Jvm内存图

幸存1区

幸存0区

伊甸区

字符串常量池（String pool）gc 不会回收

永久区

养老区

新生区:

新new的对象在伊甸区，满了后垃圾回收下，对象移到幸存区

幸存区满了后移到养老区，垃圾回收下

养老区满了后触发FULL GC ,如果还没有能回收的，就报OutOfMemoryError错误

栈内存 Stack

1.特点：空间小，速度快。

2.存放对象的引用和基本类型；基本类型的计算在栈内存中；

启用一个线程时，JVM就为他分配一个Java栈，当线程调用一个方法时，JVM就会在该线程对应的栈中压入一个帧，当执行这个方法时，它使用这个帧来存储参数、局部变量、中间运算结，栈中只保存基础数据类型的对象和自定义对象的引用【栈先进后出】，栈帧由三部分组成：局部变量区、操作数栈、帧数据区。

3.由线程创建，线程结束，由jvm回收

4.设置java –Xss(size)

超过报StackOverflowError错误，启动的线程过多报栈溢出，1个线程初始大小约占1M【内存溢出】

5.

堆内存Heap

1.特点：空间大速度慢，一般对象在这里生成，使用，消亡；对象计算在堆内存中

2.设置java –Xmx1536m –Xms1024m 最小 最大

3.新生区：养老区一般为1:3

4.优化：可以通过参数设置回收策略，比如设置收集线程个数

5. 对象不能及时释放内存满了后，报堆溢出OutOfMemoryError错误【内存泄漏】

方法区

1.又叫静态区，跟堆一样，被所有的线程共享。方法区包含所有的class和static变量。  
2.方法区中包含的都是在整个程序中永远唯一的元素，如class，static变量。

