

[云计算技术认知度大调查](#)

[开源虚拟化及云计算资料下载](#)

[辟虚拟化新天地 进思杰技术社区](#)

[立即注册2010甲骨文全球大会](#)

开发频道

[首页](#) | [Java](#) | [.NET](#) | [Web](#) | [XML](#) | [语言工具](#) | [测试](#) | [游戏](#) | [移动](#) | [架构](#) | [项目管理](#) | [全部文章](#)

您所在的位置：[首页](#) > [开发](#) > [Java](#) > [深入JVM](#) >

深入Java核心 Java内存分配原理精讲(1)

<http://developer.51cto.com> 2010-09-09 10:09 zl198751 JavaEye博客 [我要评论\(2\)](#)

栈、堆、常量池虽同属Java内存分配时操作的区域，但其适用范围和功用却大不相同。本文将深入Java核心，详细讲解Java内存分配方面的知识。

Java内存分配与管理是Java的核心技术之一，之前我们曾介绍过[Java的内存管理与内存泄露](#)以及[Java垃圾回收](#)方面的知识，今天我们再次深入Java核心，详细介绍一下Java在内存分配方面的知识。一般Java在内存分配时会涉及到以下区域：

寄存器：我们在程序中无法控制

栈：存放基本类型的数据和对象的引用，但对象本身不存放在栈中，而是存放在堆中

堆：存放用new产生的数据

热点 国内十大网站使用的Web服务器



我们通过Firebug查看http

返回头，得知国内十大网

站所用Web服务器

常量池：存放常量

非RAM存储：硬盘等永久存储空间

Java内存分配中的栈

在函数中定义的一些基本类型的变量数据和对象的引用变量都在函数的栈内存中分配。

当在一段代码块定义一个变量时，Java就在栈中 为这个变量分配内存空间，当该变量退出该作用域后，Java会自动释放掉为该变量所分配的内存空间，该内存空间可以立即被另作他用。

Java内存分配中的堆

堆内存用来存放由new创建的对象和数组。 在堆中分配的内存，由Java虚拟机的自动垃圾回收器来管理。

在堆中产生了一个数组或对象后，还可以在栈中定义一个特殊的变量，让栈中这个变量的取值等于数组或对象在堆内存中的首地址，栈中的这个变量就成了数组或对象的引用变量。 引用变量就相当于为数组或对象起的一个名称，以后就可以在程序中使用栈中的引用变量来访问堆中的数组或对象。引用变量就相当于为数组或者对象起的一个名称。

引用变量是普通的变量，定义时在栈中分配，引用变量在程序运行到其作用域之外后被释放。而数组和对象本身在堆中分配，即使程序运行到使用 new 产生数组或者对象的语句所在的代码块之外，数组和对象本身占据的内存不会被释放，数组和对象在没有引用变量指向它的时候，才变为垃圾，不能在堆中使用，但仍然占据内存空间不放，在随后的一个不确定的时间被垃圾回收器收走（释放掉）。这也是Java比较占内存的原因。

实际上，**栈中的变量指向堆内存中的变量，这就是Java中的指针！**

TOP10

24小时

本周

本月

- 01 Eclipse插件大全 挑选最牛的TOP30
- 02 JDK最新版本下载及JDK安装与配置
- 03 2010年9月编程语言排行榜：Perl 不走..
- 04 一些应该熟记于心的jQuery函数和技巧
- 05 多图详解教程：Eclipse 3.6连接Tomcat..
- 06 Java EE开发四大常用框架
- 07 国内十大网站使用的Web服务器调查
- 08 专访人人网黄晶：SNS网站后台架构探秘
- 09 Eclipse 3.6新特性与应用详解
- 10 MooTools团队成员：我们为何强于jQuery

TAG 热点标签

移动开发 云计算 ARP攻防 思科培训



订阅51CTO 邮刊

[点击这里查看样刊](#)

全站热点

常量池 (constant pool)

常量池指的是在编译期被确定，并被保存在已编译的.class文件中的一些数据。除了包含代码中所定义的各种基本类型（如int、long等等）和对象型（如String及数组）的常量值(final)还包含一些以文本形式出现的符号引用，比如：

