海 子

不经历风雨，怎能见彩虹？做一个快乐的程序员。

联系方式：
邮箱：dolphin0520@163.com
微信：scu_dolphin0520

昵称：海 子
园龄：7年4个月
粉丝：5716
关注：6
+加关注

< 2018年8月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Java垃圾回收机制

Java垃圾回收机制

说到垃圾回收（Garbage Collection，GC），很多人就会自然而然地把它和Java联系起来。在Java中，程序员不需要去关心内存动态分配和垃圾回收的问题，这一切都交给了JVM来处理。顾名思义，垃圾回收就是释放垃圾占用的空间，那么在Java中，什么样的对象会被认定为“垃圾”？那么当一些对象被确定为垃圾之后，采用什么样的策略来进行回收（释放空间）？在目前的商业虚拟机中，有哪些典型的垃圾收集器？下面我们就来逐一探讨这些问题。以下是本文的目录大纲：

一.如何确定某个对象是“垃圾”？

二.典型的垃圾收集算法

三.典型的垃圾收集器

如果有不正之处，希望谅解和批评指正，不胜感激。

请尊重作者劳动成果，转载请标明原文链接：

http://www.cnblogs.com/dolphin0520/p/3783345.html

一.如何确定某个对象是“垃圾”？

最新随笔

1. 【置顶】博客恢复更新公告
2. 金山快盘+TortoiseSVN构建版本控制仓库
3. 在windows下安装配置Ulipad
4. Java并发编程：线程间协作的两种方式：wait、notify、notifyAll和Condition
5. JAVA多线程和并发基础面试问答（转载）
6. Java线程面试题 Top 50 （转载）
7. Java并发编程：Timer和TimerTask（转载）
8. Java并发编程：Callable、Future和FutureTask
9. Java并发编程：CountDownLatch、CyclicBarrier和Semaphore
10. Java并发编程：线程池的使用

随笔分类(194)

Android(14)

C/C++(30)

Java NIO(2)

Java并发编程(19)

在这一小节我们先了解一个最基本的问题：如果确定某个对象是“垃圾”？既然垃圾收集器的任务是回收垃圾对象所占的空间供新的对象使用，那么垃圾收集器如何确定某个对象是“垃圾”？—即通过什么方法判断一个对象可以被回收了。

在java中是通过引用来和对象进行关联的，也就是说如果要操作对象，必须通过引用来进行。那么很显然一个简单的办法就是通过引用计数来判断一个对象是否可以被回收。不失一般性，如果一个对象没有任何引用与之关联，则说明该对象基本不太可能在其他地方被使用到，那么这个对象就成为可被回收的对象了。这种方式成为引用计数法。

这种方式的特点是实现简单，而且效率较高，但是它无法解决循环引用的问题，因此在Java中并没有采用这种方式（Python采用的是引用计数法）。看下面这段代码：

```
1  public class Main {
2      public static void main(String[] args) {
3          MyObject object1 = new MyObject();
4          MyObject object2 = new MyObject();
5
6          object1.object = object2;
7          object2.object = object1;
8
9          object1 = null;
10         object2 = null;
11     }
12 }
13
14 class MyObject{
15     public Object object = null;
16 }
```

最后面两句将object1和object2赋值为null，也就是说object1和object2指向的对象已经不可能再被访问，但是由于它们互相引用对方，导致它们的引用计数都不为0，那么垃圾收集器就永远不会回收它们。

Java代码之美
Java基础(18)
Java集合
Java网络编程
Java虚拟机
JS/CSS/Jquery
Linux/Shell(4)
Mysql
OJ(14)
PHP
Python/Web框架(16)
Redis
Web开发(2)
XML(1)
计算机网络
计算机系统(3)
开发工具(7)
设计模式(1)
数据结构(11)
数据库(2)

为了解决这个问题，在Java中采取了 可达性分析法。该方法的基本思想是通过一系列的“GC Roots”对象作为起点进行搜索，如果在“GC Roots”和一个对象之间没有可达路径，则称该对象是不可达的，不过要注意的是被判定为不可达的对象不一定就会成为可回收对象。被判定为不可达的对象要成为可回收对象必须至少经历两次标记过程，如果在这两次标记过程中仍然没有逃脱成为可回收对象的可能性，则基本上就真的成为可回收对象了。