**Java内存泄露与溢出的区别**

内存溢出就是你要求分配的内存超出了系统能给你的，系统不能满足需求，于是产生溢出。

Java内存泄漏就是没有及时清理内存垃圾，指在堆上分配的内存没有被释放，从而失去对其控制，导致系统无法再给你提供内存资源（内存资源耗尽）。

区别：内存溢出，提供的内存不够；Java内存泄漏，无法再提供内存资源

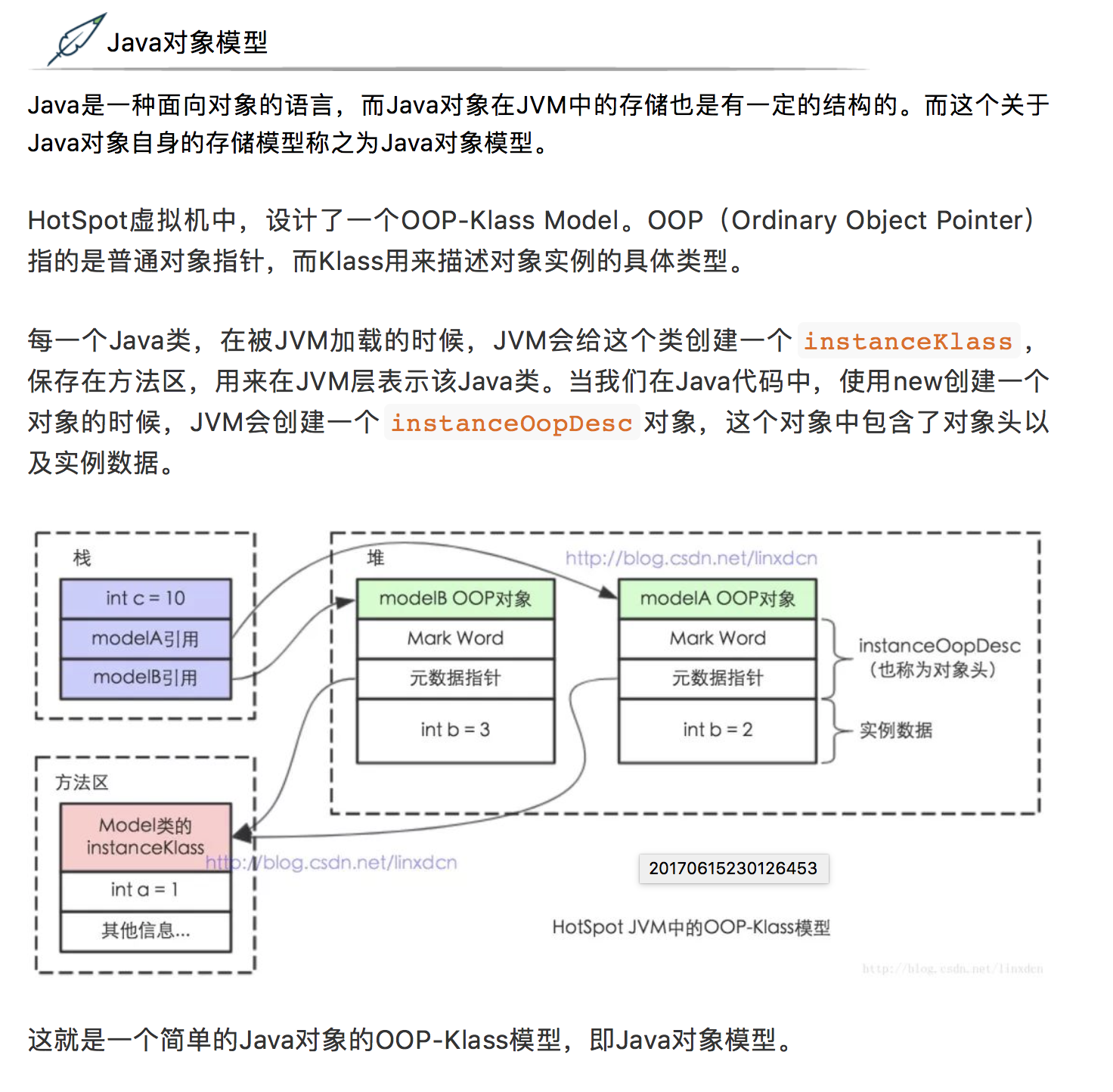
