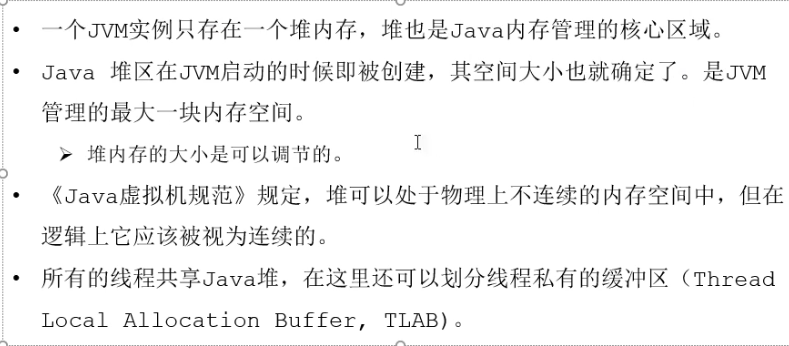
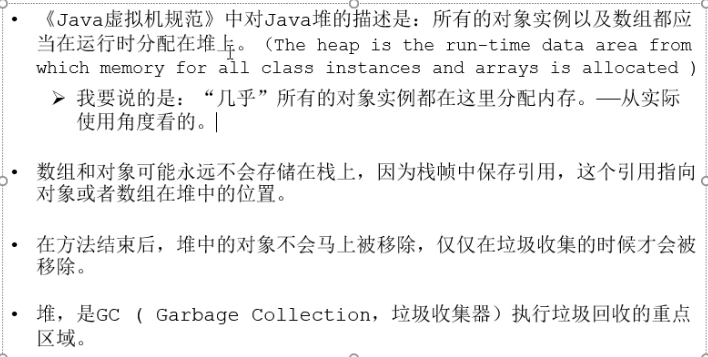
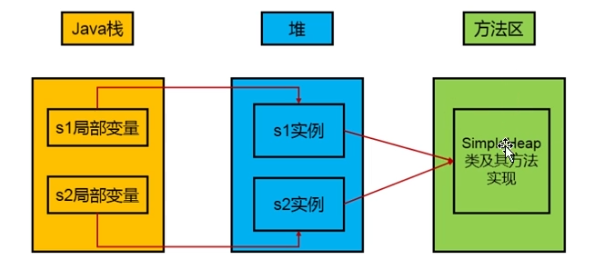
# 概述