类和接口的全限定名；

字段的名称和描述符；

方法和名称和描述符。

虚拟机必须为每个被装载的类型维护一个常量池。常量池就是该类型所用到的常量的一个有序集和，包括直接常量（string,integer和 floating point常量）和对其他类型，字段和方法的符号引用。

对于String常量，它的值是在常量池中的。而JVM中的常量池在内存当中是以表的形式存在的，对于String类型，有一张固定长度的CONSTANT_String_info表用来存储文字字符串值，注意：该表只存储文字字符串值，不存储符号引用。说到这里，对常量池中的字符串值的存储位置应该有一个比较明了的理解了。

在程序执行的时候,常量池 会储存在Method Area,而不是堆中。

堆与栈

Java的堆是一个运行时数据区,类的(对象从中分配空间。这些对象通过new、newarray、anewarray和multianewarray等指令建立，它们不需要程序代码来显式的释放。堆是由垃圾回收来负责的，堆的优势是可以动态地分配内存大小，生存期也不必事先告诉编译器，因为它是在运行时动态分配内存的，Java的垃圾收集器会自动收走这些不再使用的数据。但缺点是，由于要在运行时动态 分配内存，存取速度较慢。

- [新IE漏洞影响范围有限 微软将发布相关补丁](#)
- [Linux揭开USB主机设计的神秘面纱](#)
- [平凡黑客讲述精彩人生（一）旁观第六代网..](#)
- [套接字编程中的5个隐患](#)
- [Oracle数据库的空间管理技巧](#)
- [网络布线中常见的问题](#)
- [Exchange故障排错图解教程](#)
- [Oracle收购BEA后的中间件以及SOA产品线战..](#)
- [浅谈服务器的可用性](#)
- [光纤接续方法及操作步骤详解](#)

栈的优势是，存取速度比堆要快，仅次于寄存器，栈数据可以共享。但缺点是，存在栈中的数据大小与生存期必须是确定的，缺乏灵活性。栈中主要存放一些基本类型的变量数据（int, short, long, byte, float, double, boolean, char）和对象句柄(引用)。

🔗 栈有一个很重要的特殊性，就是存在栈中的数据可以共享。假设我们同时定义：

```
1. int a = 3;
2. int b = 3;
```

编译器先处理int a = 3；首先它会在栈中创建一个变量为a的引用，然后查找栈中是否有3这个值，如果没找到，就将3存放进来，然后将a指向3。接着处理int b = 3；在创建完b的引用变量后，因为在栈中已经有3这个值，便将b直接指向3。这样，就出现了a与b同时均指向3的情况。

这时，如果再令 a=4；那么编译器会重新搜索栈中是否有4值，如果没有，则将4存放进来，并令a指向4；如果已经有了，则直接将a指向这个地址。因此a值的改变不会影响到b的值。

要注意这种数据的共享与两个对象的引用同时指向一个对象的这种共享是不同的，因为这种情况a的修改并不会影响到b, 它是由编译器完成的，它有利于节省空间。而一个对象引用变量修改了这个对象的内部状态，会影响到另一个对象引用变量。

共3页: 1 [\[2\]](#) [\[3\]](#) [下一页](#)

【内容导航】

第 1 页：[Java中的内存分配](#)

第 2 页：[示例与分析](#)

第 3 页：[String常量池问题的几个例子](#)

原文：[深入Java核心 Java内存分配原理精讲](#)

标 签：[Java内存分配](#)

[上一篇：JVM上的动态语言 各大巨头的新宠](#) [下一篇：术语汇编 Java虚拟机概述](#)

网友评论

查看所有评论 (2)

paolo

发表时间:09-17 20:23

很好，值得学习

yangliu9420

发表时间:09-11 12:23

讲的不错，赞一个先，，

发表评论

发表人：

网友您好，十一长假期间，51CTO的评论功能暂时关闭，给您带来的不便还望谅解，感谢您的支持！

▣ 读 书

- Ext JS高级程序设计
- ASP.NET开发实战宝典
- 架构之美
- 信息存储与管理：数字信息的存储、管理和保护
- 谋断互联网职业：网络建设与营销必备攻略

