如果长生命周期的对象持有短生命周期的引用，就很可能会出现内存泄露。

```
public class Simple {
    Object object;
    public void method1(){
    object = new Object();
    //...其他代码
    }
}
```

//

这里的object实例，其实我们期望它只作用于method1()方法中，且其他地方不会再用到它，但是，当method1()方法执行完成后，object对象所分配的内存不会马上被认为是可以被释放的对象，只有在Simple类创建的对象被释放后才会被释放。解决方法就是将object作为method1()方法中的局部变量。将类的成员变量改写为方法内的局部变量。也可以这样手动设置为null：

```
public class Simple {
    Object object;
    public void method1(){
    object = new Object();
    //...其他代码
    object = null;
    }
}
```

//

这样，之前"new Object()"分配的内存，就可以被GC回收。在内存对象明明已经不需要的时候，还仍然保留着这块内存和它的访问方式（引用），这是所有语言都有可能会出现的内存泄漏方式。