## 概述

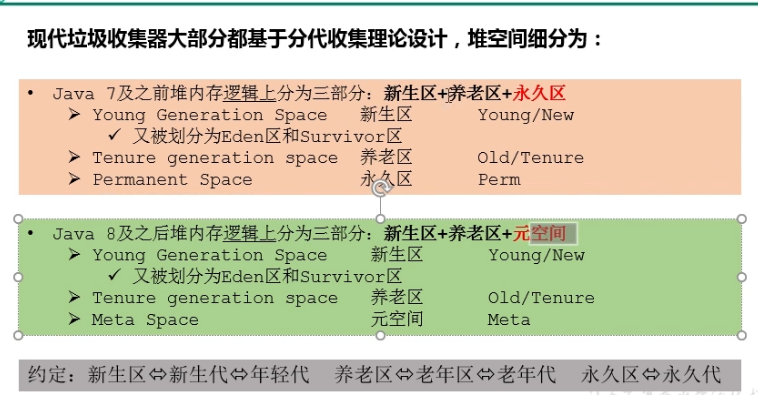




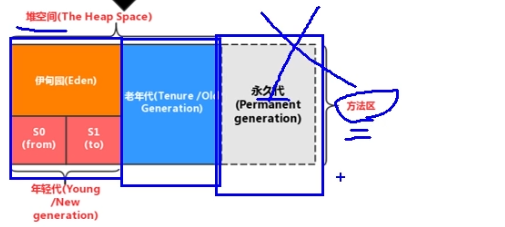
## 堆栈方法区的关系

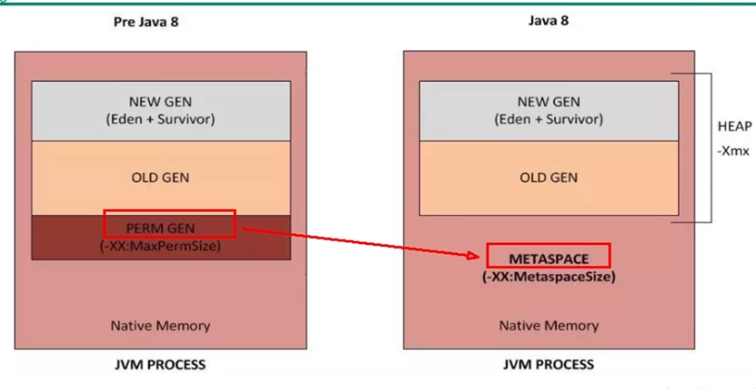


# 二、堆内存的内部结构

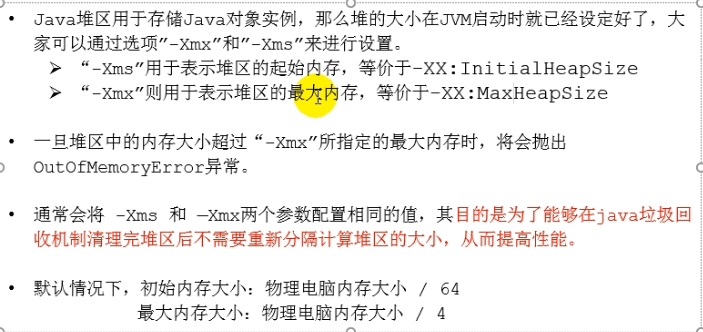


元空间暂时不属于堆空间





# 三、堆空间的大小设置



1. 设置堆空间大小的参数

-Xms 用来设置堆空间（年轻代+老年代）的初始内存大小

-X 是jvm的运行参数

ms 是memory start

-Xmx 用来设置堆空间（年轻代+老年代）的最大内存大小

2. 默认堆空间的大小

初始内存大小：物理电脑内存大小 / 64

最大内存大小：物理电脑内存大小 / 4

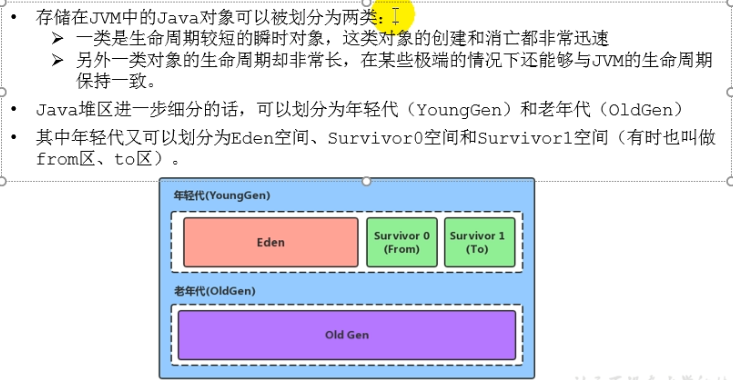
3. 手动设置：-Xms600m -Xmx600m

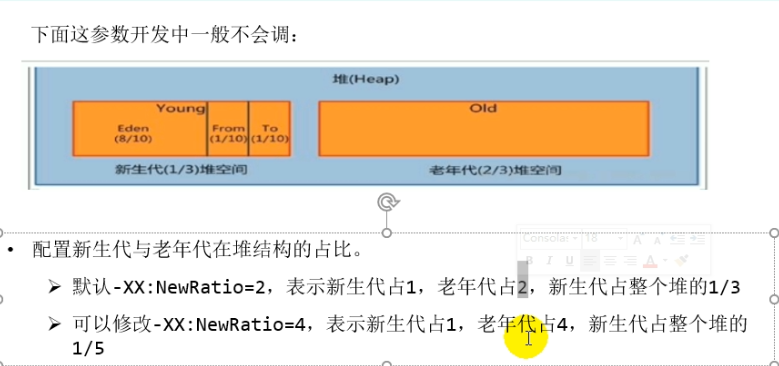
开发中建议将初始堆内存和最大的堆内存设置成相同的值。因为扩容和削容要耗费性能所以直接设置成一样的

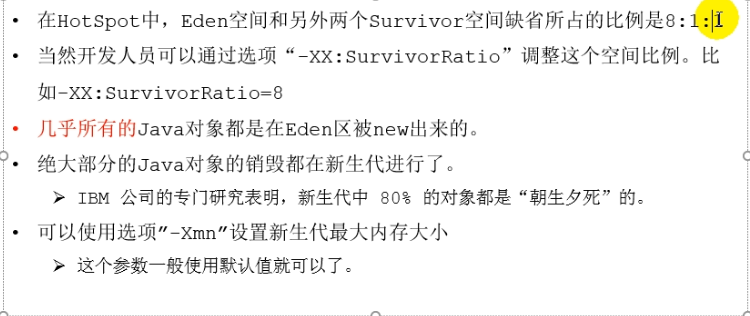
4. 查看设置的参数：方式一： jps / jstat -gc 进程id

方式二：-XX:+PrintGCDetails

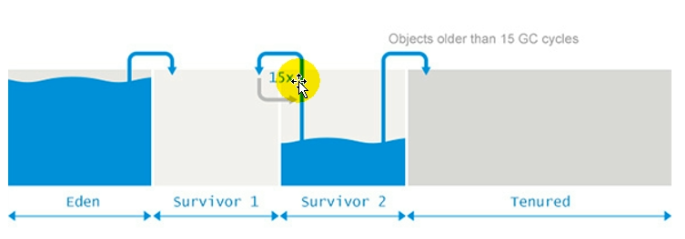
# 四、新生代与老年代







实际上测试中 伊甸园区和survivor区 的比例是6：1：1



参数设置

-XX:NewRatio ： 设置新生代与老年代的比例。默认值是2.

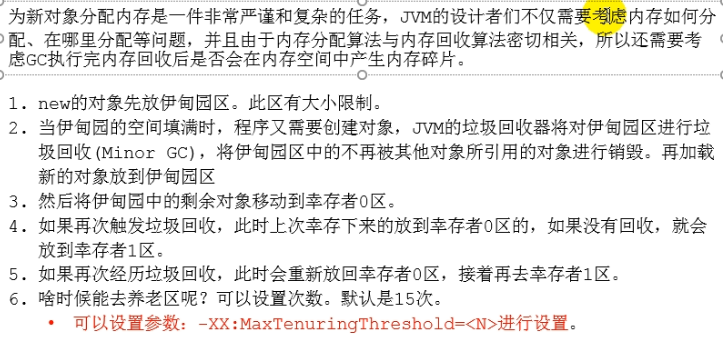
-XX:SurvivorRatio ：设置新生代中Eden区与Survivor区的比例。默认值是8

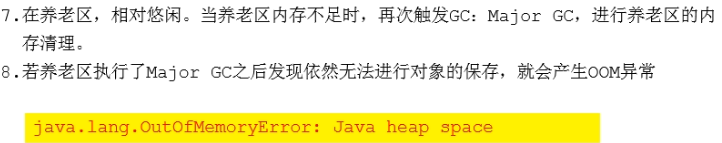
-XX:-UseAdaptiveSizePolicy ：关闭自适应的内存分配策略 （暂时用不到）