▣ 论 坛

- 语音网关配置,谁帮我写下注释,每句..
- 系统错误请帮忙解决
- Oracle 10G r2 training course
- explorer.exe占用CPU100%
- 华为认证含金量！

▣ 博 客

- 关于黑盒测试
- 如何卸载有密码保护的symantec客户端
- JAVA 取得本机mac地址的方法
- 网工实用命令总结
- IE7.0默认主页修改

▣ 下 载

- 雅虎公司C#笔试题
- 数据结构与算法分析(Java版)
- 数据结构考研东北大学pdf
- 五子棋源码c++API
- 网络工程师历年试题答案电子书

Copyright©2005-2010 [51CTO.COM](#) 版权所有 未经许可 请勿转载

[云计算技术认知度大调查](#)

[开源虚拟化及云计算资料下载](#)

[辟虚拟化新天地 进思杰技术社区](#)

[JavaOne和甲骨文开发者大会](#)

开发频道

[首页](#) | [Java](#) | [.NET](#) | [Web](#) | [XML](#) | [语言工具](#) | [测试](#) | [游戏](#) | [移动](#) | [架构](#) | [项目管理](#) | [全部文章](#)

您所在的位置：[首页](#) > [开发](#) > [Java](#) > [深入JVM](#) >

深入Java核心 Java内存分配原理精讲(2)

<http://developer.51cto.com> 2010-09-09 10:09 zl198751 JavaEye博客 [我要评论\(2\)](#)

栈、堆、常量池虽同属Java内存分配时操作的区域，但其适用范围和功用却大不相同。本文将深入Java核心，详细讲解Java内存分配方面的知识。

String是一个特殊的包装类数据。可以用：

```
1. String str = new String("abc");
2. String str = "abc";
```

两种的形式来创建，第一种是用new()来新建对象的，它会在存放于堆中。每调用一次就会创建一个新的对象。而第二种是先在栈中创建一个对String类的对象引用变量str，然后通过符号引用去字符串常量池里找有没有"abc",如果没有，则将"abc"存放在字符串常量池，并令str指向"abc"，如果已经有"abc"则直接令str指向"abc"。

热点 国内十大网站使用的Web服务器



我们通过Firebug查看http

返回头，得知国内十大网

站所用Web服务器

比较类里面的数值是否相等时，用equals()方法；当测试两个包装类的引用是否指向同一个对象时，用==，下面用例子说明上面的理论。

```
1. String str1 = "abc";
2. String str2 = "abc";
3. System.out.println(str1==str2); //true
```

可以看出str1和str2是指向同一个对象的。

```
1. String str1 =new String ("abc");
2. String str2 =new String ("abc");
3. System.out.println(str1==str2); // false
```

用new的方式是生成不同的对象。每一次生成一个。

因此用第二种方式创建多个 " abc " 字符串,在内存中 其实只存在一个对象而已. 这种写法有利与节省内存空间. 同时它可以在一定程度上提高程序的运行速度，因为JVM会自动根据栈中数据的实际情况来决定是否有必要创建新对象。而对于String str = new String("abc");的代码，则一概在堆中创建新对象，而不管其字符串值是否相等，是否有必要创建新对象，从而加重了程序的负担。

另一方面, 要注意: 我们在使用诸如String str = "abc"; 的格式定义类时，总是想当然地认为，创建了String类的对象str。担心陷阱！对象可能并没有被创建！而可能只是指向一个先前已经创建的对象。

只有通过new()方法才能保证每次都创建一个新的对象。

由于String类的immutable性质，当String变量需要经常变换 其值时，应该考虑使用StringBuffer类，以提高程序效率。

TOP10

24小时 本周 本月

- 01 Eclipse插件大全 挑选最牛的TOP30
- 02 JDK最新版本下载及JDK安装与配置
- 03 2010年9月编程语言排行榜：Perl 不走..
- 05 一些应该熟记于心的jQuery函数和技巧
- 06 多图详解教程：Eclipse 3.6连接Tomcat..
- 07 Java EE开发四大常用框架
- 08 国内十大网站使用的Web服务器调查
- 09 专访人人网黄晶：SNS网站后台架构探秘
- 10 Eclipse 3.6新特性与应用详解
- MooTools团队成员：我们为何强于jQuery

