

[VIP课]

JVM入门与实战

DO ONE THING AT A TIME AND DO WELL.

➤ 大白老师QQ号: 1828627710



咕泡学院- 大白老师

前大众点评架构师

十余年Java经验，曾任职于1号店、大众点评、同程旅游、阿里系公司，担任过技术总监、首席架构师、team leader、系统架构师。有着多年的前后台大型分布式项目架构经验，在处理高并发、性能调优上有独到的方法论。精通Java、J2EE架构、Redis、MongoDB、Netty，消息组件如Kafka、RocketMQ。



课程目标



- 了解JVM内存模型以及每个分区详解
- 熟悉运行时数据区，特别是堆内存结构和特点
- 熟悉GC三种收集方法的原理和特点
- 熟练使用GC调优工具，快速诊断线上问题
- 生产环境CPU负载升高怎么处理
- 生产环境给应用分配多少线程合适
- JVM字节码是什么东西





1

JVM入门与实战

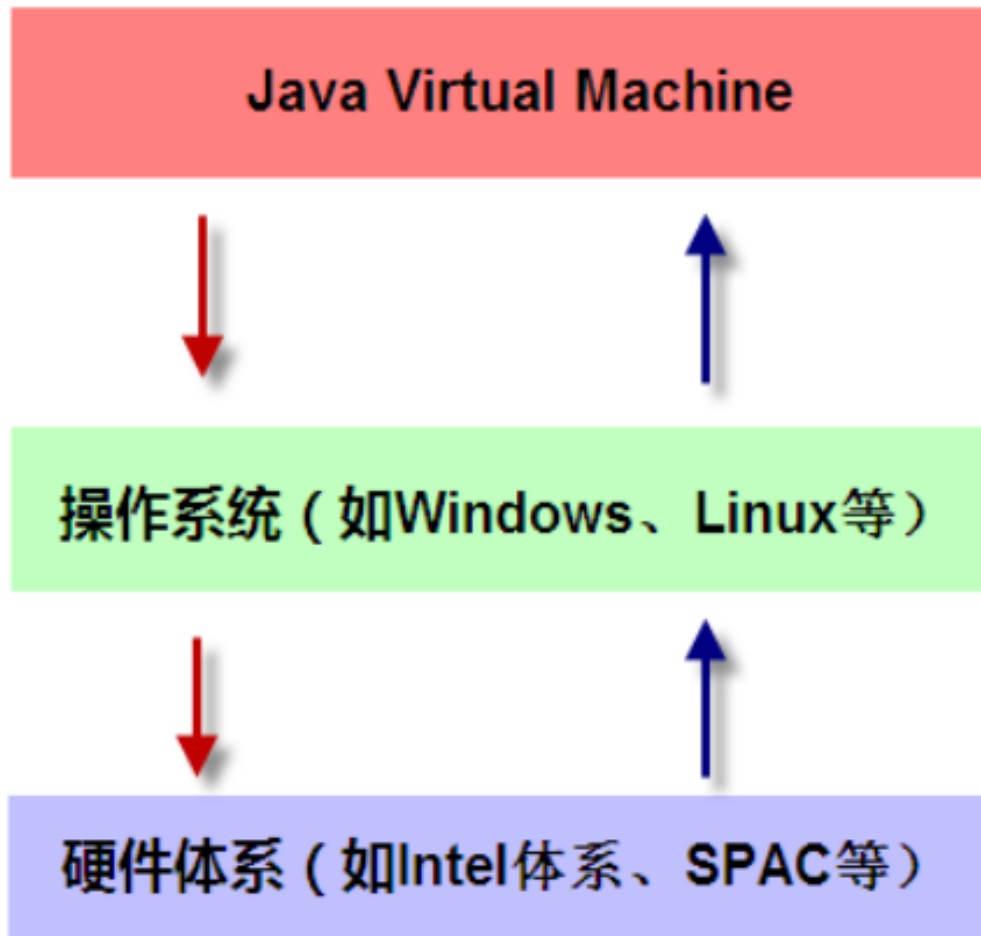
JVM入门与实战

- 1 JVM体系结构概述
- 2 堆体系结构概述
- 3 GC参数调优入门
- 4 总 结

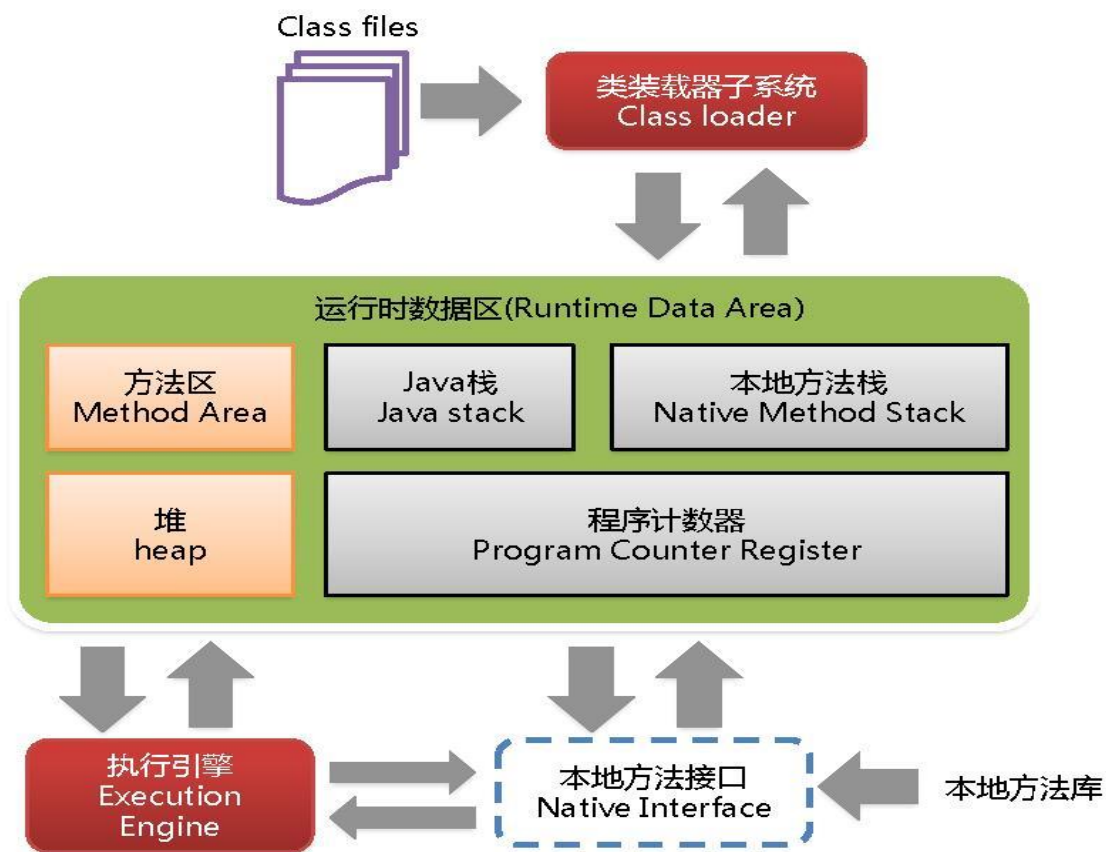


JVM体系结构概述

• JVM位置

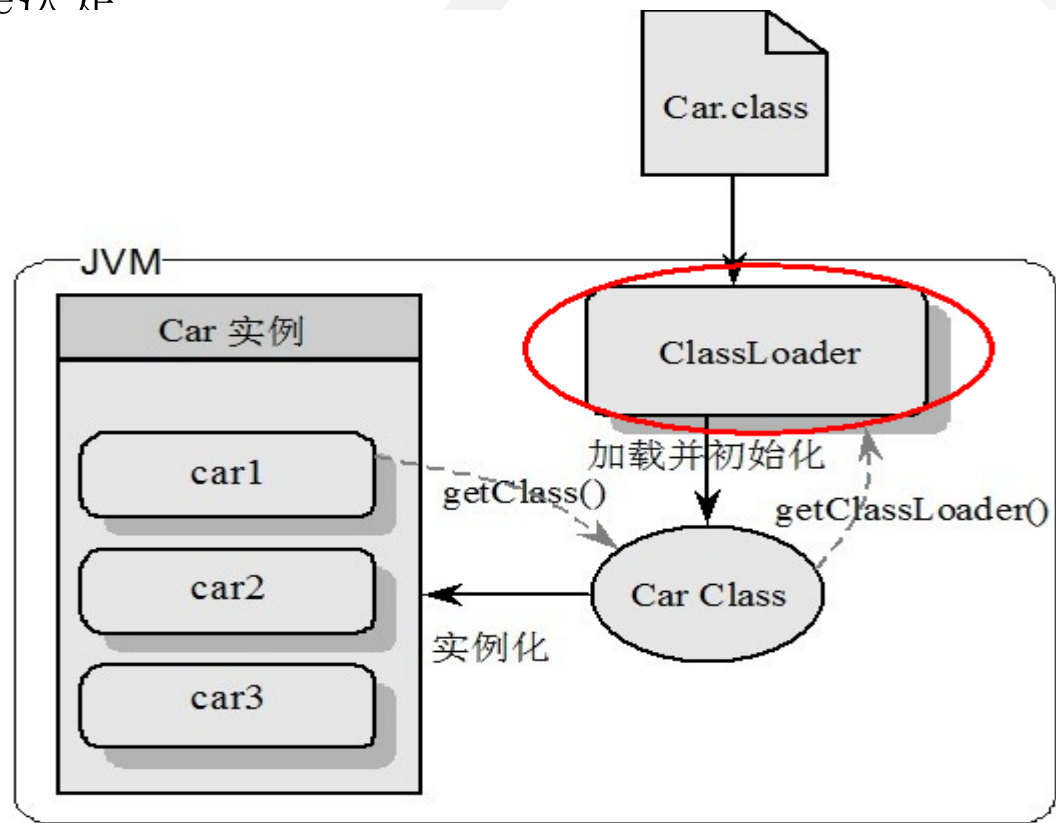


JVM体系结构概览

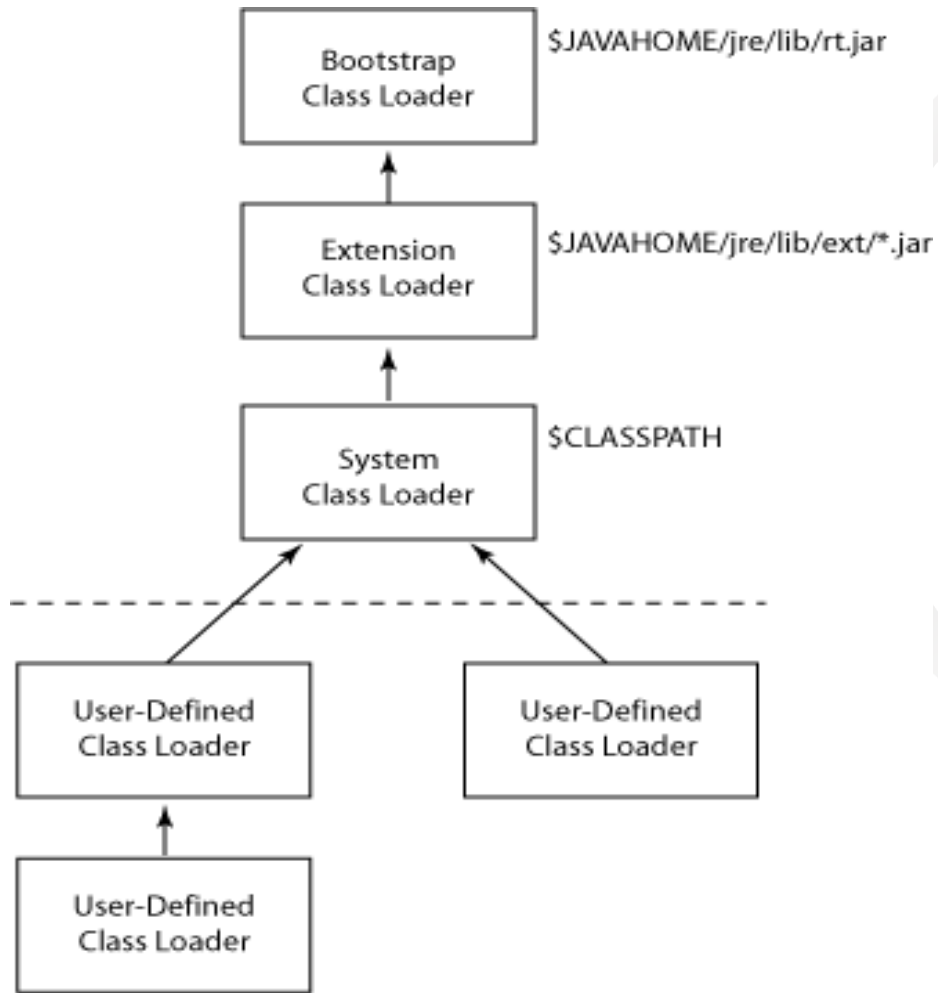


类装载器ClassLoader