-Xmn:设置新生代的空间的大小。 （一般不设置）

# 对象分配过程

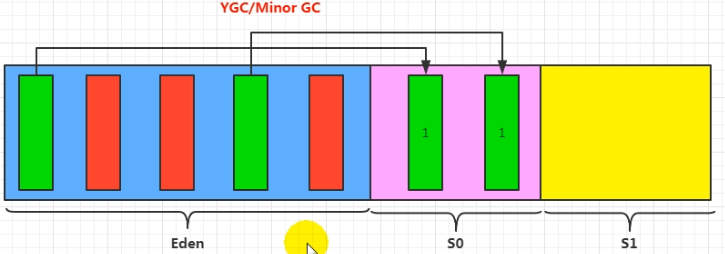
## 规则



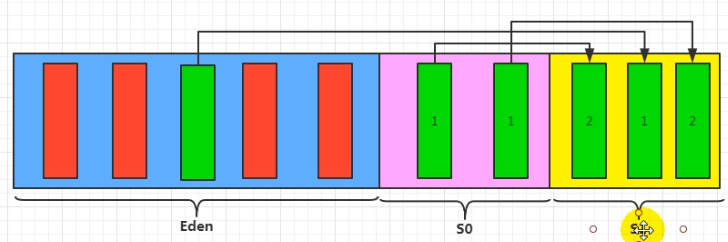


## 具体过程

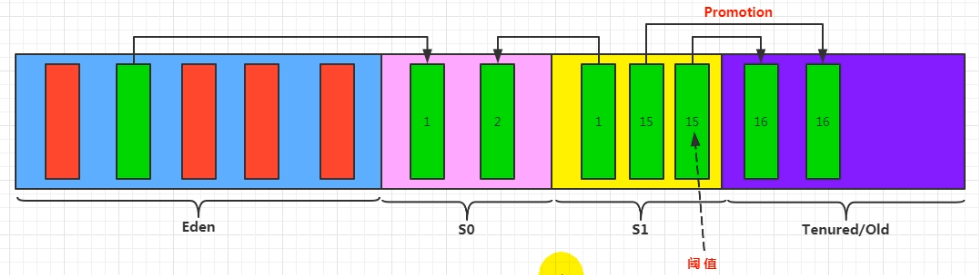
一、伊甸园区满的时候执行youngGC （红色的被销毁，绿色的进入s0区，年龄age设置为1），此时伊甸园区清空



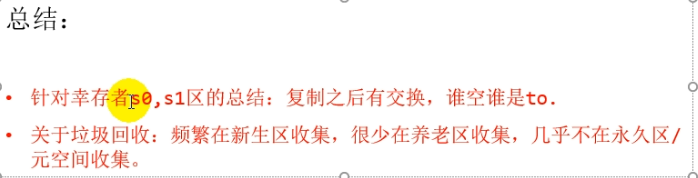
二、第二次执行 youngGC时，把伊甸园区中的绿色对象放入空的幸存者区中（s1）,然后把另一个幸存者区（s0）中的对象放入s1中，并且年龄+1，此时伊甸园区和s0中都清空，下一次执行youngGC时，再执行同样的操作



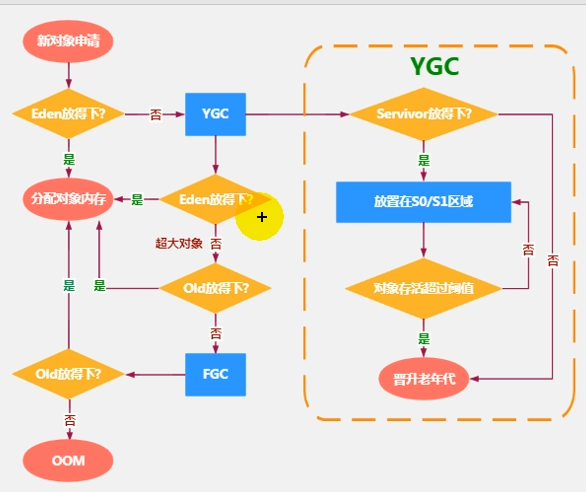
三、当幸存者区中的对象年龄变为15时（默认15，可配置MaxTenuringThreshold），则Promotion（晋升）进入老年区