TAG 热点标签

移动开发 云计算 ARP攻防 思科培训



订阅51CTO 邮刊

[点击这里查看样刊](#)

全站热点

1. 首先String不属于8种基本数据类型，String是一个对象。因为对象的默认值是null，所以String的默认值也是null；但它又是一种特殊的对象，有其它对象没有的一些特性。

2. new String()和new String(" ")都是申明一个新的空字符串，是空串不是null；

3. String str= " kvill " ；String str=new String (" kvill ")的区别

示例：

```
1. String s0="kvill";
2. String s1="kvill";
3. String s2="kv" + "ill";
4. System.out.println( s0==s1 );
5. System.out.println( s0==s2 );
```

结果为：

true

true

首先，我们要知结果为道Java 会确保一个字符串常量只有一个拷贝。

因为例子中的s0和s1中的 " kvill " 都是字符串常量，它们在编译期就被确定了，所以s0==s1为true；而 " kv " 和 " ill " 也都是字符串常量，当一个字符串由多个字符串常量连接而成时，它自己肯定也是字符串常量，所以s2也同样在编译期就被解析为一个字符串常量，所以s2也是常量池中 " kvill " 的一个引用。所以我们得出s0==s1==s2；用new String() 创建的字符串不是常量，不能在编译期就确定，所以new String() 创建的字符串不放入常量池中，它们有自己的地址空间。

▪ 组网答疑：路由器能替代防火墙吗？

▪ Win2003最新优化方法大全之一

▪ MIDP 2.0 Media API为游戏增加音效

▪ 平凡黑客讲述精彩人生（二）黑客游戏潜规..

▪ 活学活用 动态IP地址捕获及其应用

▪ 微软windows即插即用缓存溢出漏洞

▪ 赛门铁克Veritas NetBackup又暴漏洞

▪ 4天工作制逐步风行 谷歌150次班车接送员工

▪ 98和2K双系统并存常见问题解答

▪ 4月第3周安全回顾 恶意微处理器亮相 垃圾..

示例：

```
1. String s0="kvill";
2. String s1=new String("kvill");
3. String s2="kv" + new String("ill");
4. System.out.println( s0==s1 );
5. System.out.println( s0==s2 );
6. System.out.println( s1==s2 );
```

结果为：

false

false

false

例2中s0还是常量池中"kvill"的应用，s1因为无法在编译期确定，所以是运行时创建的新对象"kvill"的引用，s2因为有后半部分new String("ill")所以也无法在编译期确定，所以也是一个新创建对象"kvill"的应用;明白了这些也就知道为何得出此结果了。

4. String.intern()：

再补充介绍一点：存在于.class文件中的常量池，在运行期被JVM装载，并且可以扩充。String的intern()方法就是扩充常量池的一个方法；当一个String实例str调用intern()方法时，Java 查找常量池中是否有相同Unicode的字符串常量，如果有，则返回其的引用，如果没有，则在常量池中增加一个Unicode等于str的字符串并返回它的引用；看示例就清楚了

示例：

```
1. String s0= "kvill";
2. String s1=new String("kvill");
3. String s2=new String("kvill");
4. System.out.println( s0==s1 );
5. System.out.println( "*****" );
6. s1.intern();
7. s2=s2.intern(); //把常量池中"kvill"的引用赋给s2
8. System.out.println( s0==s1);
9. System.out.println( s0==s1.intern() );
10. System.out.println( s0==s2 );
```

结果为：

false

false //虽然执行了s1.intern(),但它的返回值没有赋给s1

true //说明s1.intern()返回的是常量池中"kvill"的引用

true

最后我再破除一个错误的理解：有人说，“使用 String.intern() 方法则可以将一个 String 类的保存到一个全局 String 表中，如果具有相同值的 Unicode 字符串已经在这个表中，那么该方法返回表中已有字符串的地址，如果在表中没有相同值的字符串，则将自己的地址注册到表中”如果我把他说的这个全局的 String 表理解为常量池的话，他的最后一句话，“如果在表中没有相同值的字符串，则将自己的地址注册到表中”是错的：