**内存泄漏：****OutOfMemoryError，**因为内存溢出或没有可用的内存提供给垃圾回收器时，Java 虚拟机无法分配一个对象，这时抛出该异常。

发生原因:创建对象太多

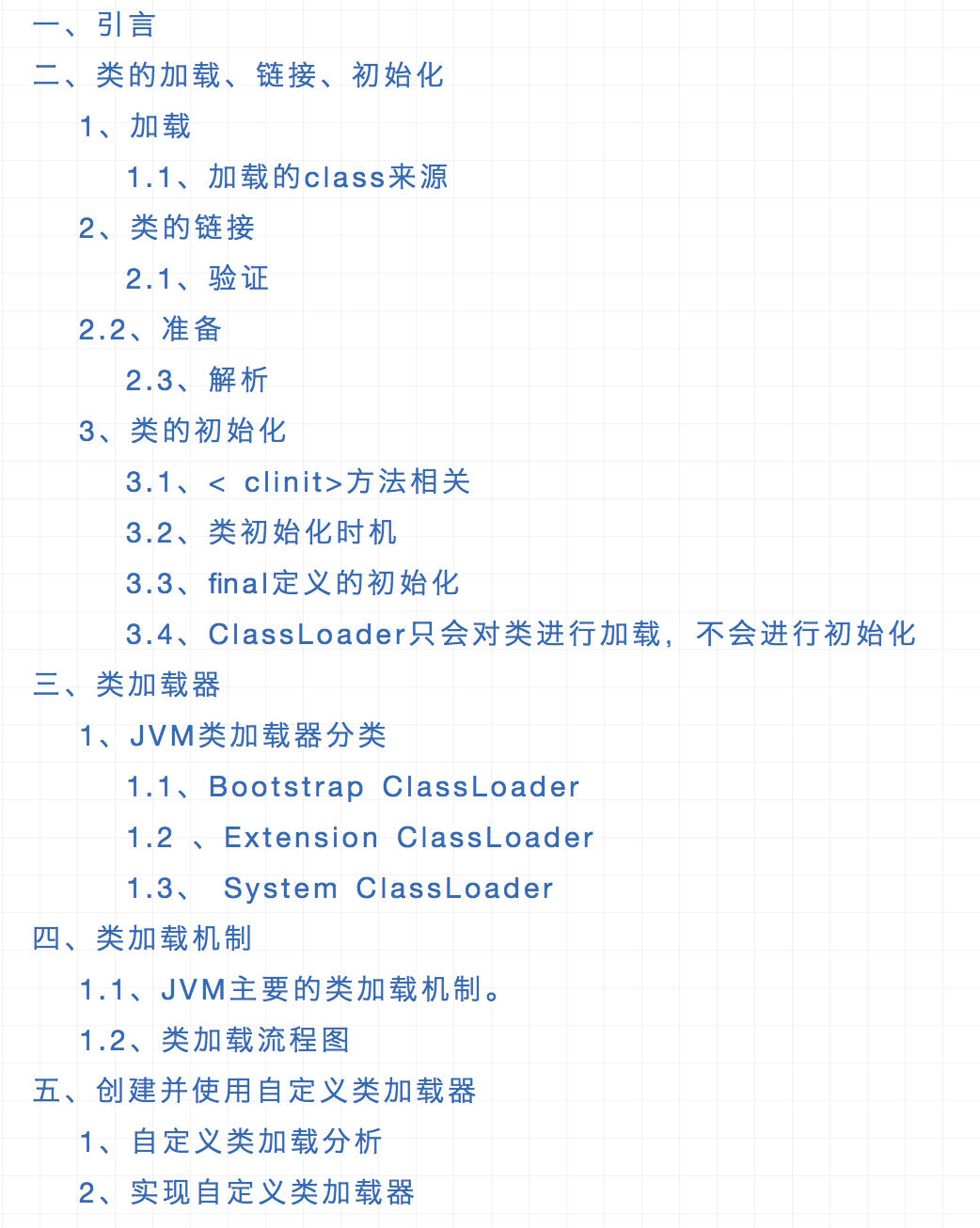
**内存溢出：****StackOverflowError：**当应用程序递归太深而发生堆栈溢出时，抛出该错误。