负责加载class文件，class文件在文件开头有特定的文件标示，并且ClassLoader只负责class文件的加载，至于它是否可以运行，则由Execution Engine决定



类装载器ClassLoader2



- **虚拟机自带的加载器**
 - 启动类加载器 (Bootstrap) C++
 - 扩展类加载器 (Extension) Java
 - 应用程序类加载器 (AppClassLoader) Java
 - 也叫系统类加载器，加载当前应用的classpath的所有类
- **用户自定义加载器**
 - Java.lang.ClassLoader的子类，用户可以定制类的加载方式



Execution Engine

执行引擎负责解释命令，提交操作系统执行



PC寄存器

- 程序计数器是一块较小的内存空间，是当前线程所执行的字节码的行号指示器
- 程序计算器处于线程独占区
- 如果线程执行的是java方法，记录的是正在执行的虚拟机字节码指令的地址，如果是native方法，这个计数器值为undefined



栈区

Stack 栈是什么

栈也叫栈内存，主管Java程序的运行，是在线程创建时创建，它的生命期是跟随线程的生命期，线程结束栈内存也就释放，对于栈来说不存在垃圾回收问题，只要线程一结束该栈就Over，生命周期和线程一致，是线程私有的。8种基本类型的变量+对象的引用变量+实例方法都是在函数的栈内存中分配。

栈存储什么？

- 局部变量表:输入参数和输出参数以及方法内的变量类型；局部变量表在编译期间完成分配，当进入一个方法时，这个方法在帧中分配多少内存是固定的
- 栈操作（Operand Stack）:记录出栈、入栈的操作；
- 动态链接
- 方法出口

