GC总结：

* 串行 GC (serial收集器)
  + 年轻代使用拷贝-复制，老年代使用标记-清除-整理
  + 发生GC时，会暂停所有工作线程，知道收集结束
  + 内存越小，GC次数越多
* 并行 GC (ParNew收集器，ParallerScavenge收集器)
  + 年轻代使用拷贝-复制，老年代使用标记-清除-整理
  + GC处理时，暂停业务处理，所有线程处理GC垃圾回收。平常运行时，所以线程都去处理业务。因此，吞吐量比较高。
  + 内存越小，GC次数越多
  + ParNew收集器只能与CMS收集器配合，适用于低延迟
  + ParallelGc + ParallelGc Old 适用于高吞吐量
* CMS GC
  + 以获取最短回收停顿时间为目标
  + 年轻代使用拷贝-复制，老年代使用标记-清除
  + CMS默认GC线程数是（处理器核心数+3）/4(处理器4核以上推荐)，并且老年代只清除，无整理（无法处理浮动垃圾）。所以当GC发生时，吞吐量不如并行 GC,但是可以控制GC时间
  + CMS GC 6个阶段
    - 初始化标记 - 暂停GC
    - 并行标记
    - 并行预清理
    - 最终标记 - 暂停GC
    - 并行清理
    - 并行重置
  + 因为无整理，并且CMS GC 6阶段 暂停时间短，所以延迟比较低
  + 内存越小，GC次数越多
* G1 GC
  + 化整为零的思路
  + 不分代,使用 region(2048) 存储数据（更高的内存占用负担，本质是hash表），分为：
    - 原始快照（SATB）算法
    - Eden区 （标记-复制 算法）
    - 存活区
    - 老年区 (标记-复制-整理 算法)
  + GC 4个阶段
    - * 初始标记（Initial Marking）
      * 并发标记（Concurrent Marking）
      * 最终标价（Final Marking）
      * 筛选回收（Live Data Counting and Evacuation）