**单例模式讨论篇：单例模式与垃圾回收**

 Jvm的垃圾回收机制到底会不会回收掉长时间不用的单例模式对象，这的确是一个比较有争议性的问题。将这一部分内容单独成篇的目的也是为了与广大博友广泛的讨论一下这个问题。为了能让更多的人看到这篇文章，请各位博友看完文章之后，点一下“顶”，让本篇文章排名尽量的靠前。笔者在此谢过。

讨论命题：当一个单例的对象长久不用时，会不会被jvm的垃圾收集机制回收。

        首先说一下为什么会产生这一疑问，笔者本人再此之前从来没有考虑过垃圾回收对单例模式的影响，直到去年读了一本书，《设计模式之禅》秦小波著。在书中提到**在j2ee应用中，jvm垃圾回收机制会把长久不用的单例类对象当作垃圾，并在cpu空闲的时候对其进行回收。**之前读过的几本设计模式的书，包括《java与模式》，书中都没有提到jvm垃圾回收机制对单例的影响。并且在工作过程中，也没有过单例对象被回收的经历，加上工作中很多前辈曾经告诫过笔者：尽量不要声明太多的静态属性，因为这些静态属性被加载后不会被释放。因此对jvm垃圾收集会回收单例对象这一说法持怀疑态度。渐渐地，发现在同事中和网上的技术人员中，对这一问题也基本上是鲜明的对立两派。那么到底jvm会不会回收长久不用的单例对象呢。

        对这一问题，笔者本人的观点是：**不会回收。**

下面给出本人的测试代码

**[java]** [view plain](http://blog.csdn.net/zhengzhb/article/details/7331354) [copy](http://blog.csdn.net/zhengzhb/article/details/7331354)

1. **class** Singleton {
2. **private** **byte**[] a = **new** **byte**[6\*1024\*1024];
3. **private** **static** Singleton singleton = **new** Singleton();
4. **private** Singleton(){}
6. **public** **static** Singleton getInstance(){
7. **return** singleton;
8. }
9. }
11. **class** Obj {
12. **private** **byte**[] a = **new** **byte**[3\*1024\*1024];
13. }
15. **public** **class** Client{
16. **public** **static** **void** main(String[] args) **throws** Exception{
17. Singleton.getInstance();
18. **while**(**true**){
19. **new** Obj();
20. }
21. }
22. }

        本段程序的目的是模拟j2ee容器，首先实例化单例类，这个单例类占6M内存，然后程序进入死循环，不断的创建对象，逼迫jvm进行垃圾回收，然后观察垃圾收集信息，如果进行垃圾收集后，内存仍然大于6M，则说明垃圾回收不会回收单例对象。

        运行本程序使用的虚拟机是hotspot虚拟机，也就是我们使用的最多的java官方提供的虚拟机，俗称jdk，版本是jdk1.6.0\_12

        运行时vm arguments参数为：-verbose:gc -Xms20M -Xmx20M，意思是每次jvm进行垃圾回收时显示内存信息，jvm的内存设为固定20M。

**运行结果：**

……

[Full GC **18566K->6278K**(20352K), 0.0101066 secs]

[GC **18567K->18566K**(20352K), 0.0001978 secs]

[Full **GC 18566K->6278K**(20352K), 0.0088229 secs]

……

        从运行结果中可以看到总有6M空间没有被收集。因此，笔者认为，至少在hotspot虚拟机中，垃圾回收是不会回收单例对象的。

        后来查阅了一些相关的资料，hotspot虚拟机的垃圾收集算法使用根搜索算法。这个算法的基本思路是：对任何“活”的对象，一定能最终追溯到其存活在堆栈或静态存储区之中的引用。通过一系列名为根（GC Roots）的引用作为起点，从这些根开始搜索，经过一系列的路径，如果可以到达java堆中的对象，那么这个对象就是“活”的，是不可回收的。可以作为根的对象有：

* 虚拟机栈（栈桢中的本地变量表）中的引用的对象。
* **方法区中的类静态属性引用的对象。**
* 方法区中的常量引用的对象。
* 本地方法栈中JNI的引用的对象。

        方法区是jvm的一块内存区域，用来存放类相关的信息。很明显，java中单例模式创建的对象被自己类中的静态属性所引用，符合第二条，因此，单例对象不会被jvm垃圾收集。

        虽然jvm堆中的单例对象不会被垃圾收集，但是单例类本身如果长时间不用会不会被收集呢？因为jvm对方法区也是有垃圾收集机制的。如果单例类被收集，那么堆中的对象就会失去到根的路径，必然会被垃圾收集掉。对此，笔者查阅了hotspot虚拟机对方法区的垃圾收集方法，jvm卸载类的判定条件如下：

* **该类所有的实例都已经被回收，也就是java堆中不存在该类的任何实例。**
* 加载该类的ClassLoader已经被回收。
* 该类对应的java.lang.Class对象没有任何地方被引用，无法在任何地方通过反射访问该类的方法。

        只有三个条件都满足，jvm才会在垃圾收集的时候卸载类。显然，单例的类不满足条件一，因此单例类也不会被卸载。也就是说，只要单例类中的静态引用指向jvm堆中的单例对象，那么单例类和单例对象都不会被垃圾收集，依据根搜索算法，对象是否会被垃圾收集与未被使用时间长短无关，仅仅在于这个对象是不是“活”的。假如一个对象长久未使用而被回收，那么收集算法应该是最近最长未使用算法，最近最长未使用算法一般用在操作系统的内外存交换中，如果用在虚拟机垃圾回收中，岂不是太不安全了？以上是笔者的观点。

        因此笔者的观点是：**在hotspot虚拟机1.6版本中，除非人为地断开单例中静态引用到单例对象的联接，否则jvm垃圾收集器是不会回收单例对象的。**