返回常量池中的对象引用给s1。

new String("abc")是会创建两个对象的，一个是堆对象，一个是常量池中的对象，intern会去判断常量池中是否有，这个时候是有的，所以不会创建，而是改变s1的引用。

不知道这样是否更好理解？

2019-05-25

建国

在实际编码中我们应该使用什么方式创建字符串呢？

- A.String str= "abcdef";
- B.String str= new String("abcdef");
- C.String str= new String("abcdef"). intern();
- D.String str=str.intern();

作者回复：实际编码中，我们要结合实际场景来选择创建字符串的方式，例如，在创建局部变量以及常量时，我们一般使用A这种方式；如果我们要区别一个字符串创建两个不同的对象来使用时，会选择B：intern一般使用的比较少，例如我们平时会创建很多一样的字符串的对象时，且对象会保存在内存中，我们可以考虑使用intern方法来减少过多重复对象占用内存空间。

2019-05-25

Teanmy

老师好，有一点始终想不明白，请老师解惑，非常感谢！

老师先帮忙看看关于这两行代码，我的分析是否正确：

```
str1 = "abc";  
str2 = new String("abc")
```

```
str1 = "abc";  
1.str1首先是在字符串常量池中寻找"abc"，找到则取其地址，找不到则创建并返回其地址
```

```
2.将该地址赋值给栈的str1
```

```
str2 = new String("abc")
```

```
1.在堆中创建String对象。我查阅了String构造方法源码，实际值取的是"abc"的（此时"abc"已经存在字符串常量池中）引用，也就是说，str2还是指向常量池，并没有创建新的"abc"。
```

```
public String(String original) {  
    this.value = original.value;  
    this.hash = original.hash;  
}
```

```
2.堆中创建完String对象，将该对象的地址赋值给栈变量str2
```

疑问：

既然不管是以上哪种方式，最终实际引用的还是常量池中的"abc"，str2 = new String("abc")只是增加了一个堆中String的“空壳”对象而已（因为实际上char[]指向的还是常量池中的"abc"），这个空壳对象并不会占用过多内存。而.intern的实质只是减少了这个中间的String空壳对象，那何来twitter通过.intern减少大量内存？

作者回复：你好 teamny。运行时创建的字符串对象只会在堆中创建一个对象。在这个前提下，如果有相同值的对象创建，使用intern可以减少重复字符串的创建。例如，有广东省/深圳市/南山区，如果有千万个人发布消息。创建了地址对象，这样导致千万个“广东省”对象在堆内存中创建，如果长时间引用，这些对象都没法释放，使用intern将“广东省”放到常量池中，其他对象引用常量池中的同一个“广东省”字符串，而堆中的千万个对象将被回收。

如果有疑问，请继续留言。

2019-06-02

BugMaker

刘老师您好！“使用 intern 方法需要注意的一点是，一定要结合实际场景。因为常量池的实现是类似于一个 HashTable 的实现方式，HashTable 存储的数据越大，遍历的时间复杂度就会增加。如果数据过大，会增加整个字符串常量池的负担”，那个 Twitter 工程师在 QCon 全球软件开发大会上的演讲的那个 intern 方法是如何做到遍历这么多常量池的数据，同时保证性能的呢？

作者回复：你好，如果我们的数据对查询速度没有这么高要求，可以考虑使用。

2019-05-31

Only now

看了本篇几乎全部留言，感觉包括老师在内，对于“字符串常量池”和“常量池”，这两概念用的很混。

对于jdk7 以及之前的jvm版本不再去深究了，它的字符串常量池存在于方法区，但是jdk8以后，它存在于ava堆中，唯一，且由java.lang.String类维护。它和类文件常量池，运行时常量池没有半毛钱的关系。