发生原因:创建的线程太多；递归调用某个方法

2. 



<https://www.cnblogs.com/9dragon/p/11005996.html>



**GC算法大致分为以下几种：**

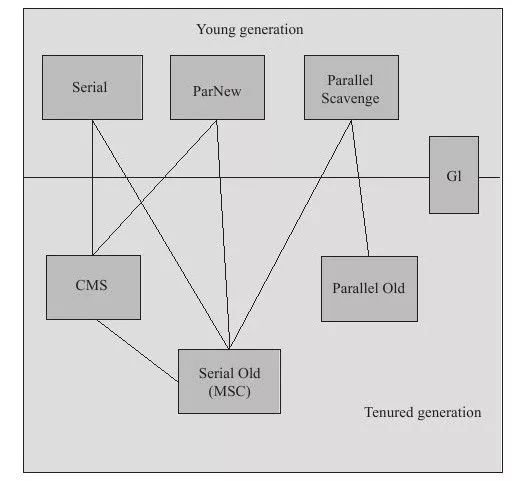
标记-复制：（新生代）

标记-清除：（老生代）

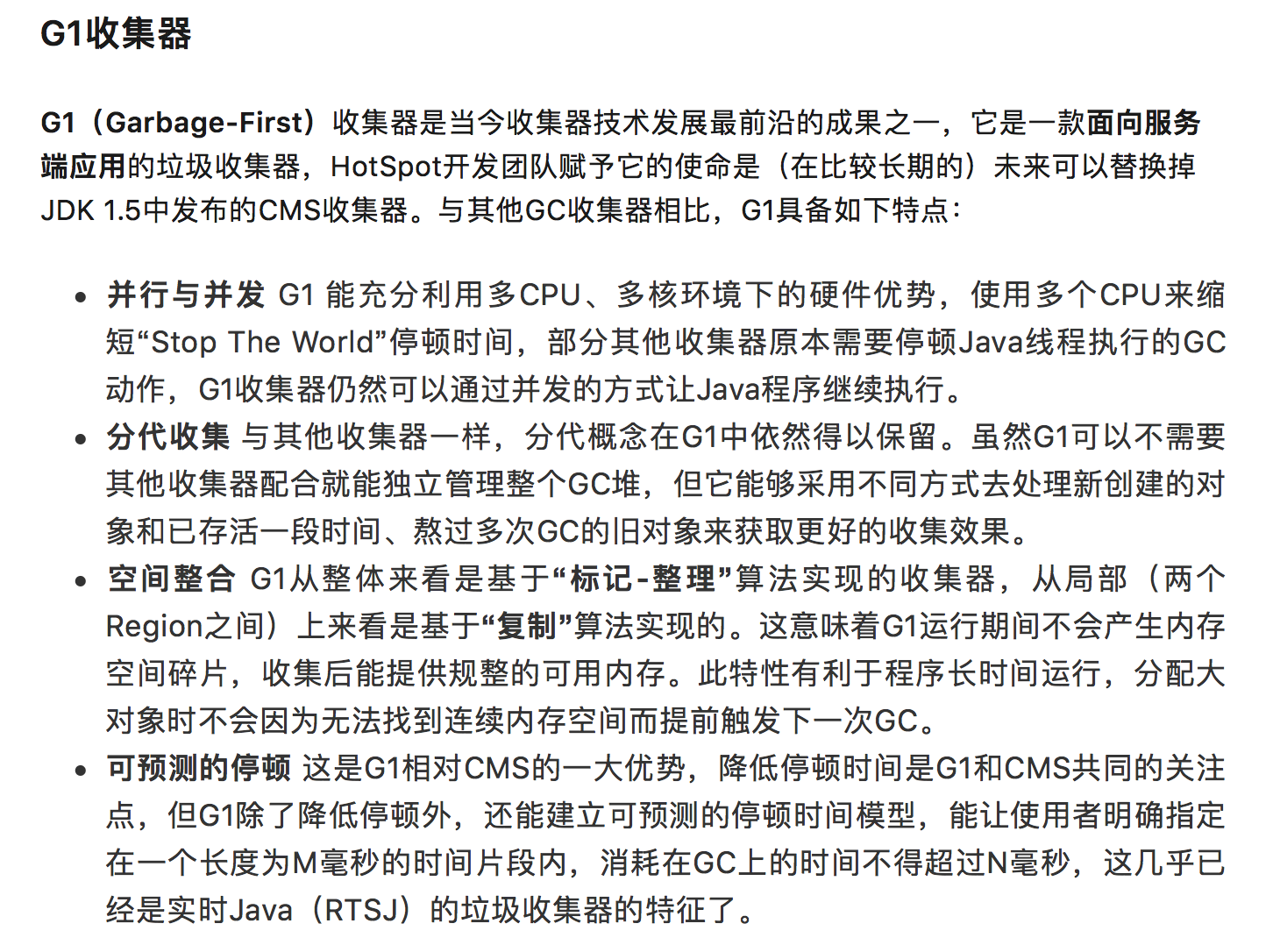
