# 请简述 JVM 垃圾回收原理。

1. JVM从GCROOT出发，对存活对象进行标记，采用的是可达性分析算法，

以下条件可以当做GCROOT

1. 线程栈中的局部变量
2. 类的静态变量
3. 年轻代采用复制算法，老年代采用标记-清理、标记-整理算法

复制算法：先标记存活对象，然后将存活对象复制到另外一块空间，将剩下的垃圾对象全部清除，由于要预留一半空间，导致空间浪费

标记-清理：先标记存活对象，然后将垃圾对象进行清除，会造成空间碎片

标记-整理：先标记存活对象，然后将存活对象移动到一起，在清除垃圾对象

1. 内存分配规则：
2. 大部分对象优先分配在新生代
3. 大对象直接进入老年代

通过-XX:PretenureSizeThreshold设置

1. 多次存活的对象进入老年代

通过-XX:MaxTenuringThreshold设置，默认一般是15

1. 动态年龄判断

新生代GC后，进入Survivor之前，将存活对象从0岁开始加S0+S1+S2…+Sn 是否大于S的50%，如果满足，那么大于等于n岁的存活对象晋升到老年代

1. 老年代空间担保机制

每次MinorGC之前都要进行判断，新生代大小是否小于老年代可用连续空间大小，如果否，继续判断是否开启空间担保机制（默认开启），判断每次新生代晋升的平均大小是否小于老年代可用连续空间大小，如果是，可以进行MinorGC，否则先进行一次FullGC

1. 垃圾回收器

新生代：

Serial：单线程、复制算法

ParNew：多线程、复制算法

老年代：

Serail Old：单线程、标记-整理算法

CMS：多线程，标记-清理算法

G1: 适用于新生代和老年代

特点：

1. 分成若干大小一样的Region区域
2. 逻辑上任然区分新生代和老年代
3. 大对象由专门Region区域存储
4. 可以设置垃圾回收的预期停顿时间

# 设计一个秒杀系统，主要的挑战和问题有哪些？核心的架构方案或者思路有哪些？

**1. 主要挑战**

* 保证三高

系统高并发

系统高可用

系统的高性能

* 系统安全

防黑客攻击/秒杀器

**2.主要问题**

* 网络带宽耗尽

运营推广页面导致带宽耗尽

* 应用服务器资源不足

Load 飙高，停止响应，服务器崩溃

* 数据库服务器资源足
* 恶意请求

秒杀器抢单

**3.解决思路**

* 网络资源

资源静态化；

限制大小；

* 应用服务资源

设置请求阀值；

简化流程；

启用缓存、异步、集群；

* 数据库资源

启用缓存、异步、集群；

* 安全问题

程序关键要素动态下发；

硬件防火墙