栈溢出

StackOverflowError, OutOfMemory



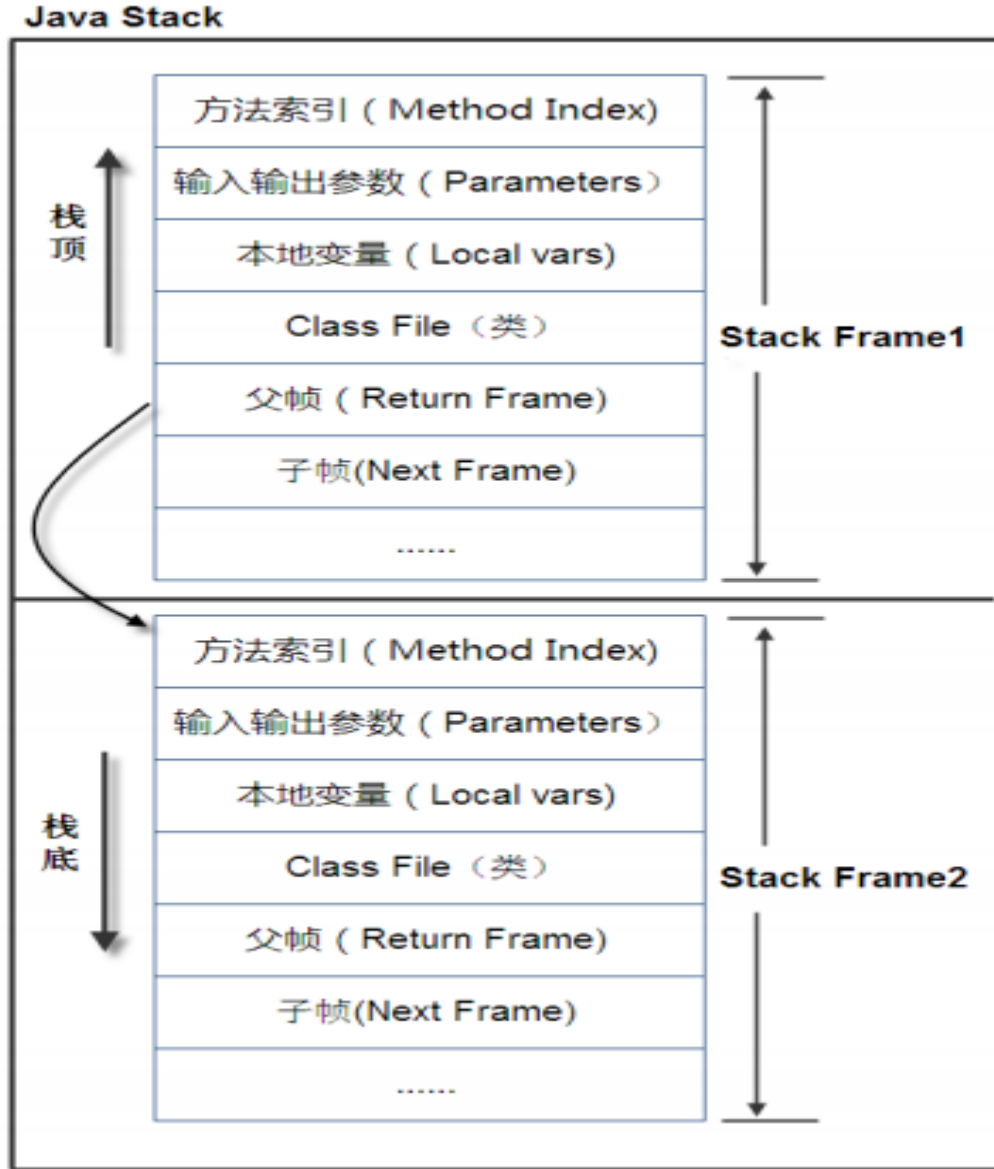
Native Interface本地接口

本地接口的作用是融合不同的编程语言为 Java 所用

Native Method Stack

它的具体做法是Native Method Stack中登记native方法，在Execution Engine 执行时加载本地方法库。





图示在一个栈中有两个栈帧：

栈帧 2 是最先被调用的方法，先入栈，

然后方法 2 又调用了方法 1，栈帧 1 处于栈顶的位置，

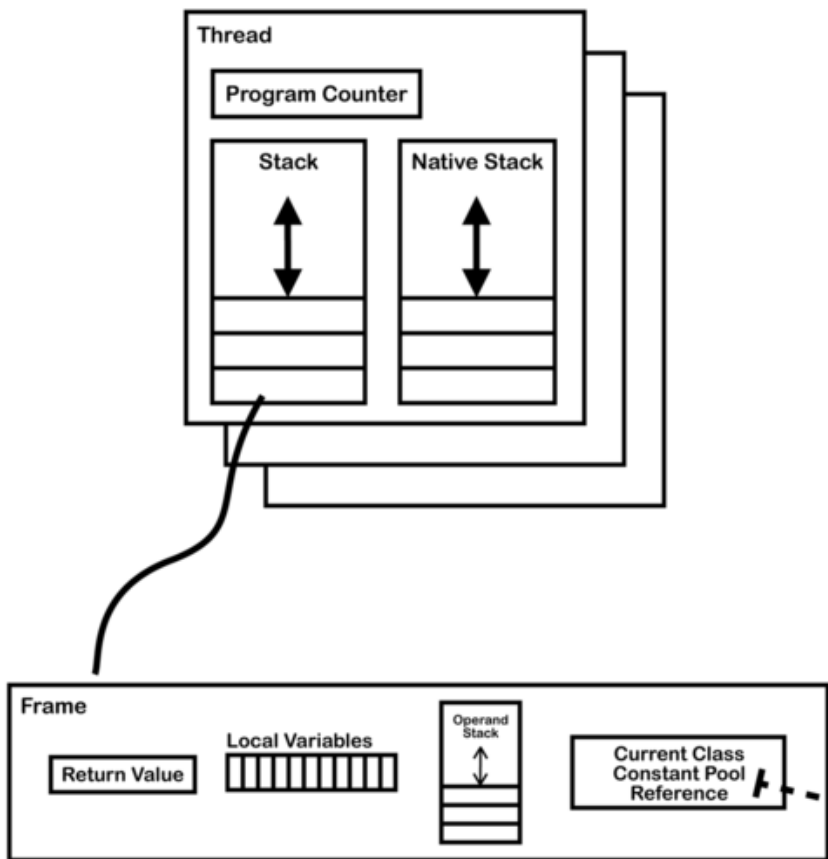
栈帧 2 处于栈底，执行完毕后，依次弹出栈帧 1 和栈帧 2，

线程结束，栈释放。

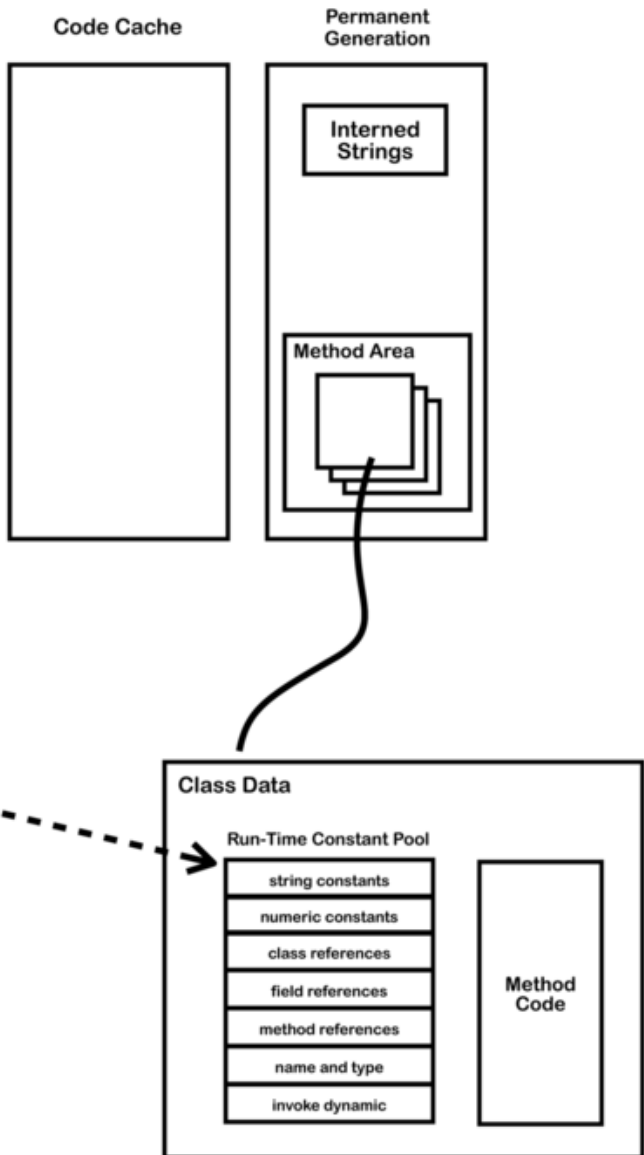
每执行一个方法都会产生一个栈帧，保存到栈(后进先出)的顶部，顶部栈就是当前的方法，该方法执行完毕后会自动将此栈帧出栈。