标记-压缩 :（老生代）

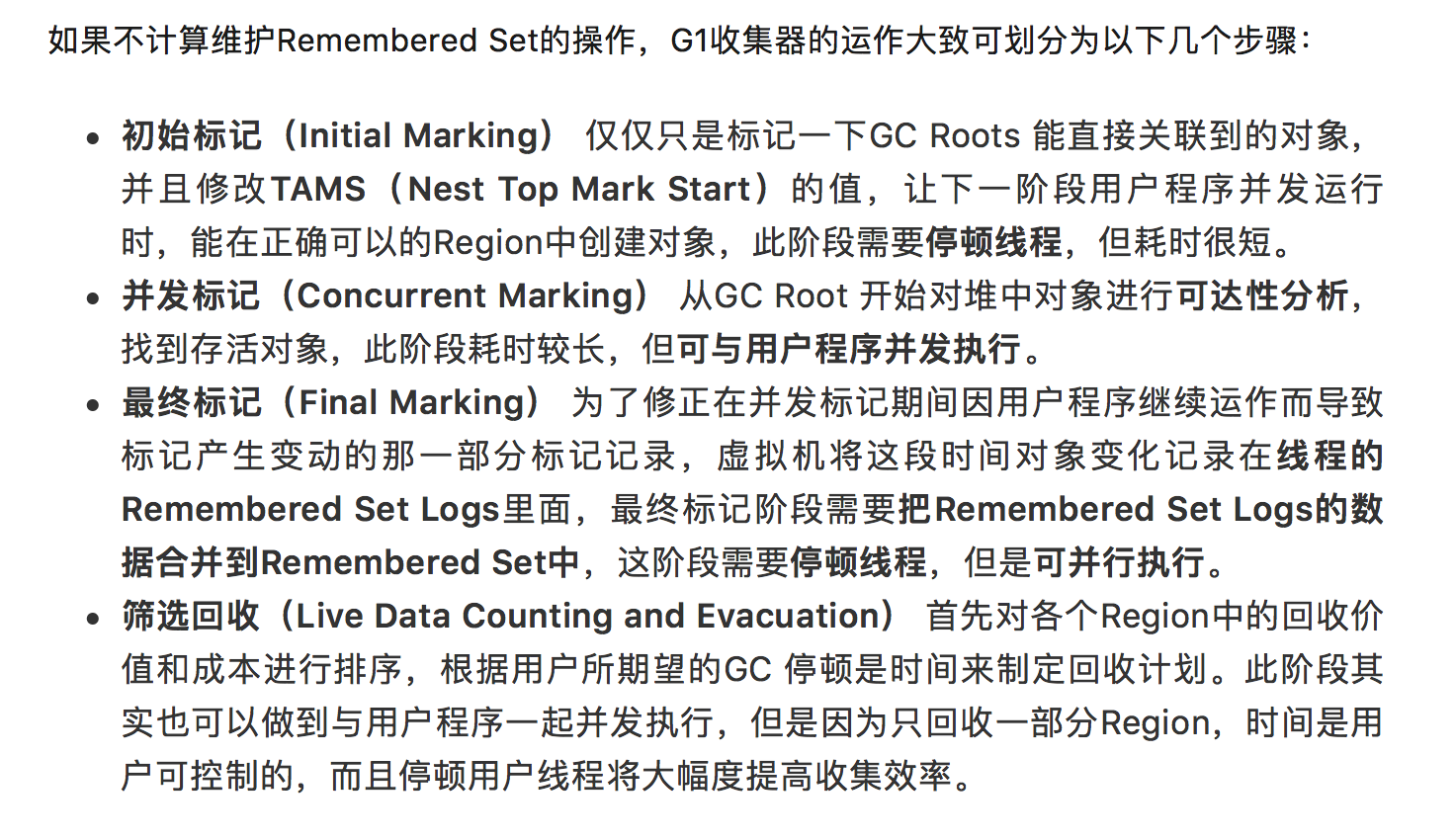
收集器：并行并发区别

<https://mp.weixin.qq.com/s/T9VzqaAWZd0XAIyVy0podA>



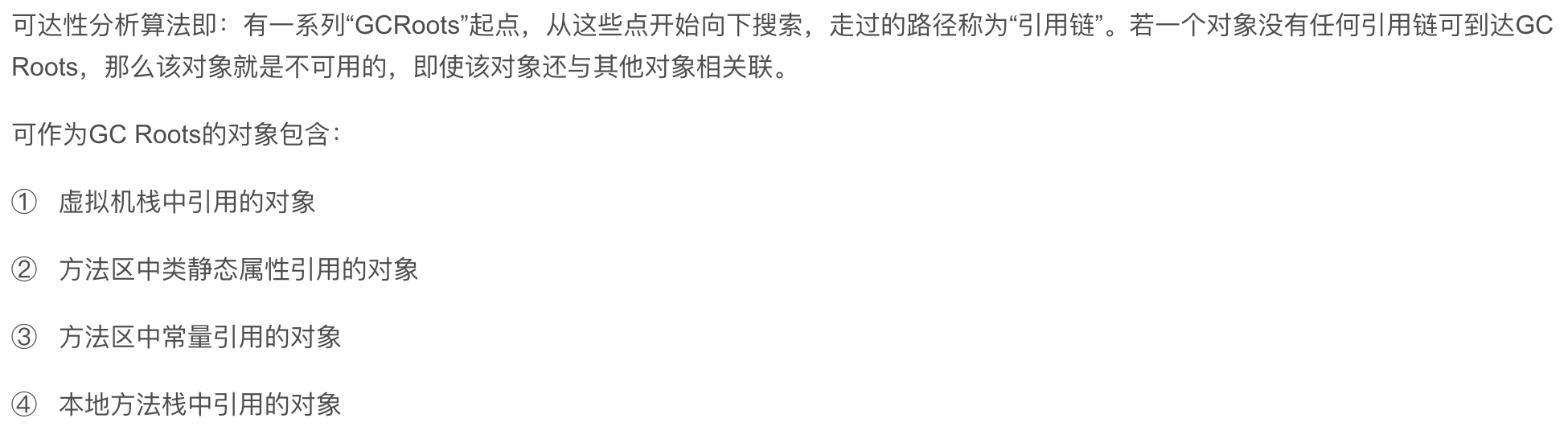








对象可达性分析：