至于可达性分析法具体是如何操作的我暂时也没有看得很明白，如果有哪位朋友比较清楚的话请不吝指教。

下面来看个例子：

```
1 Object aobj = new Object ( ) ;
2 Object bobj = new Object ( ) ;
3 Object cobj = new Object ( ) ;
4 aobj = bobj;
5 aobj = cobj;
6 cobj = null;
7 aobj = null;
```

第几行有可能会使得某个对象成为可回收对象？第7行的代码会导致有对象会成为可回收对象。至于为什么留给读者自己思考。

再看一个例子：

```
1 String str = new String("hello");
2 SoftReference<String> sr = new SoftReference<String>(new String("java"));
3 WeakReference<String> wr = new WeakReference<String>(new String("world"));
```

这三句哪句会使得String对象成为可回收对象？第2句和第3句，第2句在内存不足的情况下会将String对象判定为可回收对象，第3句无论什么情况下String对象都会被判定为可回收对象。

最后总结一下平常遇到的比较常见的将对象判定为可回收对象的情况：

1）显示地将某个引用赋值为null或者将已经指向某个对象的引用指向新的对象，比如下面的代码：

```
1 Object obj = new Object();
```

数据挖掘(1)
算法(27)
无线传感器网络(1)
信息检索
业余娱乐(7)
转载(14)
自然语言处理

常用链接
C++ Reference
MSDN 主页
SOJ
北大OJ
并发编程网
杭电OJ

积分与排名
积分 - 439038
排名 - 366

最新评论

```
obj = null;
Object obj1 = new Object();
Object obj2 = new Object();
obj1 = obj2;
```

2）局部引用所指向的对象，比如下面这段代码：

```
void fun() {

.....
    for(int i=0;i<10;i++) {
        Object obj = new Object();
        System.out.println(obj.getClass());
    }
}
```

循环每执行完一次，生成的Object对象都会成为可回收的对象。

3）只有弱引用与其关联的对象，比如：

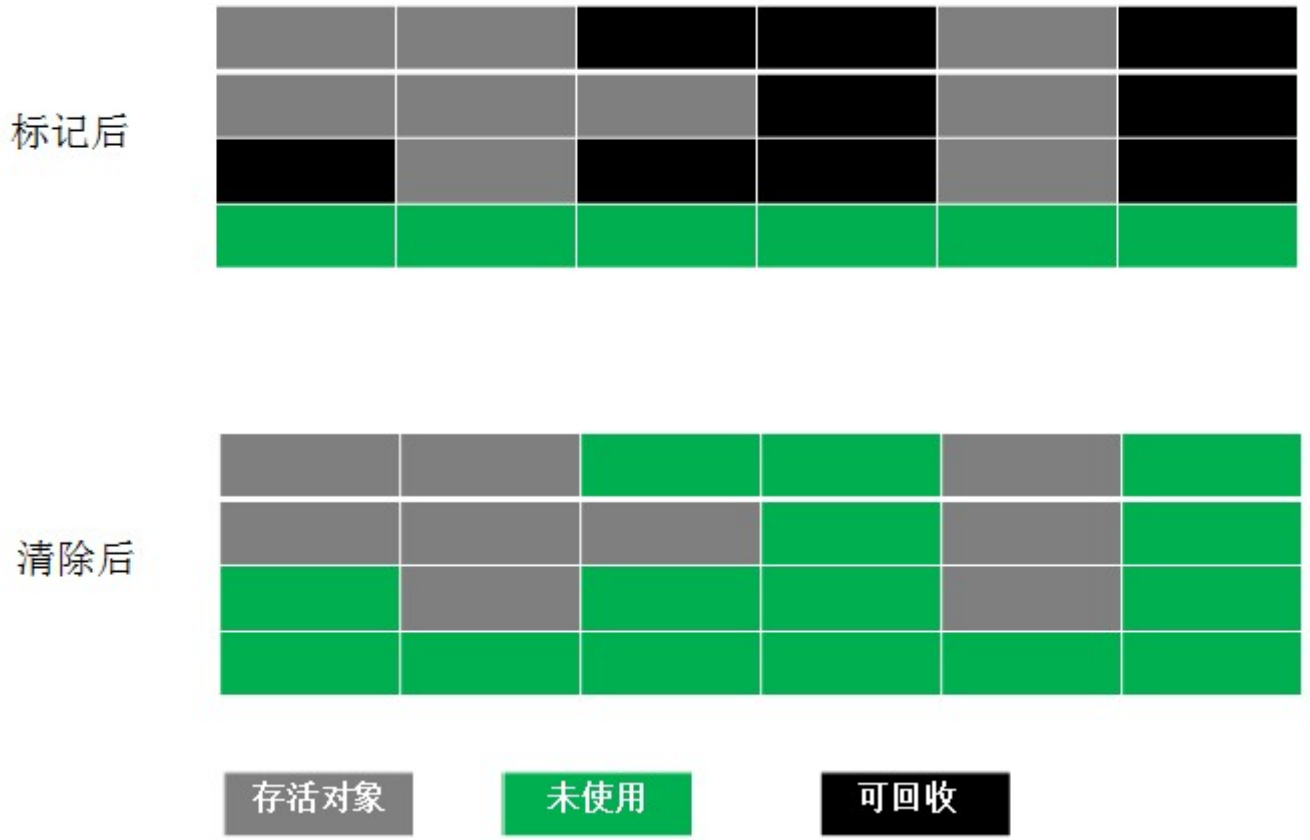
```
WeakReference<String> wr = new WeakReference<String>(new String("world"));
```