示例：

```
1. String s1=new String("kvill");
```

```

2.  String s2=s1.intern();
3.  System.out.println( s1==s1.intern() );
4.  System.out.println( s1+" "+s2 );
5.  System.out.println( s2==s1.intern() );

```

结果：

false

kvill kvill

true


在这个类中我们没有声名一个 "kvill" 常量，所以常量池中一开始是没有 "kvill" 的，当我们调用s1.intern()后就在常量池中新添加了一个 "kvill" 常量，原来的不在常量池中的 "kvill" 仍然存在，也就不是“将自己的地址注册到常量池中”了。

s1==s1.intern() 为false说明原来的 "kvill" 仍然存在；s2现在为常量池中 "kvill" 的地址，所以有s2==s1.intern()为true。

5. 关于equals()和==:

这个对于String简单来说就是比较两字符串的Unicode序列是否相当，如果相等返回true;而==是 比较两字符串的地址是否相同，也就是是否是同一个字符串的引用。

6. 关于String是不可变的

 这一说又要说很多，大家只要知道String的实例一旦生成就不会再改变了，比如说：String

str= " kv " + " ill " + " " + " ans ";就是有4个字符串常量，首先 " kv " 和 " ill " 生成了 " kvill " 存在内存中，然后 " kvill " 又和 " " 生成 " kvill " 存在内存中，最后又和生成了 " kvill ans ";并把这个字符

串的地址赋给了str,就是因为String的 " 不可变 " 产生了很多临时变量,这也就是为什么建议用

StringBuffer的原因了, 因为StringBuffer是可改变的。

下面是一些String相关的常见问题：

String中的final用法和理解

```
final StringBuffer a = new StringBuffer("111");
```

```
final StringBuffer b = new StringBuffer("222");
```

```
a=b;//此句编译不通过
```

```
final StringBuffer a = new StringBuffer("111");
```

```
a.append("222");// 编译通过
```

✧ 可见，final只对引用的"值"(即内存地址)有效，它迫使引用只能指向初始指向的那个对象，改变它的指向会导致编译期错误。至于它所指向的对象的变化，final是不负责的。

共3页: [上一页](#) [\[1\]](#) [2](#) [\[3\]](#) [下一页](#)

【内容导航】

第 1 页：[Java中的内存分配](#)

第 2 页：[示例与分析](#)

第 3 页：[String常量池问题的几个例子](#)

原文：[深入Java核心 Java内存分配原理精讲](#)

标签：[Java内存分配](#)

上一篇：[JVM上的动态语言 各大巨头的新宠](#) 下一篇：[术语汇编 Java虚拟机概述](#)

网友评论

[查看所有评论 \(2 \)](#)

paolo

发表时间:09-17 20:23

很好，值得学习

yangliu9420

发表时间:09-11 12:23

讲的不错，赞一个先，，

发表评论

发表人：

网友您好，十一长假期间，51CTO的评论功能暂时关闭，给您带来的不便还望谅解，感谢您的支持！

📖 读书

- Visual C++网络编程经典案例详解
- 深入浅出Ext JS（第2版）
- 计算机的心智：操作系统之哲学原理
- 写给大家看的Web设计书（第3版）
- IDA Pro代码破解揭秘

🗣️ 论坛

- 2005年上半年网络工程师上午试卷（..
- Windows网上邻居互访
- 2007网络工程师学习笔记
- 2007网络工程师学习笔记
- 如何在CMD下使用命令建立文件？？？

📝 博客

- UIT信息容灾概论（7）
- windows 2008 中的远程协助
- 算不上BUG
- 一个线程池(ThreadPool)的使用
- 如何学习linux编程

📄 下载

- 新版CCNA 640-802考试主要考点
- CCNA红头发笔记【全部】
- ccna资料
- TestKing CCNA 640-802最新版V9.0，..
- ccna 640-802试题

