链滴

Java 对象的绝地求生

作者：Gakkiyomi2019

原文链接：https://ld246.com/article/1603335640709

来源网站：链滴

一个Java对象的一生非常悲情，在需要时产生，在无用时消亡。当一个Java对象被可达性分析算法指GC Roots到这个对象不可达时就宣告了这个对象即将面临被消亡的处境。

那么死亡是注定的吗？幸运的是它有一次“自救”的机会。当对象进行可达性分析后发现没有与GC Roots相连的引用链，那么它将被标记一次，然后进行一次筛选，条件是此对象是否有必要执行**finalize**法，假设它这个对象没有重写**finalize**方法或者说这个方法已经调用过了，那么虚拟机就认为没有必执行了。

当然如果被判定为有必要执行，那么虚拟机会将这些需要执行的对象放置在一个名为**F-Queue**的队列中，然后一条由虚拟机建立的优先级低的线程去执行他们的finalize方法，但不保证一定会等待他们行结束。原因是如果某个对象的finalize方法执行缓慢，或者更加极端的发生了死循环，将导致整个统崩溃。

finalize方法是这些被即将消亡的对象唯一生存下来的机会，稍后收集器会对这个F-Queue中的对象行第二次的标记，如果成功自救，那么在这次标记时它将被移出“即将回收”的集合，如果没有成功救，那就在劫难逃，真的要被回收了。

如何自救?

自救的方法只有一个，那就是在唯一执行finalize方法的时候重新与引用链上的任何一个对象建立关，譬如将自己赋值给某个类变量或者对象的成员变量，他就可以存活。那么哪些对象可以成为拯救者?

这些对象也就是被称为GC Roots的对象，包括一下几种:

- 虚拟机栈（栈帧中的本地变量表）中引用的对象，譬如各个线程被调用的方法堆栈中使用到的参数局部变量，临时变量等;
- 方法区中类静态属性引用的对象;
- 方法区中常量引用的对象；譬如字符串常量池里的引用;
- 本地方法栈中JNI（即一般说的Native方法）引用的对象;

- JVM虚拟机内部的引用，如基本数据类型对应的Class对象，一些常驻的异常对象；（NPE,OutOfMoryError）,还有系统类加载器；
- 所有被同步锁(synchronized)持有的对象；

除了这些固定的GC Root集合外，还有根据内存区域不同，还可以有其他对象临时地加入，一起组成整的GC Roots集合。

代码演示 自救过程

```
/**
 * @author: fangcong
 * @description:
 * @create: Created by work on 2020-10-22 10:48
 **/
public class GcEscape {

    public static GcEscape SAVE_HOOK = null;

    public void isAlive(){
        System.out.println("yes,i am still alive :)");
    }

    @Override
    protected void finalize() throws Throwable {
        super.finalize();
        System.out.println("finalize method executed");
        GcEscape.SAVE_HOOK = this;  //建立GC Root 引用
    }

    public static void main(String[] args) throws Throwable {
        SAVE_HOOK = new GcEscape();

        SAVE_HOOK = null; //断开GC ROOT 引用 等待垃圾回收
        System.gc(); //手动执行垃圾回收
        //暂停0.5秒，等待线程执行finalize方法（此线程优先级低）
        Thread.sleep(500);
        if (SAVE_HOOK != null) {
            SAVE_HOOK.isAlive();
        } else {
            System.out.println("i am dead");
        }

        SAVE_HOOK = null; //断开GC ROOT 引用 等待垃圾回收
        System.gc(); //手动执行垃圾回收   //finalize方法只执行一次,此时不会执行

        if (SAVE_HOOK != null) {
            SAVE_HOOK.isAlive();
        } else {
            System.out.println("i am dead");
        }
    }
}
```

运行结果:

```
finalize method executed
yes,i am still alive :)
i am dead
```

生存是一切，这个对象做了它能做的。**但是在开发中并不推荐使用这个方法**，代价昂贵，不确定性大。