二.典型的垃圾收集算法

在确定了哪些垃圾可以被回收后，垃圾收集器要做的事情就是开始进行垃圾回收，但是这里面涉及到一个问题是：如何高效地进行垃圾回收。由于Java虚拟机规范并没有对如何实现垃圾收集器做出明确的规定，因此各个厂商的虚拟机可以采用不同的方式来实现垃圾收集器，所以在此只讨论几种常见的垃圾收集算法的核心思想。

1.Mark-Sweep（标记-清除）算法

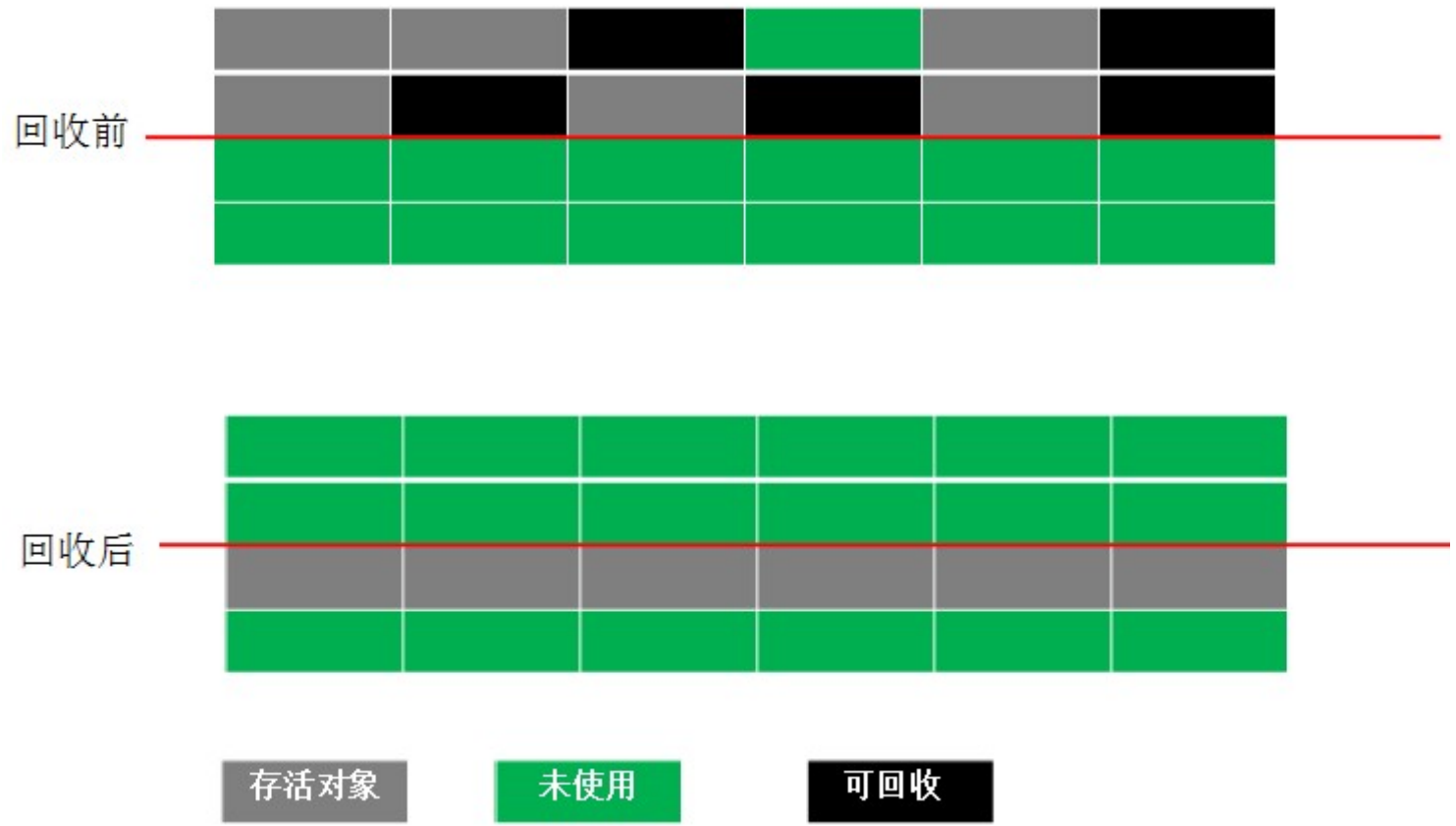
这是最基础的垃圾回收算法，之所以说它是最基础的是因为它最容易实现，思想也是最简单的。标记-清除算法分为两个阶段：标记阶段和清除阶段。标记阶段的任务是标记出所有需要被回收的对象，清除阶段就是回收被标记的对象所占用的空间。具体过程如下图所示：