云计算技术认知度大调查
开源虚拟化及云计算资料下载
辟虚拟化新天地 进思杰技术社区
立即注册2010甲骨文全球大会

开发频道 首页 | Java | .NET | Web | XML | 语言工具 | 测试 | 游戏 | 移动 | 架构 | 项目管理 | 全部文章

您所在的位置：[首页](#) > [开发](#) > [Java](#) > [深入JVM](#) >

深入Java核心 Java内存分配原理精讲(3)

<http://developer.51cto.com> 2010-09-09 10:09 zl198751 JavaEye博客 [我要评论\(2\)](#)

栈、堆、常量池虽同属Java内存分配时操作的区域，但其适用范围和功用却大不相同。本文将深入Java核心，详细讲解Java内存分配方面的知识。

String常量池问题的几个例子

下面是几个常见例子的比较分析和理解：

```
1. String a = "a1";
2. String b = "a" + 1;
3. System.out.println((a == b)); //result = true
4. String a = "atrue";
5. String b = "a" + "true";
```

热点 国内十大网站使用的Web服务器



我们通过Firebug查看http
返回头，得知国内十大网
站所用Web服务器

TOP10

	24小时	本周	本月
01	Eclipse插件大全 挑选最牛的TOP30		
02	JDK最新版本下载及JDK安装与配置		
03	2010年9月编程语言排行榜：Perl 不走..		
04	一些应该熟记于心的jQuery函数和技巧		
05	多图详解教程：Eclipse 3.6连接Tomcat..		
06	Java EE开发四大常用框架		
07	国内十大网站使用的Web服务器调查		
08			
09			
10			

```

6. System.out.println((a == b)); //result = true
7. String a = "a3.4";
8. String b = "a" + 3.4;
9. System.out.println((a == b)); //result = true

```

分析：JVM对于字符串常量的"+"号连接，将程序编译期，JVM就将常量字符串的"+"连接优化为连接后的值，拿"a" + 1来说，经编译器优化后在class中就已经是a1。在编译期其字符串常量的值就确定下来，故上面程序最终的结果都为true。

```

1. String a = "ab";
2. String bb = "b";
3. String b = "a" + bb;
4. System.out.println((a == b)); //result = false

```

分析：JVM对于字符串引用，由于在字符串的"+"连接中，有字符串引用存在，而引用的值在程序编译期是无法确定的，即"a" + bb无法被编译器优化，只有在程序运行期来动态分配并将连接后的新地址赋给b。所以上面程序的结果也就为false。 2 -

```

1. String a = "ab";
2. final String bb = "b";
3. String b = "a" + bb;
4. System.out.println((a == b)); //result = true

```

分析：和[3]中唯一不同的是bb字符串加了final修饰，对于final修饰的变量，它在编译时被解析为常量值的一个本地拷贝存储到自己的常量池中或嵌入到它的字节码流中。所以此时的"a" + bb和"a" + "b"效果是一样的。故上面程序的结果为true。

专访人人网黄晶：SNS网站后台架构探秘

Eclipse 3.6新特性与应用详解

MooTools团队成员：我们为何强于jQuery

TAG 热点标签

移动开发 云计算 ARP攻防 思科培训



订阅51CTO 邮刊

[点击这里查看样刊](#)

全站热点

- 北方网通用户连续20分钟无法访问Google
- 用EJB 3.0简化企业Java开发(上)
- 最新互联网安全报告称 目标性攻击增多黑客..
- IPv6协议在各操作系统下的安装与配置
- 深入解析浪潮AS1000G2存储系统
- 国家经贸委国家重点企业信息化工程案例
- ORACLE神话的破灭
- 宏碁：Linux开源软件是超低价笔记本关键
- 三种流行防火墙配置方案分析与对比
- Linux终究会被Ubuntu毁了？


```
1. String a = "ab";
2. final String bb = getBB();
3. String b = "a" + bb;
4. System.out.println((a == b)); //result = false
5. private static String getBB() {
6.     return "b";
7. }
```

分析：JVM对于字符串引用bb，它的值在编译期无法确定，只有在程序运行期调用方法后，将方法的返回值和"a"来动态连接并分配地址为b，故上面 程序的结果为false。