最后我有个疑问问老师，字符串常量池中的对象，在失去了所有外部引用之后，会被gc掉吗？

作者回复：非常感谢only now的总结，这一讲中没有详细去区分常量池，而是在强调字符串的使用，后面我们在JVM中可以再一起研究下常量池。

JVM文献中提到方法区是存在垃圾回收。我们可以通过Intern方法来验证这个gc问题，通过大量请求请求某个接口，传入参数创建字符串对象，之后通过intern方法在常量池中生成字符串对象，之后失去引用，观察gc情况。

2019-05-29

晓杰

回答开篇的问题：

str1会在常量池中创建一个对象

str2首先会在内存中创建一个对象，然后在加载类的时候在常量池创建一个字符串对象，同时复制到堆内存对象中，并返回堆内存对象的引用

str3会先去常量池中查看是否存在该字符串相等的对象，因为str1已经在常量池创建了一个相同的对象，所以str1和str2相等。

综上： str1和str2不相等， str1和str3相等， str2和str3不相等

2019-05-26

-W.LI-

老师你说的对！直接把char数组做为参数的String对象里的value数组地址是不一样的。调用intern()方法后就一样了。是我搞错了，然后回到第一个问题的后半部分，我打印输出a==b是false。之前有看到char数组的地址是一样的。这说明new虽然在堆中新建了一个String对象，但是里面的char数组是复用的。这样做的目的是为了节约char数组的内存开销，然后String本身就是不可变对象，复用char数组不会带来问题。问题一：这个char数组是存放在哪的啊？堆还是常量池，其实我不知道常量池具体是个啥，上课老师说的类的hashmap，这样的话就是接近O(1)随机读取。不知道它能不能存char[]。问题二：复用char[]我猜是这么实现的new创建String时会去常量池中查找对应的String存在拿取char[]复用，如果这样的话其实char[]到底存在在哪不太重要。问题三：常量池的内存会回收么？突然觉得自己对常量池一无所知。。。常量池的生命周期一无所知。

作者回复：char数组是存放在常量池中，常量是会在编译时生成字面量，在类加载时加载到常量池中。

这个存放位置还是重要的，这就相当于权职划分，每个位置都有自己的功能和职责。

常量池中的垃圾回收，也是垃圾回收器完成，只要没有根引用的对象，包括类信息等等，都会在回收期被回收掉。常量池中的常量一般是固定的，不像对中的对象。

空

老师，java8还有字符串常量池吗，都整合到堆里面去了吧

作者回复: 有的，java8字符串常量池是分配到堆中，并不代表字符串常量池就取消了。

2019-05-30

... 1

大海

String s = new String("abc").intern();

既然使用intern也会引用到常量池,那么和 使用intern 和 直接使用 String s = "abc"有差别吗

作者回复: 最终实现达到的结果是一样的，但过程不一样。我拿这个例子来说明。在程序运行期间动态创建的字符串对象，由于这类字符串是在内存中开辟的地址空间存放字符串，可以使用intern方法放在常量池中。

2019-05-26

... 64

业余草

final 标示的一般为常量，按照老师说的 new String("abc").intern() 在常量池中也存在 abc，String str1 = "abc";通过子量的方式创建，abc存储于字符串常量池中；是不是说用不用 final 都无所谓了？请帮忙详细解答一下，谢谢！！！

作者回复: 正是因为final，字符串才实现了不可变性，String内部的value已经被final修饰，所以我们不用再在编码时用final修饰。

2019-05-25

... 1

-W.LI-

老师好jdk8环境下，观察了下。不管是直接赋值还是new创建新对象，String对象的value对应的char数组地址都是同一个，这么做就是不论字符串多大，重复new只会多消耗string对象必须得的16字节内存是么？jdk8以前常量池在方法区中，属于永年代，GC不会回收这一部分空间是么？jdk8对常量池做了改动，放在了堆中，在堆中会被GC回收。我想知道为啥要做这个调整。常量池也是在新生代创建，然后几次ygc以后进入老年代么？

