思考：哪些内存需要回收

什么时候回收

如何回收

判断一个对象是否存活

1. 引用计数算法

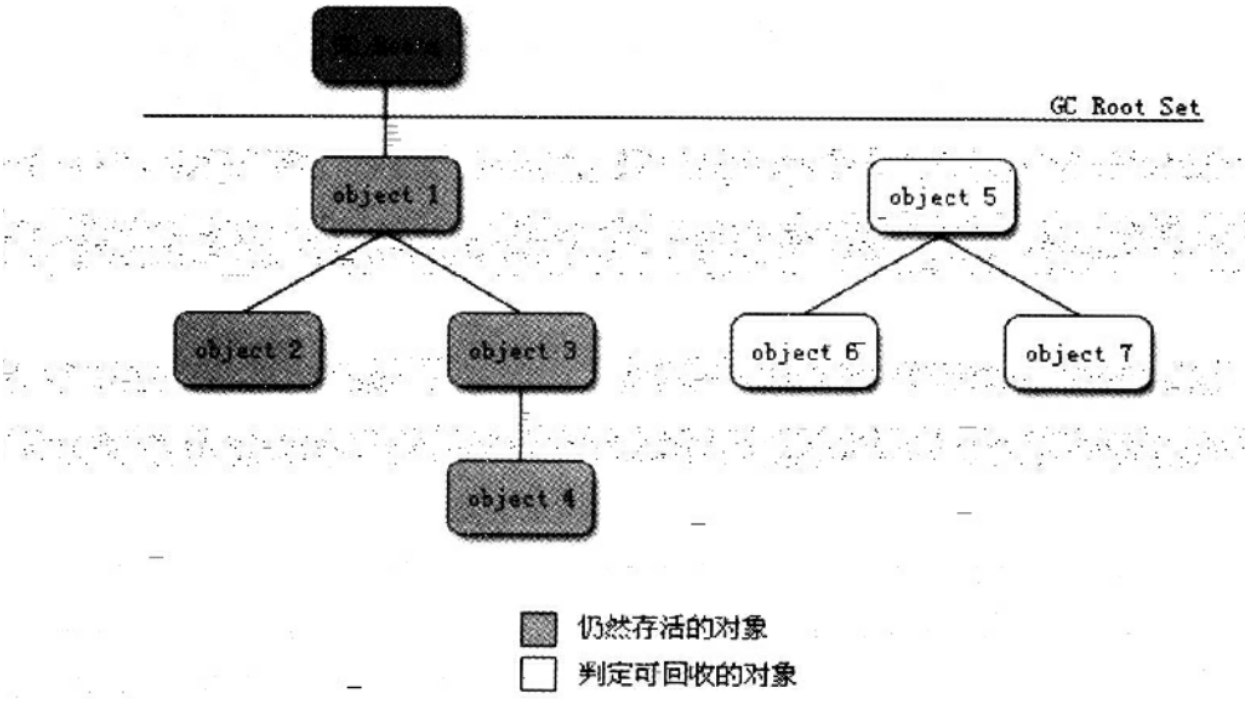
给对象添加一个引用计数器，每当一个地方引用它时，计数器加一，当引用失效时，计数器减一，任何时刻计数器为零时的对象就是不可能再被使用的，但是不能解决对象之间的循环引用问题

```
/**
 * testGC() 方法执行后，objA 和 objB 会不会被 GC 呢？
 * @author zzm
 */
public class ReferenceCountingGC {

    public Object instance = null;
```

```
private static final int _1MB = 1024 * 1024;

/**
 * 这个成员属性的唯一意义就是占点内存，以便能在GC日志中看清楚是否被回收过
 */
private byte[] bigSize = new byte[2 * _1MB];

public static void testGC() {
        ReferenceCountingGC objA = new ReferenceCountingGC();
        ReferenceCountingGC objB = new ReferenceCountingGC();
        objA.instance = objB;
        objB.instance = objA;

        objA = null;
        objB = null;

        //假设在这行发生GC，objA和objB是否能被回收？
        System.gc();
}
```

2.可达性分析算法

从一个叫做“GC ROOT”的对象作为起始点，从这个节点开始向下搜索，搜索所走过的路径称为引用链，当一个对象到GC ROOT没有任何引用链相连时，则此对象不可用



可作为GC ROOT的对象有下面几种

虚拟机栈中引用的对象

方法区中静态属性引用的对象

方法区中常量引用的对象

本地方法栈中JNI（native方法）引用的对象

3.在可达性分析算法中不可达的对象会经历两次标记过程;

在发现对象不可达后，进行第一次标记且进行一次筛选，筛选的条件是对象是否有必要执行finalize方法（若对象没有覆盖此方法或者此方法已被调用过，则没有必要执行，直接回收）

若判定为有必要执行此方法，，则对象被放入一个F-Queue队列中，并稍后由一个虚拟机自动建立、低优先级的Finalizer线程执行它（虚拟机触发这个方法，但不会承诺等待它运行结束，这样做避免在此方法中出现死循环）。稍后GC将此队列中的对象进行第二次小规模标记（若在finalize方法中将此自己（this）赋给某个类变量或自己的成员变量等），那么在第二次标记后将会移出队列，若没有移出，则被回收，但是这个方法只会执行一次