从图中可以很容易看出标记-清除算法实现起来比较容易，但是有一个比较严重的问题就是容易产生内存碎片，碎片太多可能会导致后续过程中需要为大对象分配空间时无法找到足够的空间而提前触发新的一次垃圾收集动作。

2.Copying（复制）算法

为了解决Mark-Sweep算法的缺陷，Copying算法就被提了出来。它将可用内存按容量划分为大小相等的两块，每次只使用其中的一块。当这一块的内存用完了，就将还存活着的对象复制到另外一块上面，然后再把已使用的内存空间一次清理掉，这样一来就不容易出现内存碎片的问题。具体过程如下图所示：

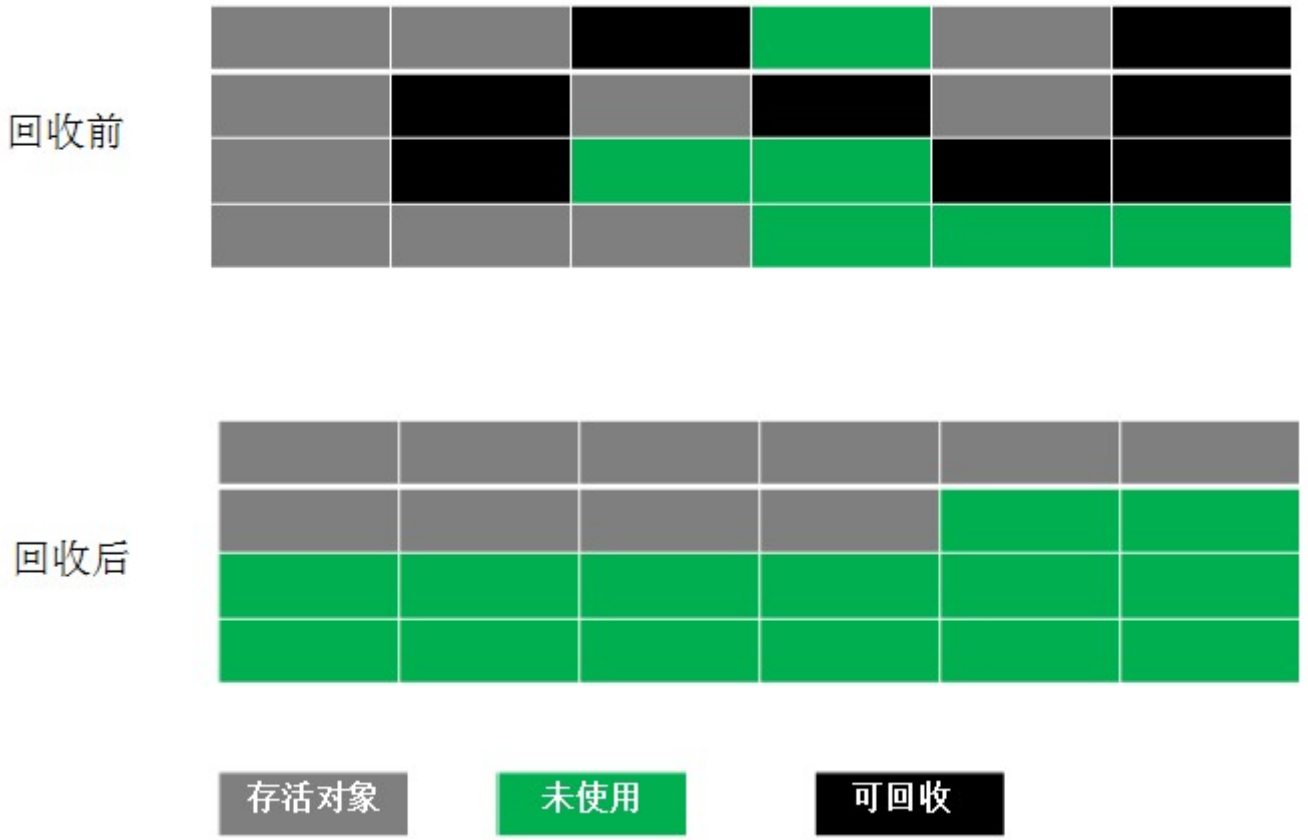


这种算法虽然实现简单，运行高效且不容易产生内存碎片，但是却对内存空间的使用做出了高昂的代价，因为能够使用的内存缩减到原来的一半。

很显然，Copying算法的效率跟存活对象的数目多少有很大的关系，如果存活对象很多，那么Copying算法的效率将会大大降低。

3.Mark-Compact（标记-整理）算法

为了解决Copying算法的缺陷，充分利用内存空间，提出了Mark-Compact算法。该算法标记阶段和Mark-Sweep一样，但是在完成标记之后，它不是直接清理可回收对象，而是将存活对象都向一端移动，然后清理掉端边界以外的内存。具体过程如下图所示：



4.Generational Collection（分代收集）算法

分代收集算法是目前大部分JVM的垃圾收集器采用的算法。它的核心思想是根据对象存活的生命周期将内存划分为若干个不同的区域。一般情况下将堆区划分为老年代（Tenured Generation）和新生代（Young Generation），老年代的特点是每次垃圾收集时只有少量对象需要被回收，而新生代的特点是每次垃圾回收时都有大量的对象需要被回收，那么就可以根据不同代的特点采取最适合的收集算法。