只有伊甸园区满的时候会触发youngGC，幸存者区中满了不会触发youngGC



## 整体流程



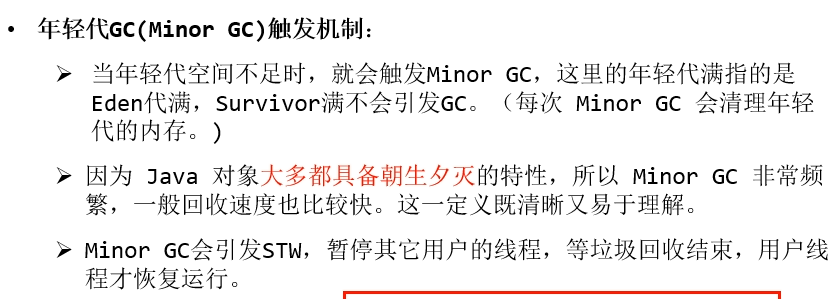
# 三种GC的讲解

调优就是希望GC少一些，因为在GC的使用会触发 STW（stop the word）机制，就会让用户线程暂停，所以影响执行效率

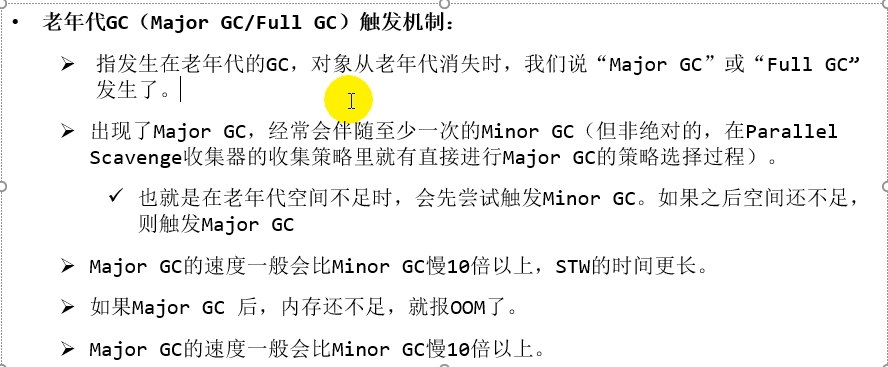
三种GC方式的区别



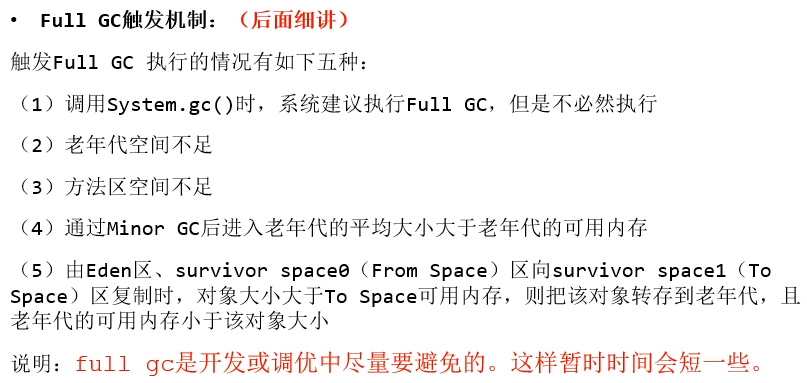
## Minor GC

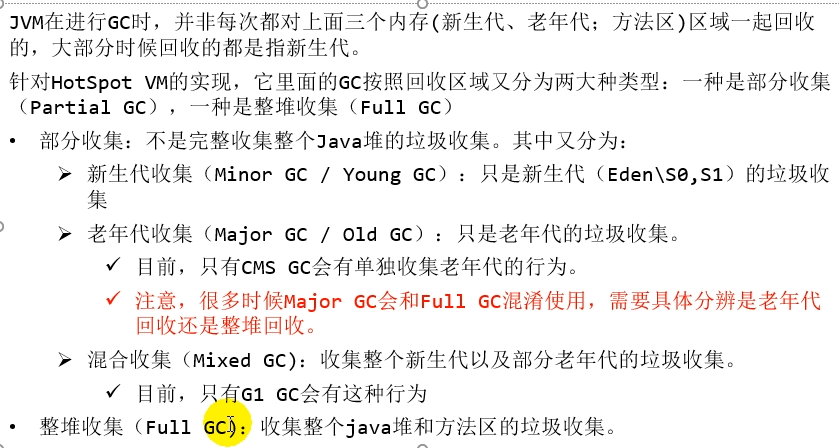


## Major GC

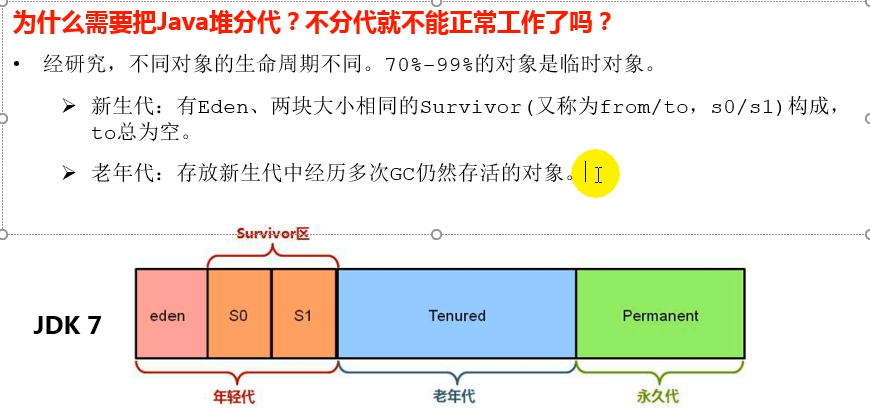