作者回复: 这位同学 好，不好意思呀，我没有理解你的第一个问题，你在问substring在java8 创建新对象的问题吗？

常量池放在堆中，是为了解决之前放在方法区时，由于常量池空间大小有限，存储对象过多导致内存溢出问题。也会存在垃圾回收，但与堆的垃圾回收不一样，这里不会进入老年代，而是直接回收。

2019-05-25

... 64

Eric

当类或接口创建时会根据 class文件里面的常量池表构造运行时常量池 可以认为在调用new String("abc")很早很早之前 运行时常量池已经有了abc这个字符串常量吗？ 这里ldc只是把对应的字符串常量的reference类型的的数据压栈 作为后面的invokespecial指令来调用构造方法的第一个参数？ 理解为ldc指令时在运行时常量池创建abc字符串常量是否合适？还是我的理解存在错误？

作者回复: 你好 Eric，从反编译文件可以看出，字符串"abc"在编译时，已在常量池中创建，这个没问题。如果是变量，这个需要在运行时在常量池中创建。

ldc是JVM的入栈指令，在这里会将常量池中的"abc"入栈，invokespecial是JVM调用构造函数的指令，此时会调用String的构造函数，“abc”作为参数。之后通过出栈指令返回引用给\$1。

大概就是这个流程，有疑问欢迎一起探讨。

2019-05-25

... 1

Stalary

老师，请问indexOf替换Split意思是先indexOf找到切割点再用subString吗？

作者回复: 你好 Stalary, 对的，虽然这种方式比split方法多一些代码，但在某些情况下性能要比split方法好

2019-05-25

... 64

WL

请问老师啥叫正则表达式的回溯问题，为啥会产生回溯问题，在网上找了半天资料没太看懂

作者回复: 你好 WL，下一讲我会详细讲正则表达式的回溯。

2019-05-25

... 1

小辉辉

false false true，第一个和第二个为false是因为比的是堆和常量池两个不同的对象，第三个为true因为比的是常量池中同一个对象。

作者回复: 理解到位！

2019-05-25

... 1

-  胡晓
老师把基础搞一遍再讲工具技巧是蛮好的
2019-05-25
-  木鱼水心
老师，String并不是真的不可变的，我们可以通过反射过去char数组，通过改变里面的元素来改变对象
2019-07-07
-  ZOU志伟
老师，有点不明白，intern()方法无论调用不调用，常量池都是会创建字符串的，hashTable都是会增大。何来谨慎使用？
作者回复：如果是字符串常量，是在常量池中创建字符串。假如是应用服务运行期间，通过数据查询出的字符串，则不会在常量池中创建。
2019-07-05
-  郁陌陵
老师，我想问一下，如开题所示代码，在常量池中字符串“abc”会有一个还是两个，如果是两个的话，怎么理解“因为常量池的实现是类似于一个HashTable 的实现方式”这句话呢
作者回复：一个哦
2019-07-05
-  李
2. 如何使用 String.intern 节省内存
这个小段最后一个图中的，第3个图什么意思，为什么不在常量池创建
作者回复：在运行时动态创建对象，是不会在常量池中创建。
2019-06-25
-  周彬
老师，在使用字符串比较时，有没有比字符串自带的包含方法更高效的比较方式，谢谢
2019-06-24
-  Geek_ebda96
老师，你好，你在文中所说的b对象将指向在常量池中存在引用对象，而原来b对象在堆内存中的空间将被垃圾回收，有两个疑问。第一个，我理解的一个java对象创建分三步，1.new一个对象，2.分配内存空间，3.把分配的空间指向池new的对象。java有没有针对new字符串对象进行优化，new之后直接先查找常量池有没有，如果有直接指向常量池，这样不是省去了分配空间的动作么，应该更快一些。第2个问题，java里的对象也就是全局变量本身也是占用了堆内存空间吧，因为你这边描述的是，b对象之前分配的内存被回收了，但b对象本身还存在，只是指向另外一个内存地址，垃圾回收的时候是不是把这些对象本身所占内存空间也要回收，对象所指向的内存空间也一起回收
作者回复：第一个问题，如果是String里面的构造函数是常量，这个优化是好的。但是如果是一个非常量，这个new动作就在先了，需要使用intern方法将字符串放到常量池，这个动作则是在后了。
第二个问题，b不是一个对象，而且一个引用，是放在了虚拟机栈中。
2019-06-11
-  伟
从 2G 降到几百兆，是不是意味着堆的常量池至少有几十上百兆？那这HashTable也不小吧？
作者回复：这个具体的性能状况没能了解到，这只是一个优化思路。
2019-06-10
-  cricket1981
请教如何对String类型的值对象加synchronized同步块？有人说synchronized(string.intern())方式不好，容易让对象GC不了，还可能造成dead lock，老师怎么看？标准做法是什么？谢谢！
2019-06-08
-  西兹兹
Zend同学提的 复制 问题，希望修改文稿为引用赋值才正确。
一开始我也纠结这点，看了Zend跟我提问内容一样
编辑回复：已修正～感谢这位细心的同学！
2019-06-08