目前大部分垃圾收集器对于新生代都采取Copying算法，因为新生代中每次垃圾回收都要回收大部分对象，也就是说需要复制的操作次数较少，但是实际中并不是按照1：1的比例来划分新生代的空间的，一般来说是将新生代划分

为一块较大的Eden空间和两块较小的Survivor空间，每次使用Eden空间和其中的一块Survivor空间，当进行回收时，将Eden和Survivor中还存活的对象复制到另一块Survivor空间中，然后清理掉Eden和刚才使用过的Survivor空间。

而由于老年代的特点是每次回收都只回收少量对象，一般使用的是Mark-Compact算法。

注意，在堆区之外还有一个代就是永久代（Permanet Generation），它用来存储class类、常量、方法描述等。对永久代的回收主要回收两部分内容：废弃常量和无用的类。

三.典型的垃圾收集器

垃圾收集算法是 内存回收的理论基础，而垃圾收集器就是内存回收的具体实现。下面介绍一下HotSpot（JDK 7)虚拟机提供的几种垃圾收集器，用户可以根据自己的需求组合出各个年代使用的收集器。

1.Serial/Serial Old

Serial/Serial Old收集器是最基本最古老的收集器，它是一个单线程收集器，并且在它进行垃圾收集时，必须暂停所有用户线程。Serial收集器是针对新生代的收集器，采用的是Copying算法，Serial Old收集器是针对老年代的收集器，采用的是Mark-Compact算法。它的优点是实现简单高效，但是缺点是会给用户带来停顿。

2.ParNew

ParNew收集器是Serial收集器的多线程版本，使用多个线程进行垃圾收集。

3.Parallel Scavenge

Parallel Scavenge收集器是一个新生代的多线程收集器（并行收集器），它在回收期间不需要暂停其他用户线程，其采用的是Copying算法，该收集器与前两个收集器有所不同，它主要是为了达到一个可控的吞吐量。