## Full GC

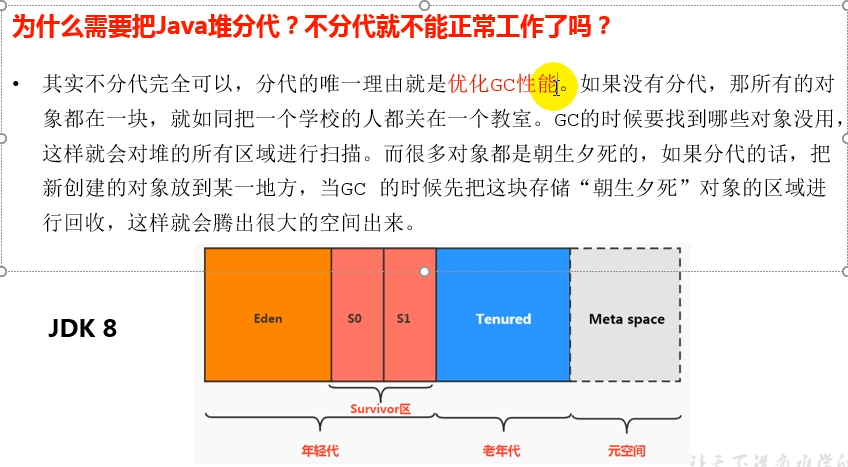




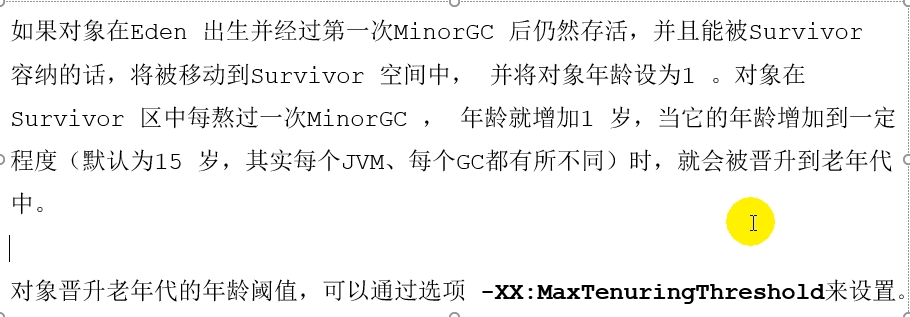
# 堆空间分代的思想

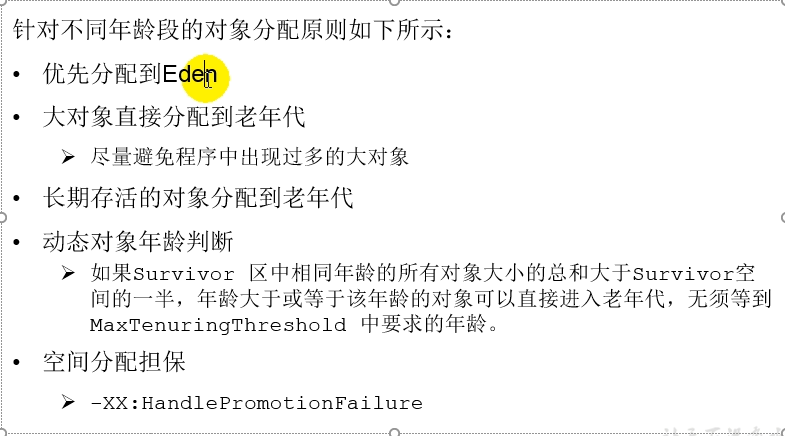


分代的唯一理由就是优化GC的性能



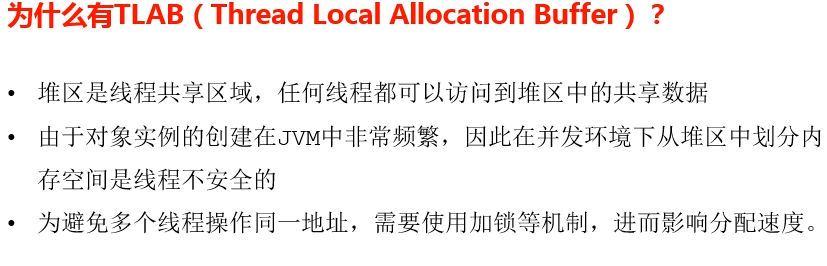
1. 内存分配策略

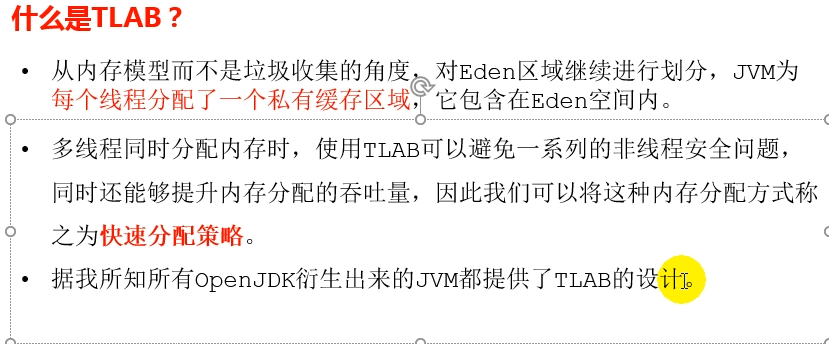


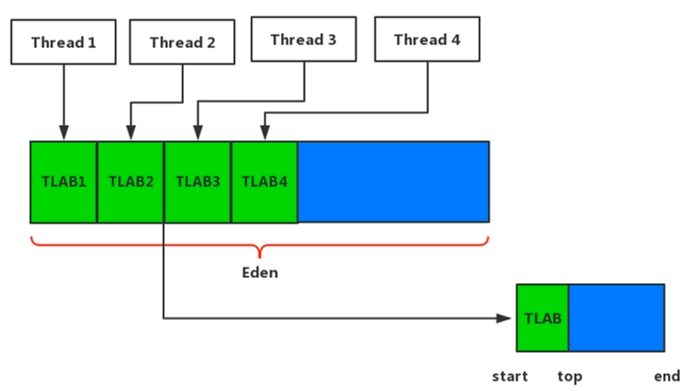


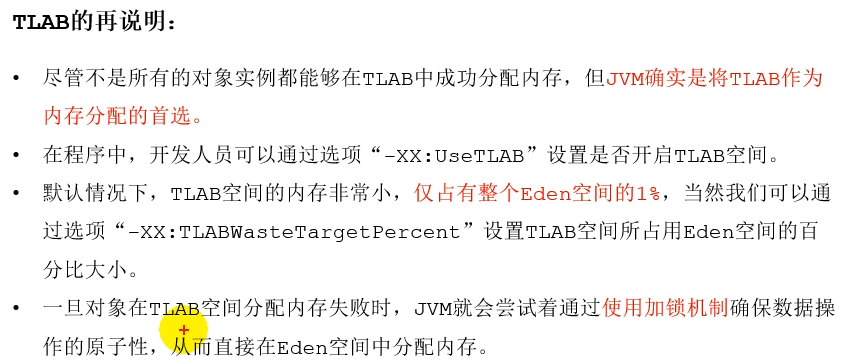
尽量避免程序中出现大对象，比如过大的集合数组之类的对象，会直接触发GC。而且若是朝生夕死的大对象的话，更影响性能。

