# Java 对象的生命周期

在Java中，对象的生命周期包括以下几个阶段：

1. 创建阶段(Created)
2. 应用阶段(In Use)
3. 不可见阶段(Invisible)
4. 不可达阶段(Unreachable)
5. 收集阶段(Collected)
6. 终结阶段(Finalized)
7. 对象空间重分配阶段(De-allocated)

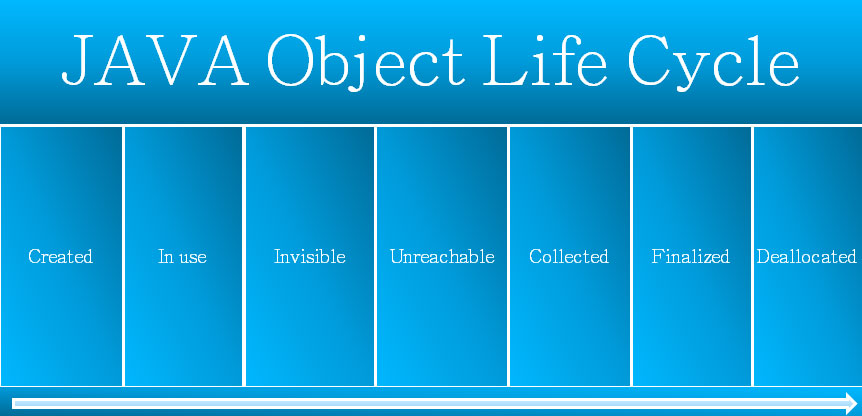
[](http://www.epractizelabs.com/certification/sun/scjp-exam-6.html)

图1. Java Object Life Cycle

## 1.创建阶段(Created)

在创建阶段系统通过下面的几个步骤来完成对象的创建过程

* 为对象分配存储空间
* 开始构造对象
* 从超类到子类对static成员进行初始化
* 超类成员变量按顺序初始化，递归调用超类的构造方法
* 子类成员变量按顺序初始化，子类构造方法调用

一旦对象被创建，并被分派给某些变量赋值，这个对象的状态就切换到了应用阶段

## 2.应用阶段(In Use)

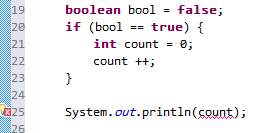
对象至少被一个强引用持有着。

## 3.不可见阶段(Invisible)

当一个对象处于不可见阶段时，说明程序本身不再持有该对象的任何强引用，虽然该这些引用仍然是存在着的。

简单说就是程序的执行已经超出了该对象的作用域了。

举例如下图：本地变量count在25行时已经超出了其作用域，则在此时称之为count处于不可视阶段。当然这种情况编译器在编译的过程中会直接报错了。



## 4.不可达阶段(Unreachable)

对象处于不可达阶段是指该对象不再被任何强引用所持有。

与“不可见阶段”相比，“不可见阶段”是指程序不再持有该对象的任何强引用，这种情况下，该对象仍可能被JVM等系统下的某些已装载的静态变量或线程或JNI等强引用持有着，这些特殊的强引用被称为”GC root”。存在着这些GC root会导致对象的内存泄露情况，无法被回收。

## 5.收集阶段(Collected)

当垃圾回收器发现该对象已经处于“不可达阶段”并且垃圾回收器已经对该对象的内存空间重新分配做好准备时，则对象进入了“收集阶段”。如果该对象已经重写了finalize()方法，则会去执行该方法的终端操作。

这里要特别说明一下：不要重载finazlie()方法！原因有两点：

* 会影响JVM的对象分配与回收速度

在分配该对象时，JVM需要在垃圾回收器上注册该对象，以便在回收时能够执行该重载方法；在该方法的执行时需要消耗CPU时间且在执行完该方法后才会重新执行回收操作，即至少需要垃圾回收器对该对象执行两次GC。

* 可能造成该对象的再次“复活”

在finalize()方法中，如果有其它的强引用再次持有该对象，则会导致对象的状态由“收集阶段”又重新变为“应用阶段”。这个已经破坏了Java对象的生命周期进程，且“复活”的对象不利用后续的代码管理。

## 6.终结阶段

当对象执行完finalize()方法后仍然处于不可达状态时，则该对象进入终结阶段。在该阶段是等待垃圾回收器对该对象空间进行回收。

## 7.对象空间重新分配阶段

垃圾回收器对该对象的所占用的内存空间进行回收或者再分配了，则该对象彻底消失了，称之为“对象空间重新分配阶段”。