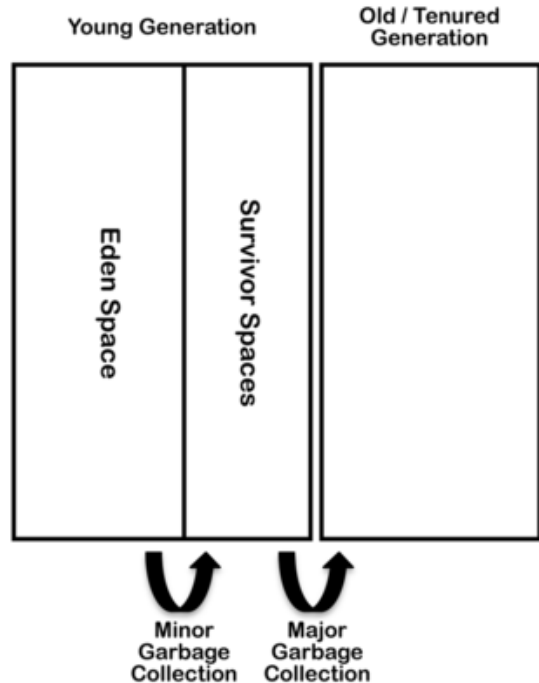
Stack



Non Heap



Heap



方法区

Method Area

方法区是被所有线程共享，所有字段和方法字节码，以及一些特殊方法如构造函数，接口代码也在此定义。简单说，所有定义的方法的信息都保存在该区域，此区属于共享区间。

- 类信息
 - 类的版本
 - 字段
 - 方法
 - 接口
- 静态变量
- 常量
- 类信息(构造方法/接口定义)
- 运行时常量池

方法区与永久代



方法区

永久存储区是一个常驻内存区域，用于存放JDK自身所携带的 Class, Interface 的元数据，也就是说它存储的是运行环境必须的类信息，被装载进此区域的数据是不会被垃圾回收器回收掉的，关闭 JVM 才会释放此区域所占用的内存。

如果出现java.lang.OutOfMemoryError: PermGen space，说明是Java虚拟机对永久代Perm内存设置不够。一般出现这种情况，都是程序启动需要加载大量的第三方jar包。例如：在一个Tomcat下部署了太多的应用。或者大量动态反射生成的类不断被加载，最终导致Perm区被占满。

Jdk1.6及之前： 有永久代， 常量池1.6在方法区

Jdk1.7： 有永久代， 但已经逐步“去永久代”， 常量池1.7在堆

Jdk1.8及之后： 无永久代， 常量池1.8在元空间



方法区-运行时常量池

它是方法区的一部分，用于存放编译期生成的各种字面量和符号引用，这部分内容将在类加载后存放到常量池中



方法区（**Method Area**），是各个线程共享的内存区域，它用于存储虚拟机加载的：类信息+普通常量+静态常量+编译器编译后的代码等等，虽然JVM规范将方法区描述为堆的一个逻辑部分，但它却还有一个别名叫做**Non-Heap**(非堆)，目的就是要和堆分开。

对于**HotSpot**虚拟机，很多开发者习惯将方法区称之为“永久代(**Parmanent Gen**)”，但严格本质上说两者不同，或者说使用永久代来实现方法区而已，永久代是方法区(相当于是一个接口**interface**)的一个实现，**jdk1.7**的版本中，已经将原本放在永久代的字符串常量池移走。

常量池（**Constant Pool**）是方法区的一部分，**Class**文件除了有类的版本、字段、方法、接口等描述信息外，还有一项信息就是常量池，这部分内容将在类加载后进入方法区的运行时常量池中存放。



Heap 堆

一个JVM实例只存在一个堆内存，堆内存的大小是可以调节的。类加载器读取了类文件后，需要把类、方法、常变量放到堆内存中，保存所有引用类型的真实信息，以方便执行器执行，堆内存分为三部分：

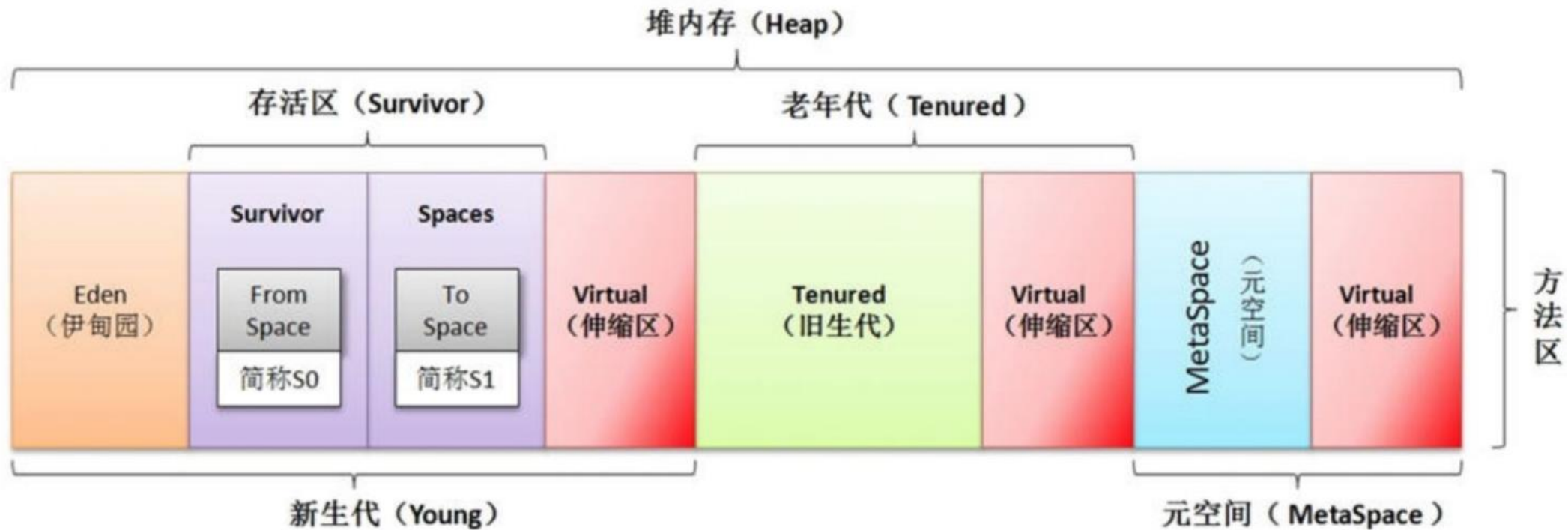
- Young Generation Space 新生区 Young/New
- Tenure generation space 养老区 Old/ Tenure
- Permanent Space 永久区 Perm



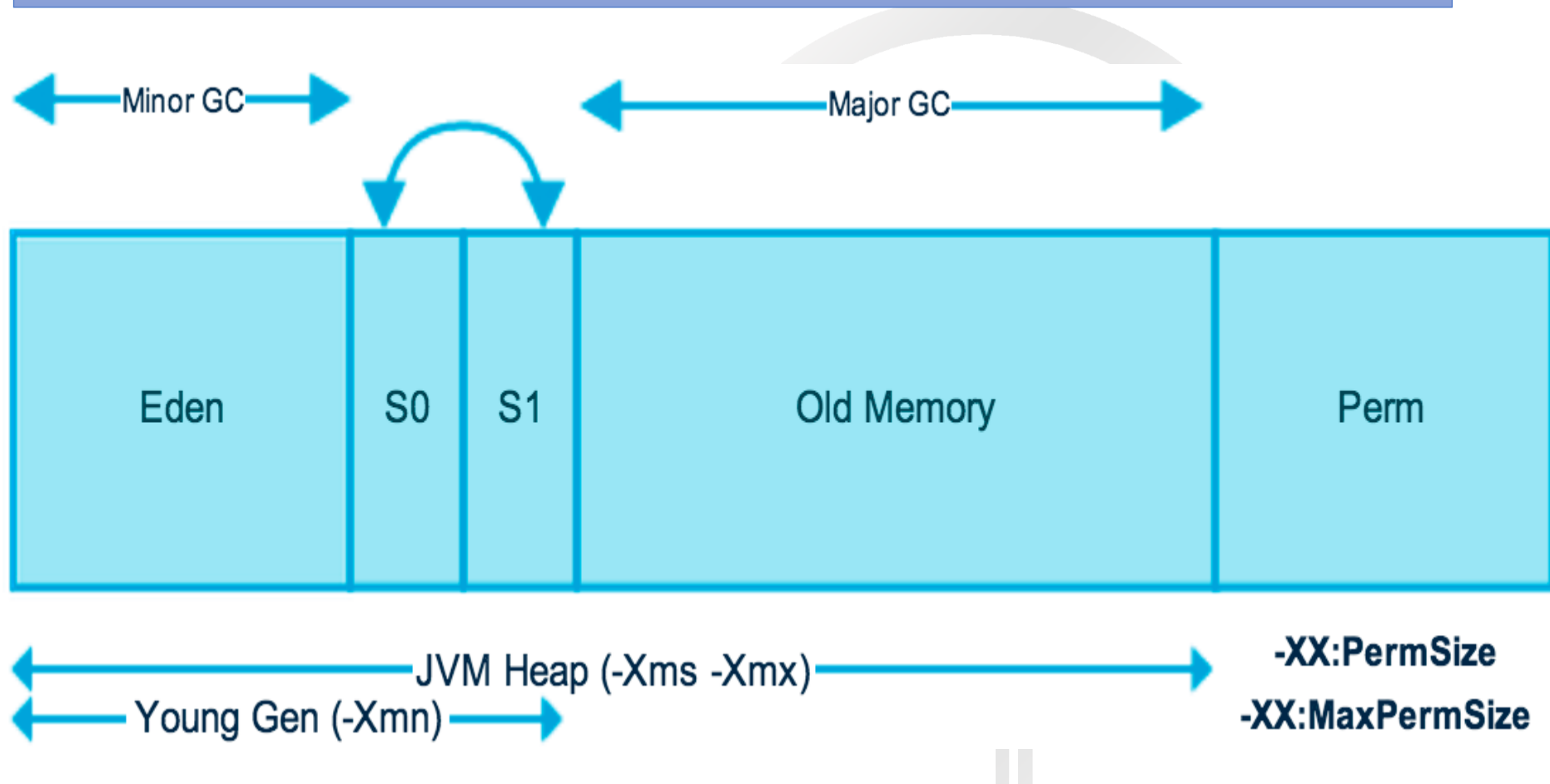
Heap堆 (Java8)