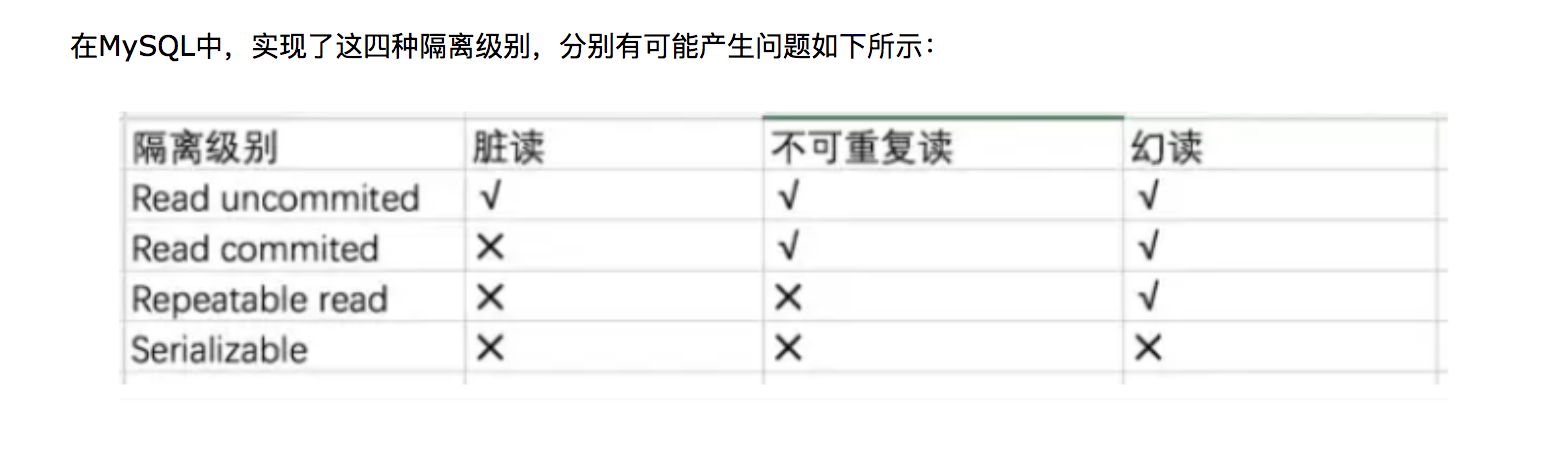
young gc，full gc：



索引在什么情况下会失效”:

1. 对列进行计算或者是使用函数,则该列的索引会失效
2. 不匹配数据类型,会造成索引失效
3. where语句中使用了IS NULL或者IS NOT NULL,会造成索引失效
4. 在WHERE中使用OR时，有一个列没有索引，那么其它列的索引将不起作用