 manis

老湿，那个图上最后一种情况是说运行时创建String不使用常量池的空间吗？
表示怀疑

作者回复：对的，运行时动态创建的字符串对象只有通过intern才会进入常量池。如果还有疑问，
可以复习下常量池的作用。

2019-06-05

 中年油腻大猪蹄子

String.indexOf()怎么切分字符串？求解

作者回复：需要结合substring方法一起用。

2019-06-05

 西门吹水之城

老师您好，看下面的留言，您看我这这么理解对吗？

```
String b=new String("abc");
for(int i=0;i<10;i++) {
    String c=new String(i+"");
}
```

上面的代码中，b和c是不同的，b在编译的时候会将abc放入常量池中，b引用的堆内存，堆内存引用常量池。c在编译时候没有字符串，在运行的时候，会直接存入内存中，不会将字符串放入常量池。这样解释可以吗？

作者回复：b是在类加载时，放入到常量池中。其他地方理解没问题。

2019-06-04

 summer

所以，一个字符串的值，如“abc”。不管是new出来的也好，还是直接 = “abc”的也罢。都会的把“ab
c”放到常量池中，两个想了解的地方。(1) 如果 stringBuffer来连接如：stringbuffer("a").append
(“b”).append(“c”).toString();这种情况下，是不是会分别创建 a, b, c三个字符串到常量池中呢？

(2) : 如要所有的值都是放在常量池中，那new String(“abc”)与 =“abc”这种，在程序里，是不是 =“a
bc”的性能更好些，毕竟new String(“abc”)多了从常量池中copy的过程。

作者回复：对的，如果只是常量，一般就是常量这种写法。

2019-06-03

 哇

老师，现在还有需要在变量用完后，手动设置变量为null的场景吗？

作者回复：不需要手动设置为null，虚拟机会帮我们自动回收。除非对象占用内存大或方法的栈帧
长时间不能被回收的情况下，我们手动设置对象为null，能提高回收效率。

2019-06-01

 Pason

在字符串变量中，对象是会创建在堆内存中，同时也会在常量池中创建一个字符串对象，意思是只要创
建了字符串变量，就会在常量池中有。那么不是和intern方法矛盾了吗，如果调用 intern 方法，会去查
看字符串常量池中是否有等于该对象的字符串，既然前面已经说了创建变量会在常量池中创建，那么int
ern方法的时候还有必要判断吗？

作者回复：你好 Pason，如果是new String(“abc”)这种方式，就会在编译加载类时在常量池中创
建“abc”，在new时，在堆中创建String对象。但在程序运行时动态创建字符串时，只会在堆中创
建字符串对象，这个时候就用到intern，就会去常量池创建对象了。