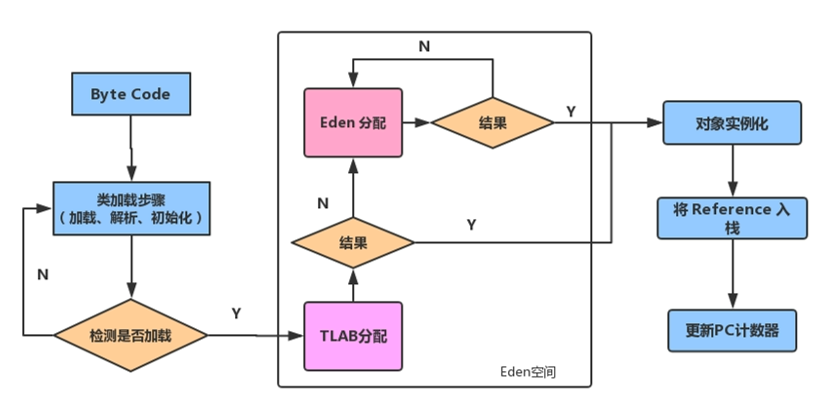
# 八、TLAB



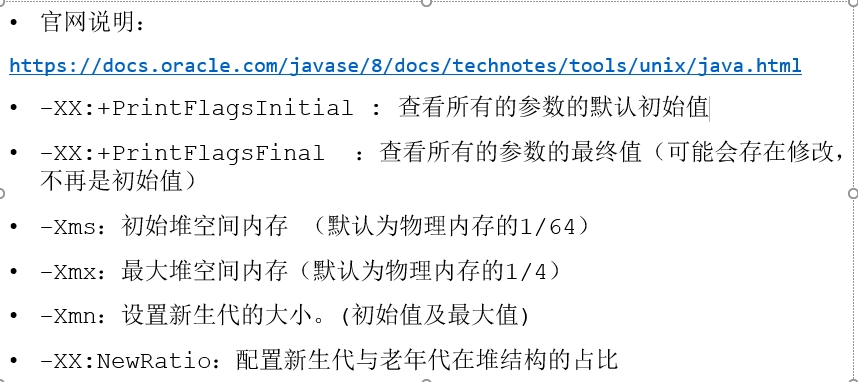


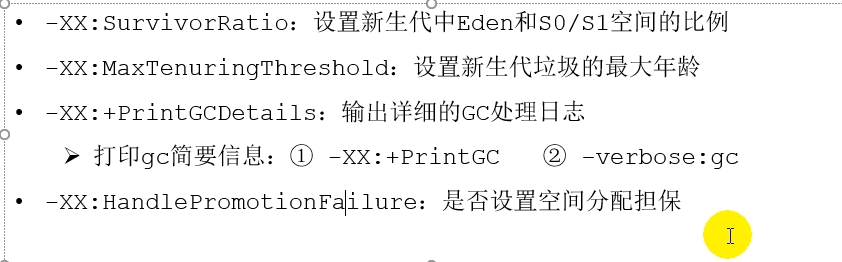






# 堆空间的参数设置





测试堆空间常用的jvm参数：

-XX:+PrintFlagsInitial : 查看所有的参数的默认初始值

-XX:+PrintFlagsFinal ：查看所有的参数的最终值（可能会存在修改，不再是初始值）

具体查看某个参数的指令： jps：查看当前运行中的进程

jinfo -flag SurvivorRatio 进程id

-Xms：初始堆空间内存 （默认为物理内存的1/64）

-Xmx：最大堆空间内存（默认为物理内存的1/4）

-Xmn：设置新生代的大小。(初始值及最大值)