一个JVM实例只存在一个堆内存，堆内存的大小是可以调节的。类加载器读取了类文件后，需要把类、方法、常变量放到堆内存中，保存所有引用类型的真实信息，以方便执行器执行。

堆内存**逻辑上**分为三部分：新生+养老+方法区

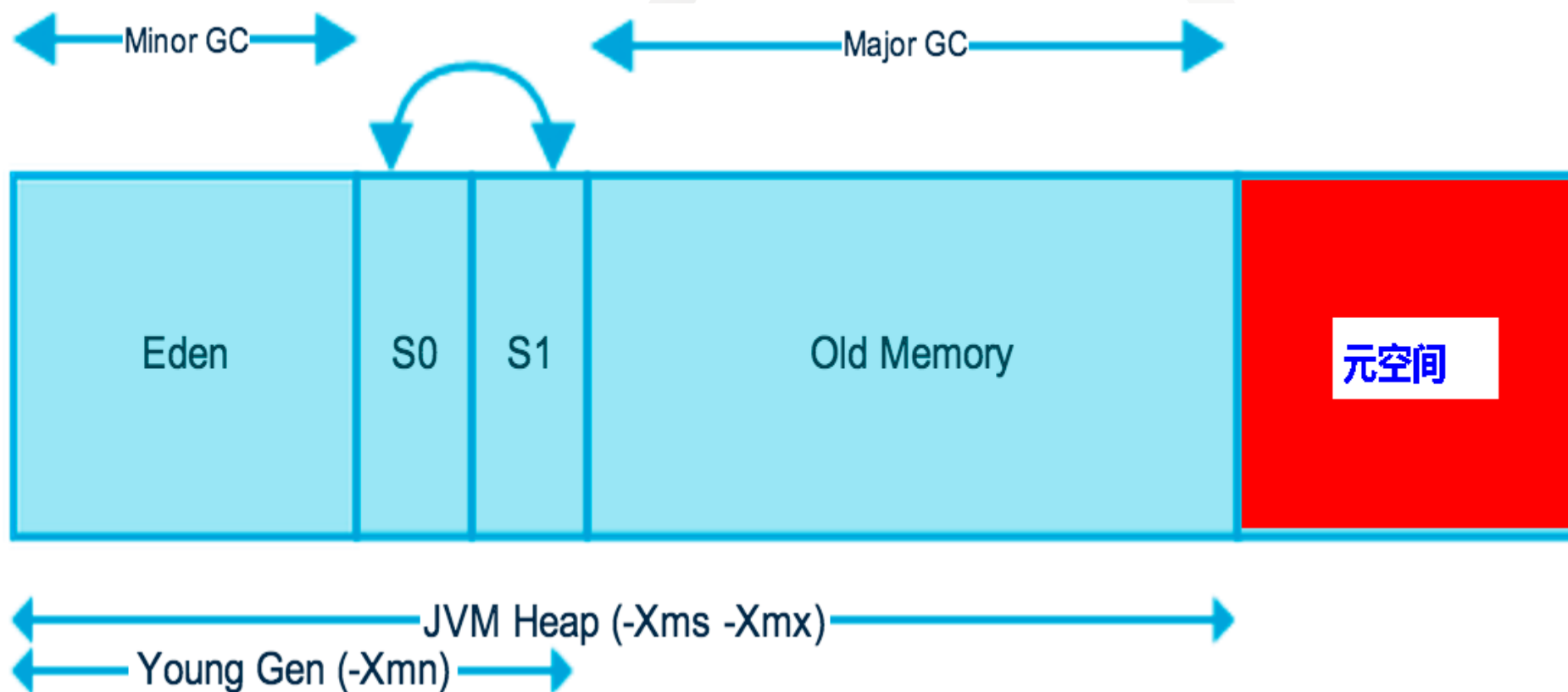


Java7



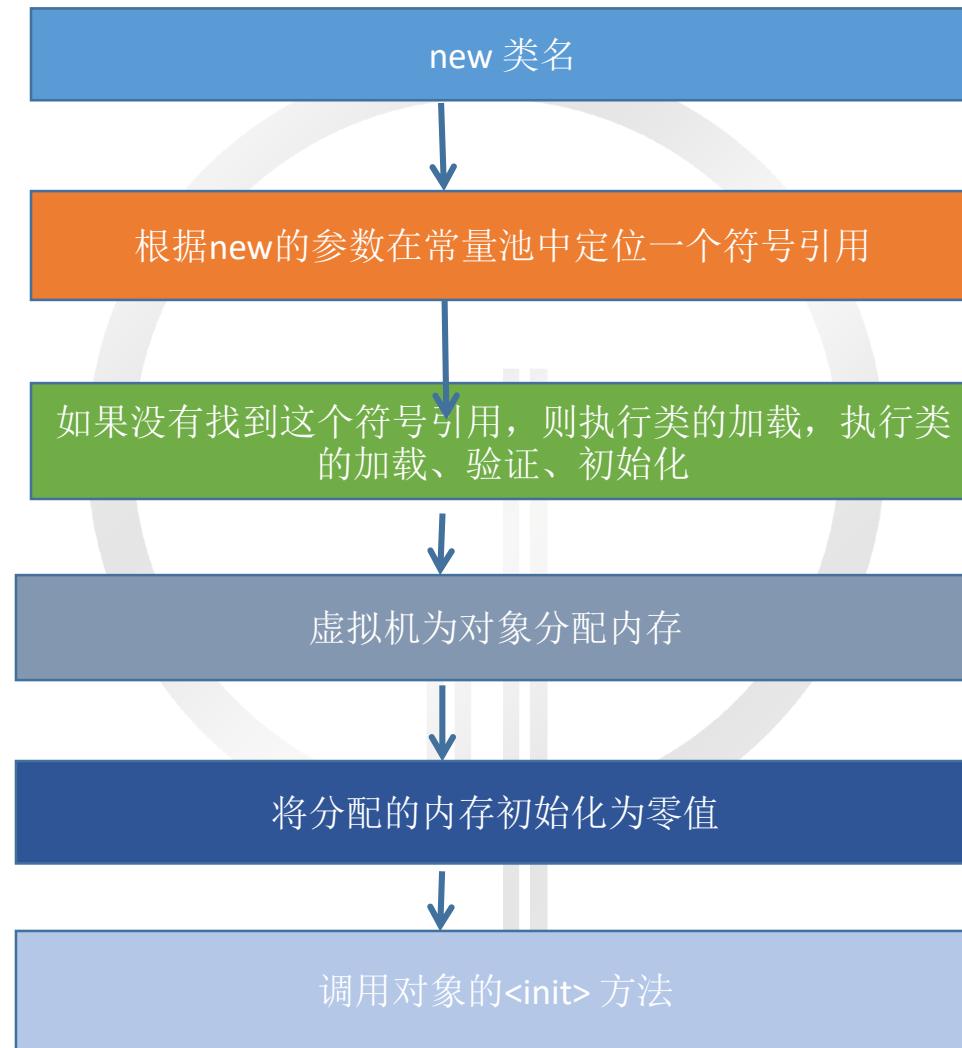
Java8

JDK 1.8之后将最初的永久代取消了，由元空间取代。



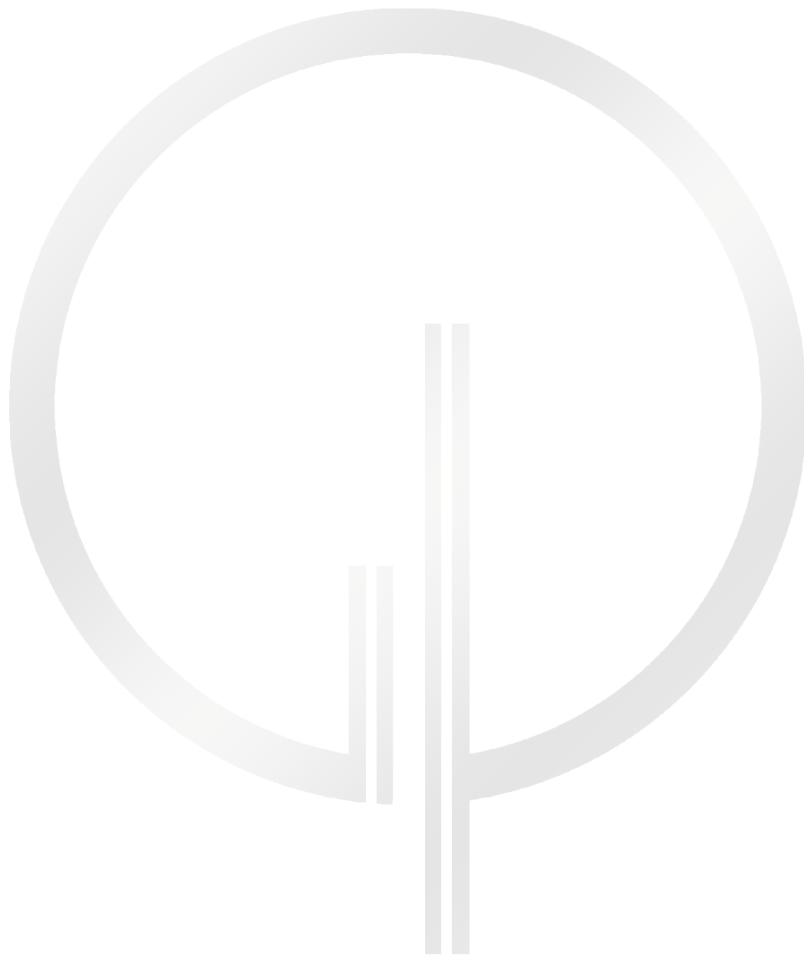
对象创建

- 给对象分配内存
- 线程安全性问题
- 初始化对象
- 执行构造方法



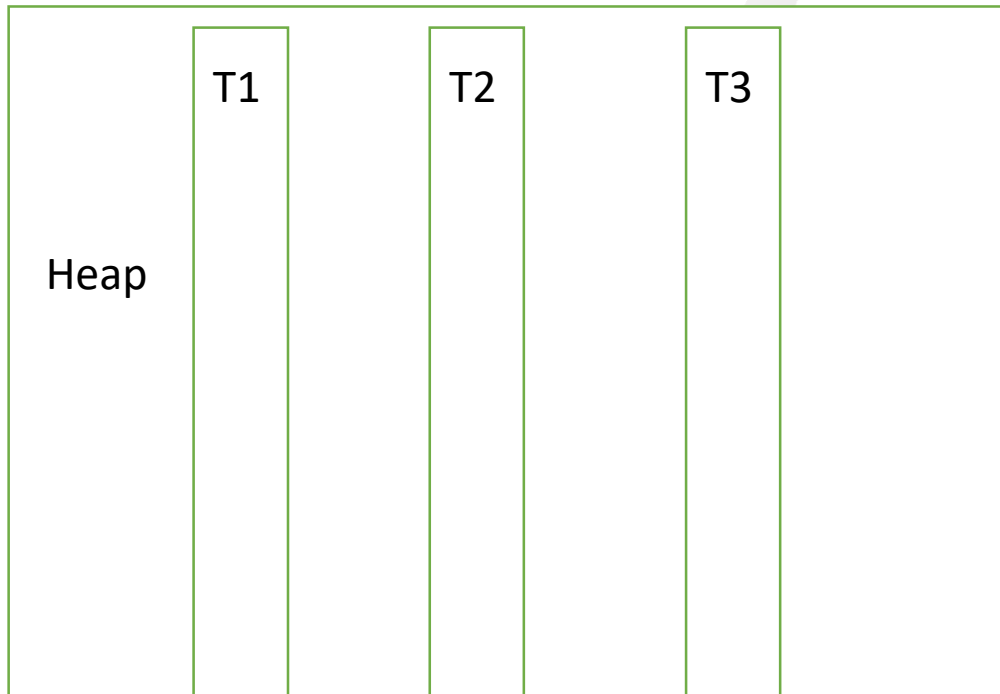
给对象分配内存

- 指针碰撞
- 空间列表



线程安全性问题

- 线程同步
- 本地线程分配缓冲(TLAB)



对象的结构

