1.引用计数法(Java中已经被淘汰)

对象初始化时年龄默认为0岁,如果小于或者等于15岁放在新生代里面,大于15岁就会放在老年代里.

GC线程不定时进行回收时,如果对象被引用的话,年龄会增加1,如果没有被引用则减1.当年龄为0岁时,会被垃圾回收机制认为是不可达对象,被清理掉.

缺点:循环依赖问题

Object a = new Object();

Object b = new Object();

a=b;b=a;a=null;b=null;

这时无法判断是否被继续引用

2.根搜索算法(现在主流)

需要一个根节点对象

堆内存中的对象必须跟其他区(方法区、虚拟机栈、本地方法栈)里的对象产生关联,才不会被回收,堆内存中两个对象产生关联还是会被回收

3.标记清除算法

0表示存活 1表示未存活

标记阶段：找到所有可访问的对象，做个标记

清除阶段：遍历堆，把未被标记的对象回收

应用场景:该算法一般应用于老年代,因为老年代的对象生命周期比较长.

缺点:由于遍历,一个一个删,但是GC回收有时间限制,对象特别多的时候会出现删不完现象,产生碎片化.

4.复制算法

针对新生代,比如user0,user1,user2是刚创建的对象在伊甸园区,发现被循环引用,然后就进入s0区,这时user0不再被继续引用了,user1,user2就会被复制到s1区,然后删除s0中的对象.user4又被创建且引用次数多,则进入s1区.

优缺点:效率高,但是浪费空间.适合新生代.

5.压缩算法

是标记清楚算法的升级版,把可到达和不可到达的对象进行排序,然后一次性删除不可到达的对象.相互关联的对象会被排在一起.

使用场景:老年代

6.分代算法

就是将以上回收算法针对堆内存不同的区进行分配.

新生代:使用复制算法,GC每次回收的时候,伊甸园区被频繁引用的对象进入幸存者区,幸存者区被频繁引用的对象进入老年代,不可达对象被删除.

老年代:标记压缩算法或者标记清除算法.

垃圾收集器:

1.串行与并行收集器

串行回收: JDK1.5前的默认算法 缺点是只有一个线程，执行垃圾回收时程序停止的时间比较长

并行回收: 多个线程执行垃圾回收适合于吞吐量的系统，回收时系统会停止运行

2.serial收集器

串行收集器是最古老，最稳定以及效率高的收集器，可能会产生较长的停顿，只使用一个线程去回收。新生代、老年代使用串行回收；新生代复制算法、老年代标记-压缩；垃圾收集的过程中会Stop The World（服务暂停）

一个单线程的收集器，在进行垃圾收集时候，必须暂停其他所有的工作线程直到它收集结束。

特点：CPU利用率最高，停顿时间即用户等待时间比较长。

适用场景：小型应用

通过JVM参数-XX:+UseSerialGC可以使用串行垃圾回收器。

3.ParNew收集器

ParNew收集器其实就是Serial收集器的多线程版本。新生代并行，老年代串行；新生代复制算法、老年代标记-压缩

参数控制：-XX:+UseParNewGC ParNew收集器

-XX:ParallelGCThreads 限制线程数量

4.parallel 收集器

Parallel Scavenge收集器类似ParNew收集器，Parallel收集器更关注系统的吞吐量。可以通过参数来打开自适应调节策略，虚拟机会根据当前系统的运行情况收集性能监控信息，动态调整这些参数以提供最合适的停顿时间或最大的吞吐量；也可以通过参数控制GC的时间不大于多少毫秒或者比例；新生代复制算法、老年代标记-压缩

采用多线程来通过扫描并压缩堆特点：停顿时间短，回收效率高，对吞吐量要求高。适用场景：大型应用，科学计算，大规模数据采集等。通过JVM参数 XX:+USeParNewGC 打开并发标记扫描垃圾回收器。

5.cms收集器

6.g1收集器