数据库隔离级别：





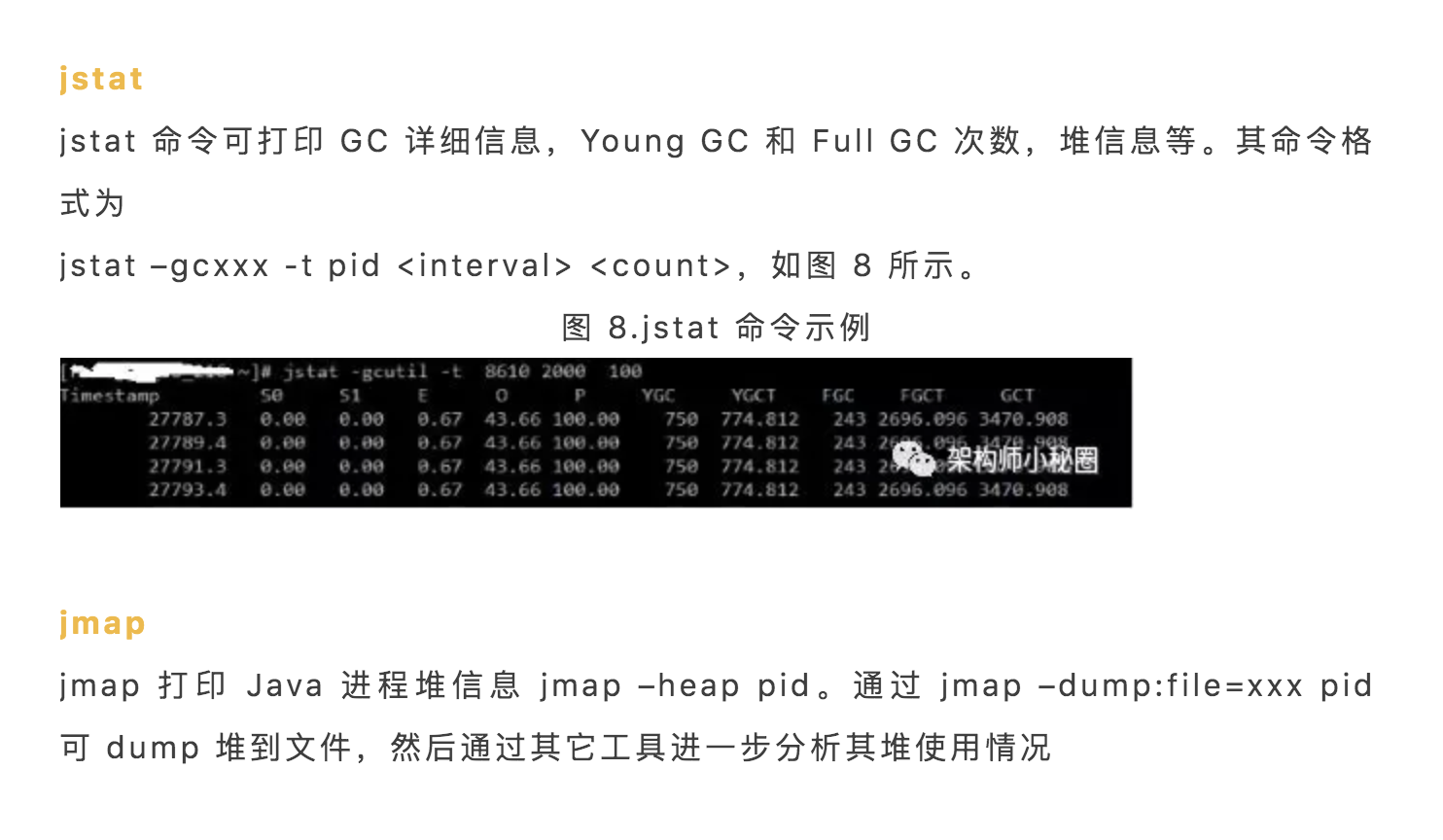
**IOC VS DI**

IOC，控制反转；DI，依赖注入；

**只有把对象交给Spring，才能由Spring帮助完成属性设置；因此，依赖注入不能单独存在，需要在IOC基础之上完成操作。**

**\*** 计算衡量标准   
 **\*** load average: 3.04, 2.71, 2.45 （1 分钟、5 分钟、15 分钟机器的负载）  
 **\*** cpu 核数： grep "physical id"| wc -l   
 **\*** 衡量标准 = 0.7 \* cpu 核数  
**\*** 获取现象id： top -H -p pid (P：按%CPU对进程排序 H：切换在进程列表中显示所有线程信息)  
**\*** 转化为16十进制： printf "%x\n" 线程id  
**\*** 查看线程栈信息 ： jstack pid |grep 线程id





mat