- **Header(对象头)**
 - 自身运行时数据 (Mark Word)
 - 哈希值
 - GC分代年龄
 - 锁状态标志
 - 线程持有锁
 - 偏向线程ID
 - 偏向时间戳
- 类型指针
- 数组长度 (只有数组对象才有)
- **InstanceData**
 - 相同宽度的数据分配到一起 (long,double)
- **Padding (对齐填充)**
 - 8个字节的整数倍



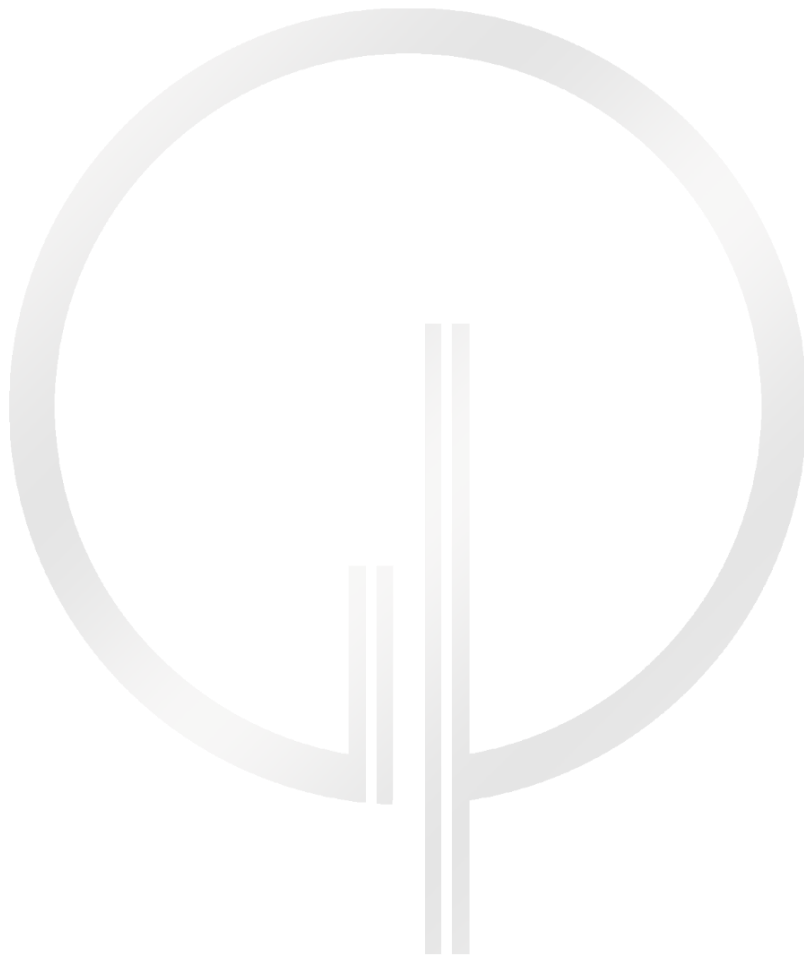
Hotspot虚拟机对象头 Mark Word

锁状态	25bit		4bit	1bit	2bit
	23bit	2bit		是否偏向锁	锁标志位
无锁态	对象的hashCode		分代年龄	0	01
轻量级锁	指向栈中锁记录的指针				00
重量级锁	指向互斥量(重量级锁)的指针				10
GC标记	空				11
偏向锁	线程ID	Epoch	分代年龄	1	01