-XX:NewRatio：配置新生代与老年代在堆结构的占比

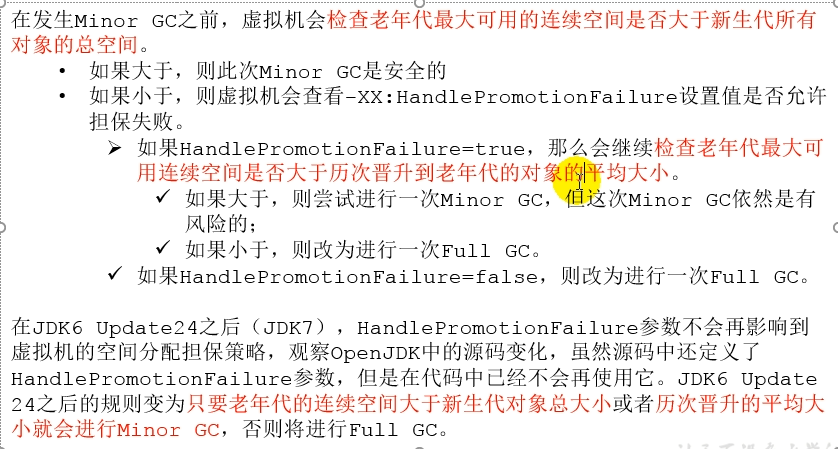
-XX:SurvivorRatio：设置新生代中Eden和S0/S1空间的比例

-XX:MaxTenuringThreshold：设置新生代垃圾的最大年龄

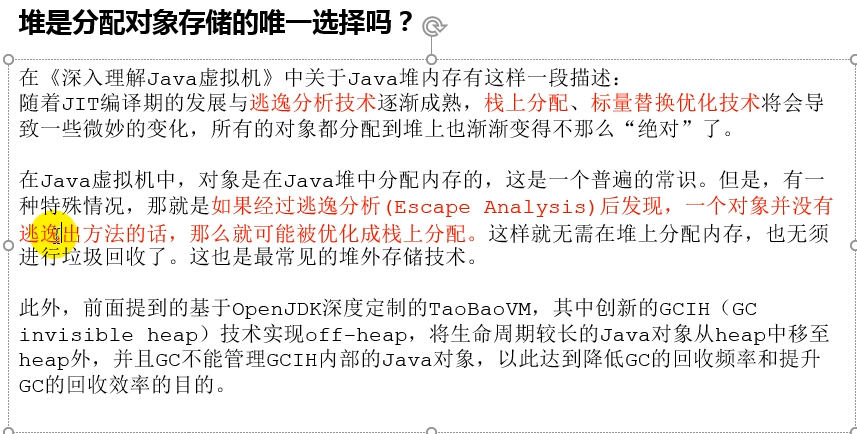
-XX:+PrintGCDetails：输出详细的GC处理日志

打印gc简要信息：① -XX:+PrintGC ② -verbose:gc

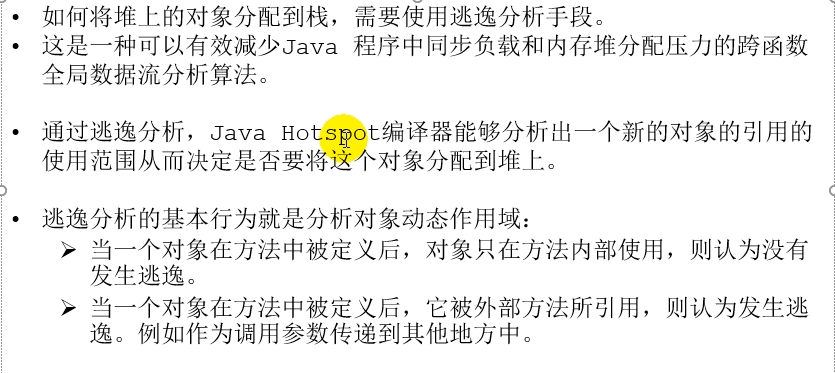
-XX:HandlePromotionFailure：是否设置空间分配担保

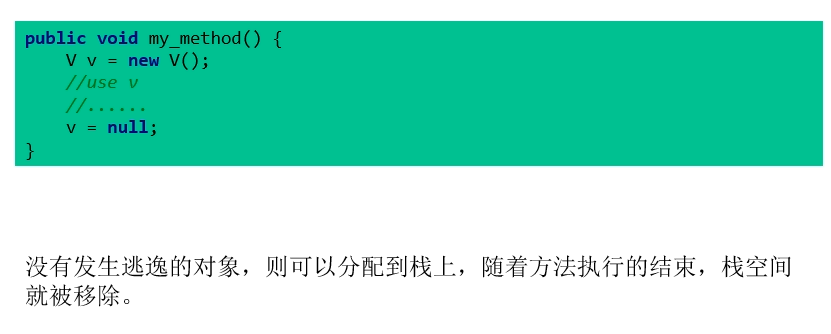


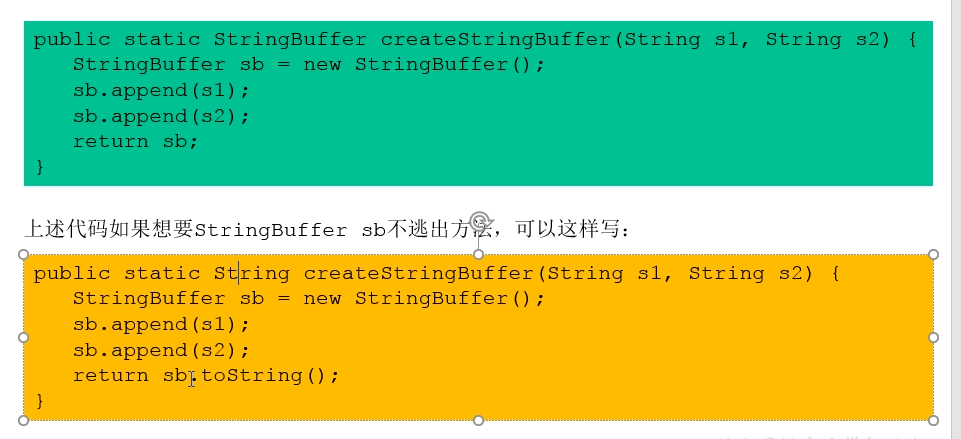
1. 堆是分配对象的唯一选择吗



## 逃逸分析

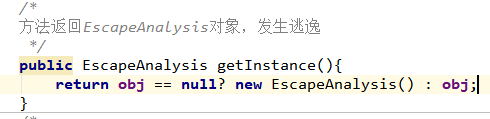


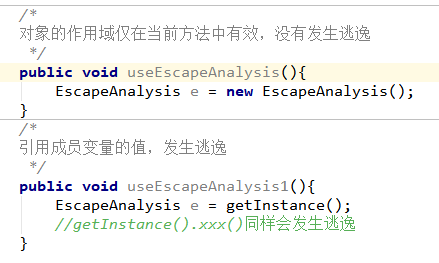


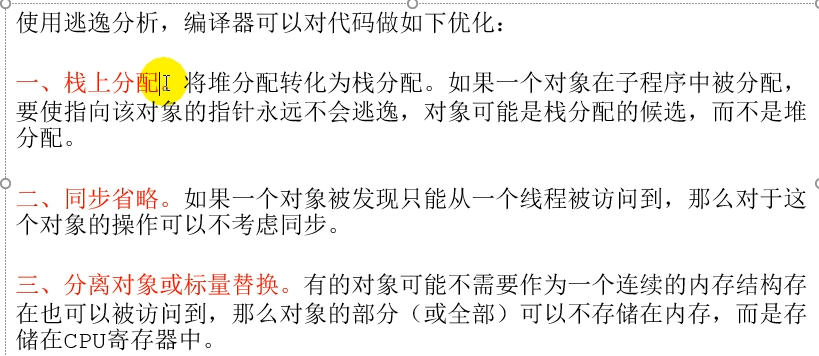


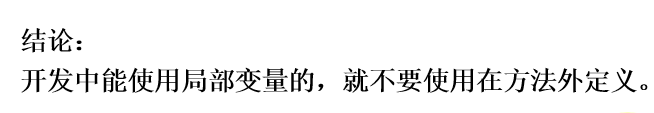
***如何快速的判断是否发生了逃逸分析，大家就看new的对象实体是否有可能在方法外被调用***