这样解释能否理解呢？

2019-05-31

 zjg

为什么直接拼接快呢？

```
public final static String generateCommentAndTopicDailyTaskTimeskey(String userid) {
    StringBuilder sb = new StringBuilder();
    sb.append("daily_comment_or_topic_times");
    sb.append("_");
    sb.append(userid);
    return sb.toString();
}
```

```
public final static String generateCommentAndTopicDailyTaskTimeskey1(String userid) {
    return "daily_comment_or_topic_times" + "_" + userid;
}
```

```
public static void main(String[] args) {
    int len = 1000000;
    long begin = System.currentTimeMillis();
```

```
for(int i=0;i<len;i++) {  
    generateCommentAndTopicDailyTaskTimeskey("12");  
}  
long end = System.currentTimeMillis();  
System.out.println(end-begin);  
  
long begin1 = System.currentTimeMillis();  
for(int i=0;i<len;i++) {  
    generateCommentAndTopicDailyTaskTimeskey("12");  
}  
long end1 = System.currentTimeMillis();  
System.out.println(end1-begin1);  
}
```

作者回复: 建议分开运行。这个比较应该没有很大差异。本身编译器会优化+。

2019-05-29

张华清

老师好，之前我的理解是对于String对象的创建：String a="a" 这种编译时期创建的对象是放入常量池，然后返回常量池对象引用。对于String b =new String("b")这种运行时期创建是放入堆内存的然后返回堆内存对象的引用，不会放入常量池，只有在调用intern方法的时候才会把放入常量池。所以对老师的这句话有点疑惑：“在一开始创建 a 变量时，会在堆内存中创建一个对象，同时会在在加载类时，在常量池中创建一个字符串对象”

作者回复: 你好，你理解后半段有一点问题。new String()是在执行代码的时候在堆中创建的对象，但new String("b")中的“b”，是在代码编译时，生成了字面量，并且在类加载时，放到了常量池中。在new String之时，会调用String的构造函数引用常量池中的字符串。

2019-05-29

星期八

老师，messageInfo.getCity().intern()这个例子中，一开始的对象是指哪个对象呢？

作者回复: 你好，这个指的是运行时在堆内存中创建的对象。

2019-05-29

半清醒

这个专栏的老师是我在极客时间上看到最负责的老师了，每个有疑惑的提问，老师都是认真回答了，不是说别的老师做的不好，而且您做的更好!!另外通过这篇文章也算更加理解String这个类了。

作者回复: 互联网教学应该是这样的，达到一对一的教学效果，大家在极客时间一起学习成长。

2019-05-28

我期待的e-mail

老师你好，堆对象在没有可达引用的时候在下一次gc就会被回收，那常量池是什么时候回收的呢？

作者回复: 在没有引用时，常量池中的对象也会被回收。

2019-05-28

胖姐

实际操作字符的时候还真没怎么注意过！因为日常的开发过程中，都是先测试一下能用就直接过去了！以后需要注意了！

作者回复: 是的，特别用到复杂的正则表达式时，要多留意，可以在校验回溯的网站校验下表达式。

2019-05-27

safe

老师，我想问一下，如果intern方法是在常量池中寻找是否存在的字符串，如果在多线程的情况下，会不会有资源竞争问题

作者回复: 你好 safe，这个常量是不可变的，所以不存在锁资源的问题，不会存在竞争。

2019-05-27

码农 布莱恩特

老师，有个问题请教下，数据库机器内存使用率一直超过80%，我该如何定位和优化

作者回复: 你好 布莱恩特，可以查看下是否有运行的sql阻塞了，可以通过在mysql中执行 show full processlist查看。

也有可能mysql配置缓存过大，检查下mysql的缓存配置。

2019-05-27

Alex

温故知新，特别好

2019-05-27

Janita

回溯问题是什么意思

编辑回复: 同学你好，明天更新的04讲就会讲到这个问题了。

2019-05-27

每天晒白牙

false
false
true

编辑回复: 点赞！

2019-05-27

胜

老师我今天遇到一个问题，请教一下使用fast json的JSONObject.parseObject(ss,parameterizedType)时产生过多IdentityHashMap.Entry导致内存泄露，请问该怎么解决？