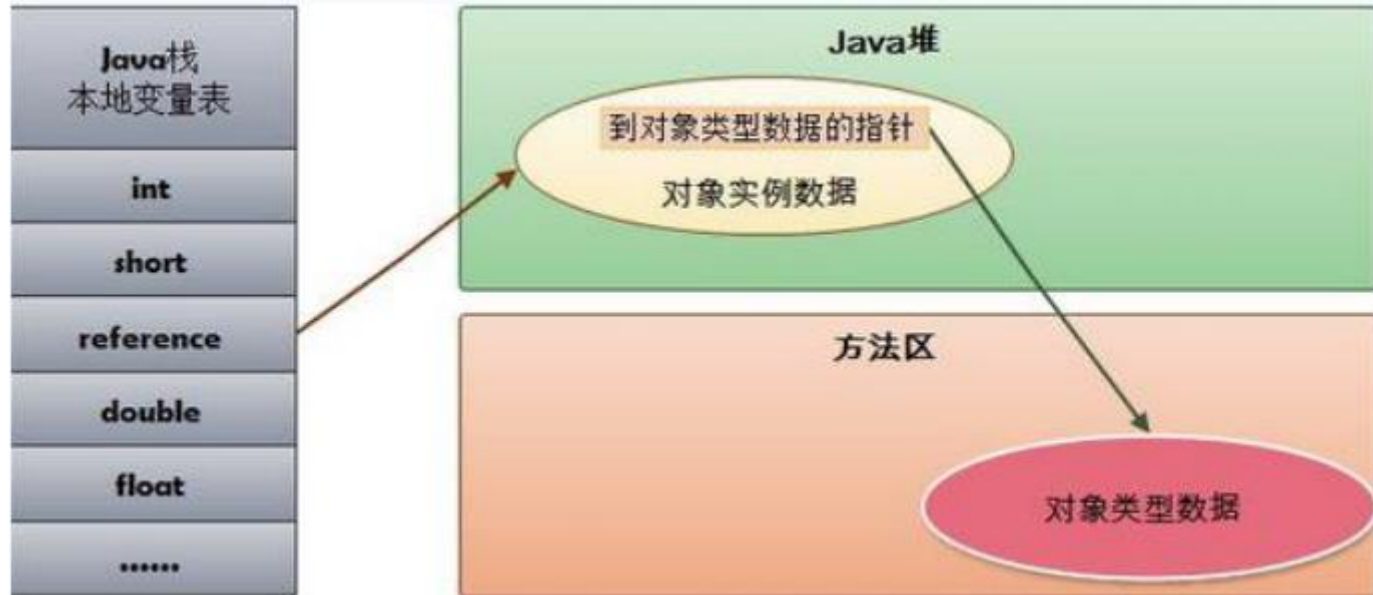


对象的访问定位

- 使用句柄
- 直接指针



栈+堆+方法区的交互关系



HotSpot是使用指针的方式来访问对象：
Java堆中会存放访问类元数据的地址，
reference存储的就直接是对象的地址



- 如何判断对象为垃圾对象

引用计数法

可达性分析

- 如何回收

回收策略

- 标记清除
- 复制
- 标记整理
- 分代算法

垃圾回收器

- Serial
- ParNew
- CMS
- G1
- ZGC

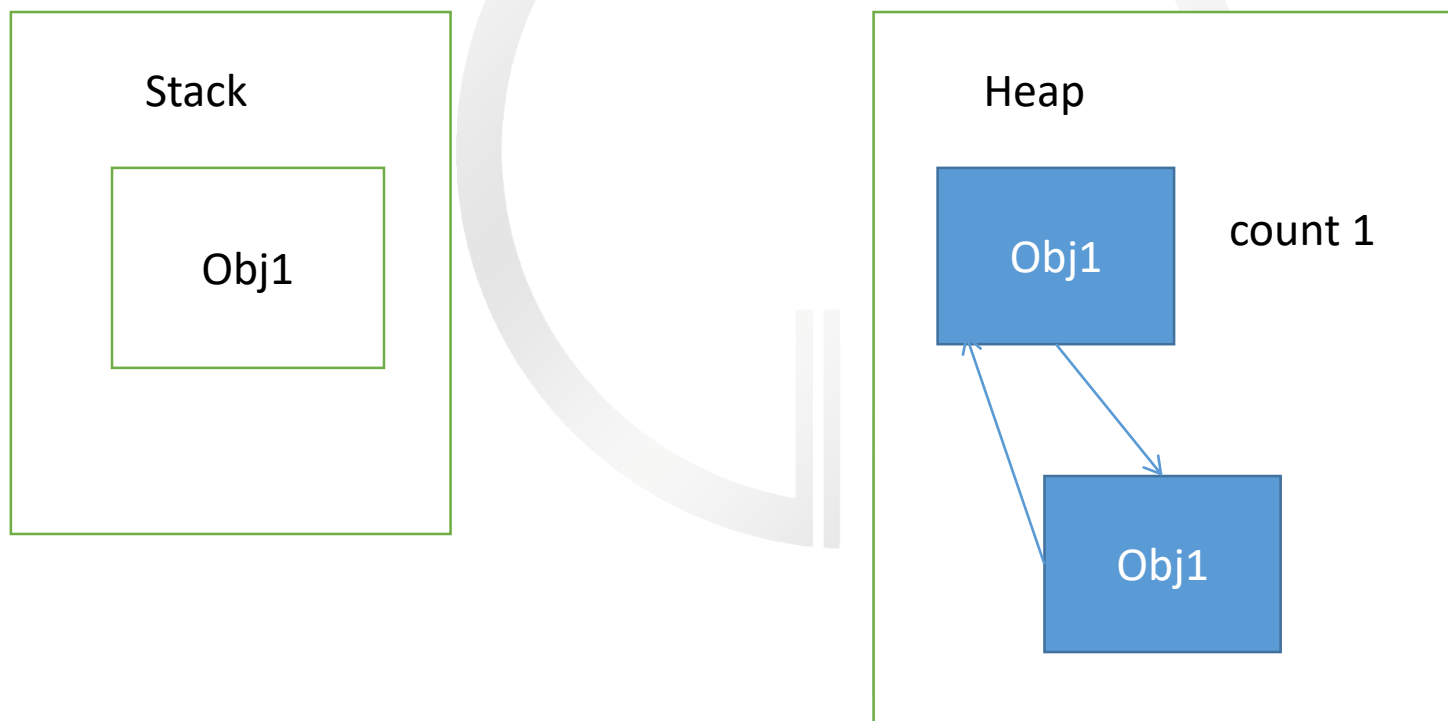
- 何时回收



