/\*\*

\* 两种标记方法

\* 1、引用计数器

\* 2、引用跟踪

\*

\* 四种类型的GC ROOT

\* 1、线程栈，栈帧本地变量引用的对象

\* 2、方法区静态变量引用的变量

\* 3、方法区常量引用的变量

\* 4、JNI方法引用的变量

\*

\* 三种垃圾手机算法

\* 1、标记-清除

\* 2、复制

\* 3、标记-整理

\*

\* 七种垃圾收集器

\* 新生代，目前均使用的是复制算法

\* 1、serial：单线程进行回收，中断所有的用户线程

\* 2、parNew：就是一个并行的serial，中断用户线程

\* 3、parallel scavenge:并行回收器，但是吞吐量优先而不是用户停顿最短优先，适用于后台业务

\*

\* 年老代

\* 4、CMS：使用标记-清除算法，多线程回收，并不中断用户线程，以用户线程停顿最短为目标

\* 这个算法分为四个部分

\* 1、初始标记（stop the world）停顿时间极短

\* 2、并发标记

\* 3、重新标记（stop the world）停顿时间极短

\* 4、并发清除

\* 该算法缺点：

\* 1、产生内存碎片，配合-XX：CMSFullGCsBeforeCompaction来解决这个问题

\* 2、服务器cpu低于4核时对应用的影响也是比较大的

\* 5、parallel old：使用标记-整理算法，并行回收，中断用户线程，只能与parallel scavenge组合使用，来达到吞吐量最大化

\* 6、serial old：使用标记-整理算法，单线程回收，中断用户的请求

\*

\* JDK中新的垃圾回收器

\* 7、G1 garbage first

\* 新一代的垃圾收集器，采用标记-整理算法

\* 对G1来说没有了以前年轻代、年老代这些内存区域的划分，他把整个java堆

\* 逻辑上分为多个大小固定的区域，并跟踪这些堆的垃圾堆积程度，优先回收

\* 堆积程度最高的区域，可以运行指定用户线程停顿的最大时间及垃圾回收线程最大使用的回收时间

\* 这个回收器相比CMS没有内存碎片，在不牺牲吞吐量的情况下，最小化用户线程停顿时间

\*

\*

\* 默认垃圾收集器

\* 新生代

\* client模式：serial

\* server模式：parNew

\* 年老代

\* client模式：serial old

\* server模式：CMS

\*

\*

\* 垃圾回收中并行与并发的区别：

\* 这里特指垃圾回收的场景下的区别

\* 并行是指多个回收线程在运行，但是用户进程还是停止的

\* 并发是用户进程与回收进程同时进行，可以用传统的多线程来理解这种情况

\* （即有可能并行执行也有可能交叉执行）

\*

\*

\* java内存分配策略：

\*

\* 回收类型

\* minor GC

\* full GC

\*/