1. **垃圾回收算法**

（1）引用计数法：堆中每个对象都有一个引用计数。被引用一次，计数加1，被引用的变量值变为null，则计数减1，知道计数为0，则变成无用对象。

优点是算法简单，缺点是“循环引用的无用对象”无法识别

（2）引用可达法（根搜索算法）

程序把所有的引用关系看做一张图，从一个节点GC ROOT开始，寻找对应的引用节点，找到这个节点以后，继续寻找这个节点的引用节点，当所有的引用节点寻找完毕之后，剩余的节点被认为是没有被引用的节点，即无用的节点

1. **通用的分代垃圾回收机制（Garbage Collection）**

不同的对象的生命周期不同。因此不同声明周期的对象可以采用不同的回收算法，以便提高回收效率。我们将对象分为三种状态：年轻代、年老代、持久代。

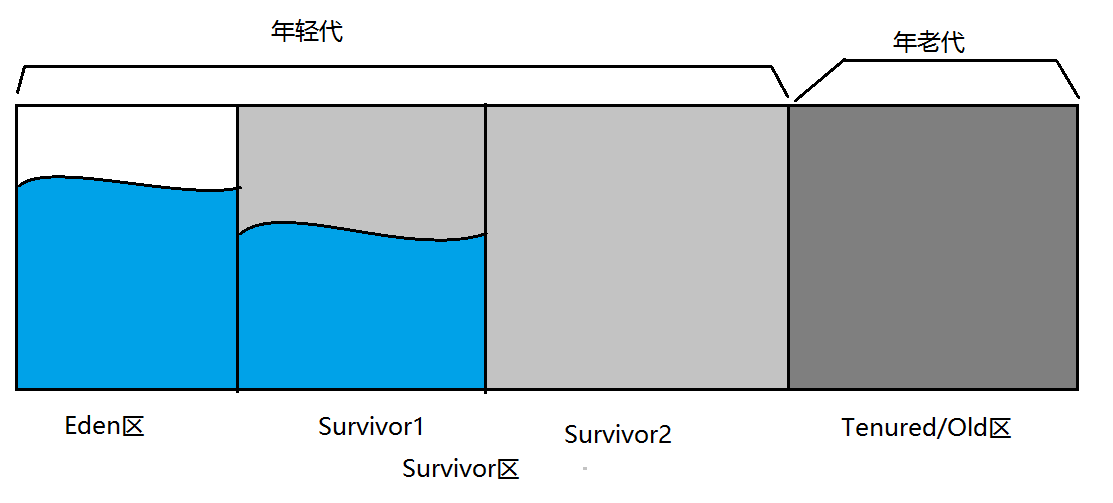
（1）年轻代：所有新生成的对象首先都是年轻代，年轻代的目标就是尽可能快速的收集掉那些生命周期短的对象。

（2）年老代：在年轻代中经历了N（默认15）次垃圾回收后仍然存活的对象，就会被放到年老代中

（3）持久代：用于存放静态文件，如Java类、方法等。持久代对垃圾回收没有显著影响

1. **JVM将堆内存划分为Eden、Survivor和Tenured/Old空间**

年轻代存储在Eden区和Survivor区、年老代存储在Tenured/Old区、持久代因为伴随着程序所以不清理



因为堆划分了不同区域所以也就有了不同区域的垃圾回收

1. Minor GC

用于清理年轻代区域。Eden区域满了就会触发一次Minor GC，清理无用对象，将有用对象复制到“Survivor1”、“Survivor2”区中（这两个区、大小空间也相同，同一时刻Survivor1和Survivor2只有一个在用，一个为空）

（2）Major GC：

用于清理老年代区域。

（3）Full GC：

用于清理年轻代、年老代区域。成本较高，会对系统性能产生影响

1. **垃圾回收过程**
2. 新创建的对象，绝大多数都会存储在Eden中
3. 当Eden满了（达到一定比例）不能创建新对象，则触发垃圾回收（Minor GC），将无用对象清理掉，然后剩余对象复制到某个Survivor中，如S1，同时清空Eden区
4. 当Eden再次满了，会将S1中的不能清空的对象存到另一个Survivor中，如S2，同时将Eden区中的不能清空的对象，也复制到S2中
5. 重复多次（默认15次）Survivor中没有被清理的对象，则会复制到老年代Old区中
6. 当Old区满了，则会触发一个完整的垃圾回收（Full GC）
7. **JVM调优和Full GC**

对JVM的调优过程中，很大一部分工作就是对于Full GC的调节，有如下原因可能导致Full GC

1. 年老代（Old）被写满
2. 持久代被写满
3. System.gc()被显示调用（程序建议GC启动，但不是调用，所以可能写了，但是仍没有调用）
4. 上次GC之后堆各区域分配策略动态变化