4.Parallel Old

Parallel Old是Parallel Scavenge收集器的老年代版本（并行收集器），使用多线程和Mark-Compact算法。

5.CMS

CMS（Current Mark Sweep）收集器是一种以获取最短回收停顿时间为目标的收集器，它是一种并发收集器，采用的是Mark-Sweep算法。

6.G1

G1收集器是当今收集器技术发展最前沿的成果，它是一款面向服务端应用的收集器，它能充分利用多CPU、多核环境。因此它是一款并行与并发收集器，并且它能建立可预测的停顿时间模型。

下面补充一下关于内存分配方面的东西：

对象的内存分配，往大方向上讲就是在堆上分配，对象主要分配在新生代的Eden Space和From Space，少数情况下会直接分配在老年代。如果新生代的Eden Space和From Space的空间不足，则会发起一次GC，如果进行了GC之后，Eden Space和From Space能够容纳该对象就放在Eden Space和From Space。在GC的过程中，会将Eden Space和From Space中的存活对象移动到To Space，然后将Eden Space和From Space进行清理。如果在清理的过程中，To Space无法足够来存储某个对象，就会将该对象移动到老年代中。在进行了GC之后，使用的便是Eden space和To Space了，下次GC时会将存活对象复制到From Space，如此反复循环。当对象在Survivor区躲过一次GC的话，其对象年龄便会加1，默认情况下，如果对象年龄达到15岁，就会移动到老年代中。

一般来说，大对象会被直接分配到老年代，所谓的大对象是指需要大量连续存储空间的对象，最常见的一种大对象就是大数组，比如：

```
byte[] data = new byte[4*1024*1024]
```

这种一般会直接在老年代分配存储空间。

当然分配的规则并不是百分之百固定的，这要取决于当前使用的是哪种垃圾收集器组合和JVM的相关参数。

参考资料：

《深入理解Java虚拟机》

分类：Java基础

标签：Java

« 上一篇：java 字节流和字符流的区别 转载
» 下一篇：Java中的static关键字解析

posted @ 2014-06-18 15:28 海 子 阅读(42516) 评论(13) 编辑 收藏

评论列表

#1楼 2014-06-18 16:07 打一个情

写的挺好。

支持(0) 反对(0)

#2楼 2014-06-18 20:18 JeffWong

不错。自动垃圾回收算法看上去Java比C#复杂不少，C#采用的是分代收集算法，对大对象堆的分代处理Java可能更灵活一些，Framework4.5+对大对象的垃圾回收进行了改良。引用计数法除了Python再用，JavaScript高级程序设计里讲过，好像是IE6~8也是用的引用计数法，很容易引发内存泄露。

支持(0) 反对(0)

#3楼 2015-05-13 09:24 sstong123

思路很清晰，一目了然!

支持(0) 反对(0)

#4楼 2015-07-05 16:36 风吟无声

可达性分析不就是和图差不多吗？我觉得按照图理解就可以。

支持(0) 反对(0)

#5楼 2015-09-16 16:50 laudukang

思路清晰

支持(0) 反对(0)

#6楼 2015-09-20 21:46 -琥珀川-

马克

支持(0) 反对(0)

#7楼 2015-09-24 16:54 蓝雪精灵

回收对象说的是回收对象所占用的内存吧?

支持(2) 反对(0)

#8楼 2016-01-05 10:27 Mors

我想问下，那个你留个读者思考的，答案是第七行的代码，第四行时，aobj指向的对象不就已经不可达了吗?

支持(5) 反对(0)

#9楼 2016-03-23 19:40 秦康

http://stackoverflow.com/questions/6366211/what-are-the-roots

支持(0) 反对(0)

#10楼 2016-06-29 16:18 sonn

都是《深入理解java虚拟机》的内容，我觉得总结的不是特别好。没什么自己的思维，基本是原文。

支持(0) 反对(0)

#11楼 2017-05-25 01:11 fupeng

写的很好

支持(0) 反对(0)

#12楼 2017-08-31 12:30 稻草客666

虽是《深入理解java虚拟机》 总结的不错

支持(0) 反对(0)

#13楼 2017-12-13 19:38 做个有梦想的咸鱼

很赞

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。