链滴

如何判断对象已死

作者：xixiaoming
原文链接：https://ld246.com/article/1501234062669
来源网站：链滴

已死的对象就是不可能被任何途径使用的对象，有以下几种方法判断一个对象是否已经死了：

引用计数

给对象添加一个引用计数器，每当有一个地方引用他，计算器就加1；当引用失效时，计数器减1；任时刻计数器为0的对象就是死的对象。

1. 这种方式被很多技术所采用，如FlashPlayer（AS3）、Python等，但是Java没有采用这种算法，因是它很难解决对象之间相互循环引用的问题，例如 ObjectA.param=ObjectB，ObjectB.param=OjectA，对象A和B相互引用但是除此之外他们再无任何其他引用，这样他们的引用计数都不为0，永远会被回收

根搜索

java采用的是根搜索算法，这个算法的基本思路是通过一系列名为“GC Roots”的对象作为起始点，这些节点开始向下搜索，搜索所经过的路径称为引用链，当GC Roots到一个对象不可达时，这个对就是不可用的

1. Java中可作为GC Roots的对象包括：

1）虚拟机栈中引用的对象

2）方法区中类静态属性和常量引用的对象

3）本地方法栈中Native方法引用的对象

何为引用（和ROOT有关系）

简单描述引用就是一块内存中存储的数值代表的是另外一块内存的起始地址，则成这块内存代表着一引用，这种描述很纯粹但是是不太正确的，可以作为理解时用

1. 引用分为强引用、软引用、弱引用、虚引用

1）强引用：如Object obj=new Object();只要强引用存在，GC肯定不会回收被引用的对象

2）软引用：非必需的对象，当系统要发生内存溢出之前，会把这些对象列入回收范围

3）弱引用：更加非必需的对象，弱引用关联的对象只能生存到下一次垃圾回收之前

4）虚引用：也叫幽灵引用或者幻影引用，它是最弱的一种引用，虚引用不会引用一个对象的生命周，也无法通过一个虚引用获取一个对象实例，只是虚引用的对象被回收时会有一个通知返回回去

两次标记——死刑

1. 一个对象在进行根搜索后发现没有与GC Roots相关联的引用链（和皇帝没有关系），那么它将被一次标记并且进行一次判断，判断该对象是否覆盖了finalize方法（被下了死刑），如果是的，则认有必要执行回收（死刑），如果不是则判断finalize方法是否已经被虚拟机调用过，如果已经调用过，则认为没必要执行了（因为已经执行过死刑了）

2. 如果一个对象被判定为执行死刑，那么这个对象将会被放置在一个名为F-Queue的队列中（关起来），这个队列会在另外一条线程中执行死刑，因为队列中的对象可能存在死循环等情况（危险分子）在GC主线程中执行可能导致整个GC崩溃

3. 稍后GC便会新开一条线程来执行死刑，它先对F-Queue中的对象进行第二次标记，这时如果队列有个对象与引用链上任何一个对象建立关联（有人给你喊刀下留人），那么在这次标记中它将会被移F-Queue，不然就死了

永久代就不会被杀吗

JVM规范中说过方法区（永久代）可以不实现垃圾回收，但是被上了免死金牌的永久代也是可以杀的只是杀的少些而已，不像新生代，一次垃圾收集就可以杀掉70%~95%的对象

1. 永久代回收只要针对两部分内容：废弃常量和无用的类

1）回收废弃常量例子：一个字符串“abc”已经进入了常量池中，但是当前系统没有任何一个String象叫做“abc”，而且如果必要的话（没内存啦），那么这个对象将会被请出常量池（朝廷没钱了，不给一些朝中无人的王爷发钱了，自己过日子去）

2）判断一个类似“无用的类”

·该类所有的实例都被杀掉

·加载该类的ClassLoader已经被杀掉

·该类的Class对象没有在任何地方被引用，没有在任何地方通过反射访问该类地方法

满足这三个条件也并不是一定要杀掉，详见类卸载，主要在经常要大量使用反射、CGLib、动态代理动态生成JSP、OSGI等场景需要配置下，以保证永久代不会溢出

一个例子

问题：下面代码中，第几行的哪个对象符合垃圾回收标准?

```
1 Object a = new Object();
2 Object b = new Object();
3 Object c = new Object();
4 a = b;
5 a = c;
6 c = null;
7 a = null;
```

答案：直到第7行时才有一个对象c符合了垃圾回收标准