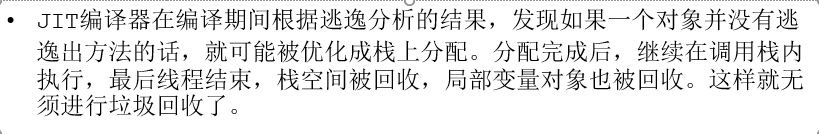








### 栈上分配

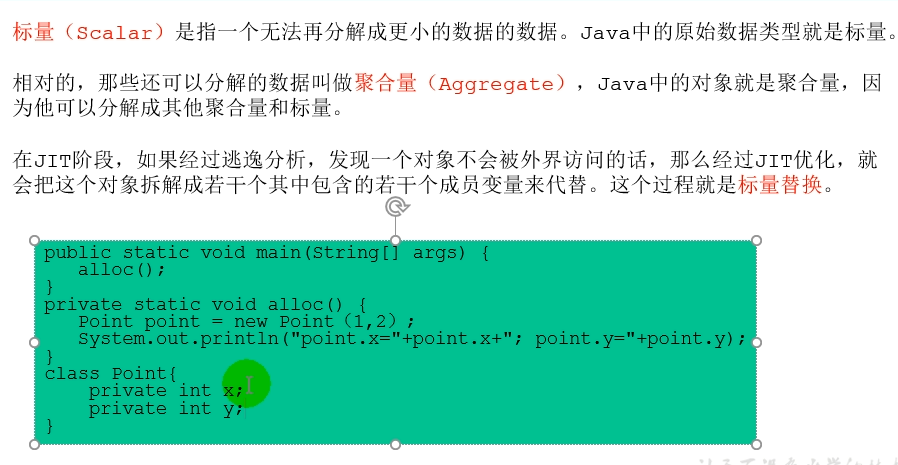


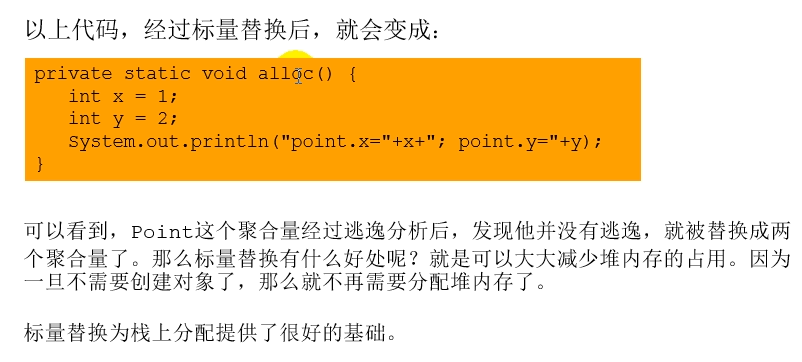
好处1、不会占用堆内存，不用在堆中分配内存，省时间

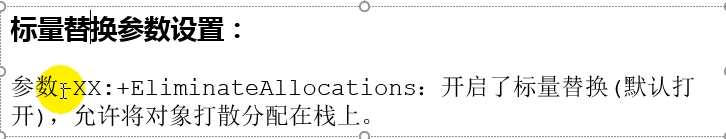
1. 不会GC

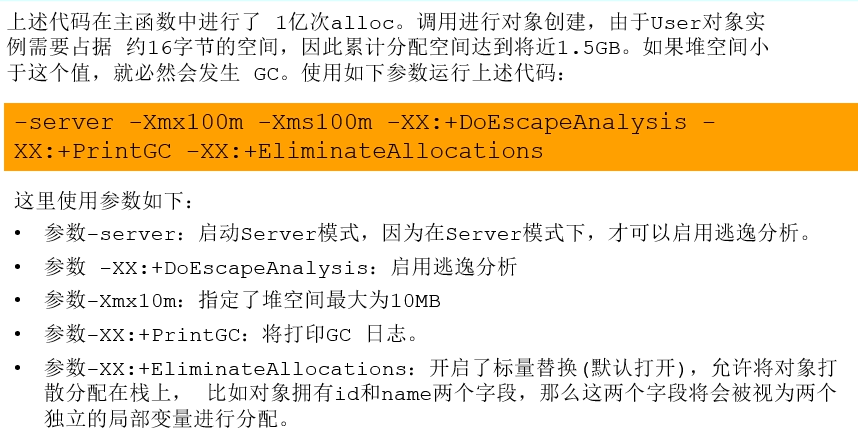
### 同步省略

### 标量替换

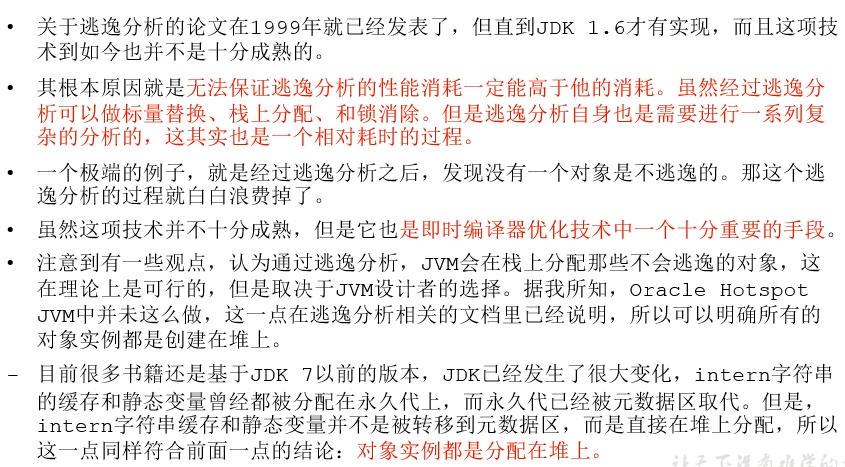








### 总结



# 十、本章总结