通过上面4个例子可以得出得知：

```
String s = "a" + "b" + "c";
```

就等价于String s = "abc";

```
String a = "a";
```

```
String b = "b";
```

```
String c = "c";
```

```
String s = a + b + c;
```

这个就不一样了，最终结果等于：

```
1. StringBuffer temp = new StringBuffer();
2. temp.append(a).append(b).append(c);
```



```
3. String s = temp.toString();
```

由上面的分析结果，可就不难推断出String 采用连接运算符（+）效率低下原因分析，形如这样的代码：

```
1. public class Test {
2.     public static void main(String args[]) {
3.         String s = null;
4.         for(int i = 0; i < 100; i++) {
5.             s += "a";
6.         }
7.     }
8. }
```

每做一次 + 就产生个StringBuilder对象，然后append后就扔掉。下次循环再到达时重新产生个StringBuilder对象，然后 append 字符串，如此循环直至结束。如果我们直接采用 StringBuilder 对象进行 append 的话，我们可以节省 N - 1 次创建和销毁对象的时间。 所以对于在循环中要进行字符串连接的应用，一般都是用StringBuffer或StringBulider对象来进行 append操作。

String对象的intern方法理解和分析：

```
1. public class Test4 {
2.     private static String a = "ab";
3.     public static void main(String[] args){
4.         String s1 = "a";
5.         String s2 = "b";
```

```
6. String s = s1 + s2;  
7. System.out.println(s == a); //false  
8. System.out.println(s.intern() == a); //true  
9. }  
10. }
```

这里用到Java里面是一个常量池的问题。对于s1+s2操作，其实是在堆里面重新创建了一个新的对象,s保存的是这个新对象在堆空间的的内容，所以s与a的值是不相等的。而当调用s.intern()方法，却可以返回s在常量池中的地址值，因为a的值存储在常量池中，故s.intern和a的值相等。

总结

栈中用来存放一些原始数据类型的局部变量数据和对象的引用(String,数组,对象等等)但不存放对象内容

堆中存放使用new关键字创建的对象.

字符串是一个特殊包装类,其引用是存放在栈里的,而对象内容必须根据创建方式不同定(常量池和堆).
有的是编译期就已经创建好,存放在字符串常量池中,而有的是运行时才被创建.使用new关键字,存放在堆中。

【编辑推荐】

1. [深入Java底层：内存屏障与JVM并发详解](#)
2. [20个开发人员非常有用的Java功能代码](#)
3. [Java内存溢出的详细解决方案](#)
4. [深入理解Java多态性](#)
5. [Java虚拟机内部构成浅析](#)

【责任编辑：[red7](#) TEL：（010）68476606】

共3页: [上一页](#) [\[1\]](#) [\[2\]](#) 3

【内容导航】

第 1 页：[Java中的内存分配](#)

第 2 页：[示例与分析](#)

第 3 页：[String常量池问题的几个例子](#)

原文：[深入Java核心 Java内存分配原理精讲](#)

标 签：[Java内存分配](#)

上一篇：[JVM上的动态语言 各大巨头的新宠](#) 下一篇：[术语汇编 Java虚拟机概述](#)

网友评论

[查看所有评论 \(2 \)](#)

paolo

发表时间:09-17 20:23

很好，值得学习

yangliu9420

发表时间:09-11 12:23

讲的不错，赞一个先，，

发表评论

发表人：

网友您好，十一长假期间，51CTO的评论功能暂时关闭，给您带来的不便还望谅解，感谢您的支持！

▣ 读书

- Visual Basic开发技术大全
- Mac OS X 10.6 Snow Leopard操作系统快速入门
- 嵌入式设计及Linux驱动开发指南——基于ARM9处

▣ 论坛

- 通过IP安全策略能不能有效防御聚生..
- PIX简要配置及相关说明
- 开机一直“滴滴”的问题

<div><div>理..</div><div><div><div></div></div><div><div>- Linux系统管理与网络管理</div></div><div><div>- 重构（Ruby版）</div></div></div></div>	<div><div><div></div></div><div><div>Cisco IOS特性介绍</div></div><div><div>- C语言的发展问题</div></div></div>
---	--