作者回复: 看了下Github上也有人遇到跟你一样类似的问题，git上给出的建议是更新到最新版本。

2019-05-26

胖

会在常量池存储字符串对象的时机
1. 调用String的intern;
2. 声明字符串常量;
3. 字符串直接常量相加，例如: String c = "a" + "b"; 会在常量池保存"ab"

作者回复: 再补充一种，在new String("ab")时，也会创建一份在常量池，堆内存中对象会再引用常量池对象

2019-05-26

爪哇夜未眠

老师好，文章提到“使用 intern 方法需要注意的一点是，一定要结合实际场景...HashTable 存储的数据越大，遍历的时间复杂度就会增加...”
那建议多大的量可以使用呢？像Twitter那种，几十G压缩到几百兆的量，算大吗？会有问题吗？

作者回复: 你好，建议不超过百兆，这个可以根据具体环境和场景进行测试。

2019-05-26

some_bird

关于字符串: String str2= new String("abc") 中变量str2在线段指向了堆中一个存储了内容为"abc"的对象，还是指向了堆中存放地址（指向字符串常量池中"abc"的地址）的对象

作者回复: 你好 some_bird，指向队中创建的内存对象，我们可以通过反编译字节码文件，可以发现在创建完对象，通过JVM返回指令返回的是在内存中创建的对象引用。

2019-05-25

六六六

str1,str3都指向的是常量池，str2指向的是堆，所以答案是false, false, true。
想请教老师一个关于机器load的问题，我们有个应用的线上机器load有时会忽然飙升一下，然后又很快降下来，基本可以排除是流量问题，我现在的想法是通过定时脚本来不断检查load，当load升高之后，打印jstack信息。想问下老师还有没有其他的排查方案或者工具呢！谢谢老师

2019-05-25

余火

String s1="123";字符串"123"已经被放入常量池，那要定义一个和s1一样的，用String s2=s1;不就好了，为什么还要用intern方法。推特的那个优化也一样，setCity(messageInfo.getCity())就好了，为什么要用intern呢，这里有些不懂，请老师指教一下，多谢。

作者回复：你好余火，我们在类对象中的String属性是在堆内存中开辟的地址空间，所以如果不使用intern方法，新生成的对象中的String字段的属性是引用堆内存的地址，而堆内存中创建的字段即使value值一样也会开辟新地址空间。如果使用intern方法，则会引用常量池中的引用，而如果常量池中如果存在字符串对象，则复用，不会再创建了。

2019-05-25

胡小榕

```
String s = new String("abc").intern();
```

"堆内存中原有的对象由于没有引用指向它，将会通过垃圾回收器回收。"这个如何理解呢？是说s指向了"abc"的引用，而new String("abc")这个对象会被回收吗？请老师讲讲

作者回复：我们知道，在堆创建String对象后，如果没有intern方法，会将引用指向堆中的对象。调用intern方法之后，会将引用指向常量池中的字符串对象，此时堆中的对象则失去了引用。JVM的垃圾回收将没有被引用的对象回收。

不知道你是否理解了，有问题保持留言。

2019-05-25

DemonLee

即使使用 + 号作为字符串的拼接，也一样可以被编译器优化成 StringBuilder 的方式。

老师好，这个是从Java8才开始支持的吧，我的问题是：常量池在内存里面的位置与堆内存有啥区别呀

作者回复：编译优化在Java7就实现了。常量池在JDK1.7之前是放在非堆内存区中，且空间大小有限，在1.7以后放在了堆内存中，区别就是逻辑空间隔离与其他堆内存，垃圾回收也是不同于堆中的垃圾回收。

2019-05-25