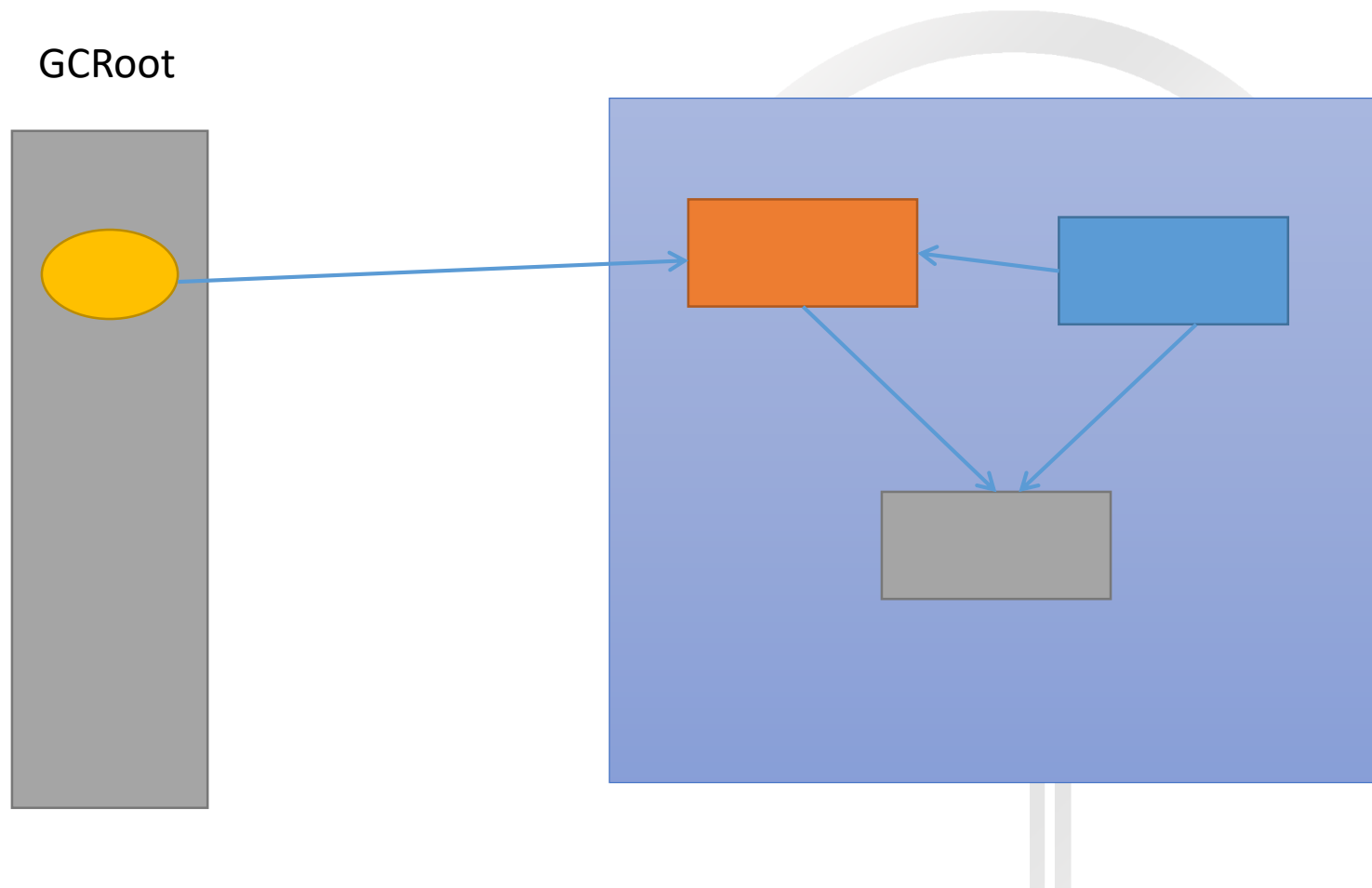
引用计数法

在对象中添加一个引用计数器，当有地方引用这个对象的时候，计数器+1，当失效的时候，计数器-1

`-verbose:gc -XX:+PrintGCDetails`

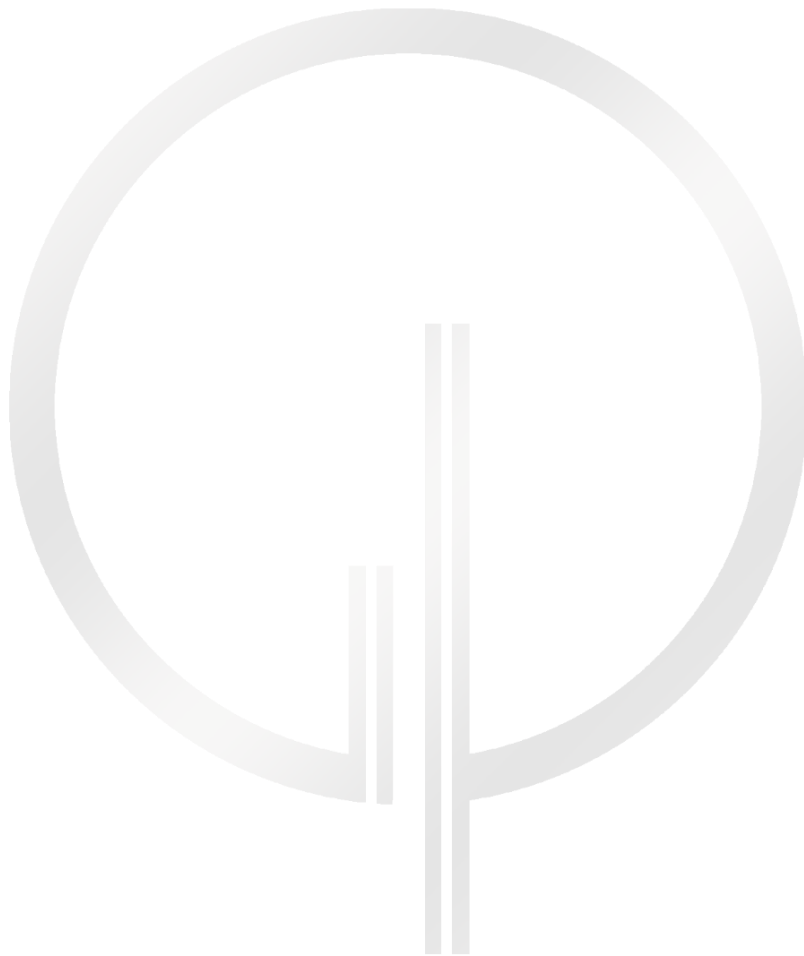


可达性分析法



作为GCRoot的对象

- 虚拟机栈(局部变量表中的)
- 方法区的类属性所引用的对象
- 方法区的常量所引用的对象
- 本地方法栈所引用的对象

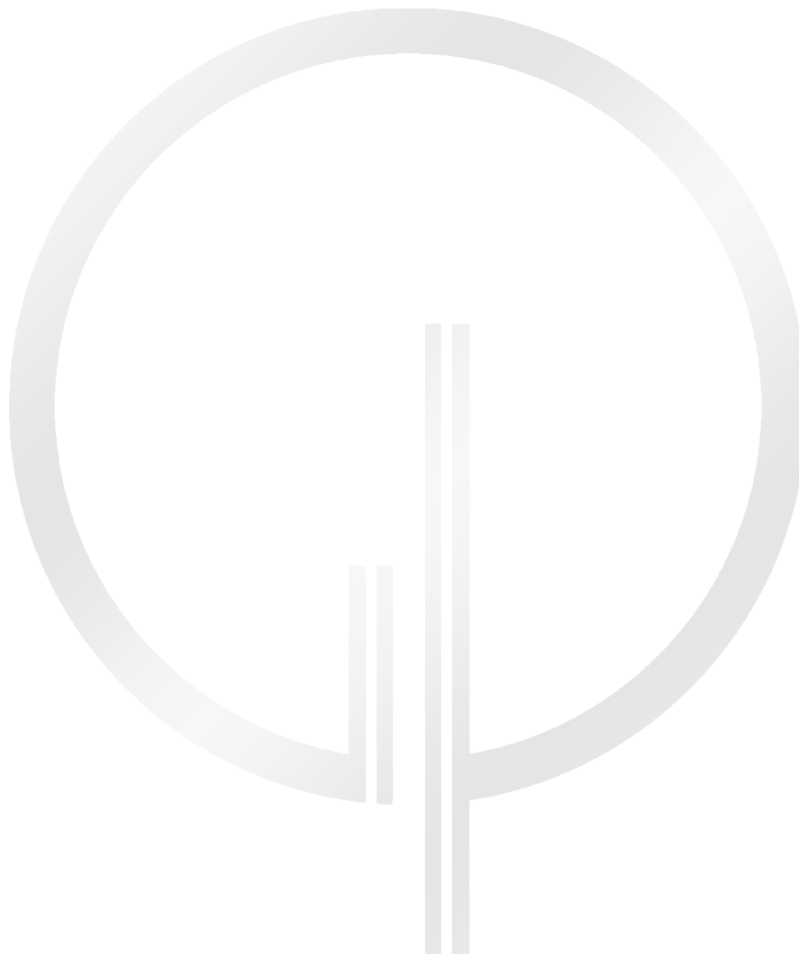


标记清除算法

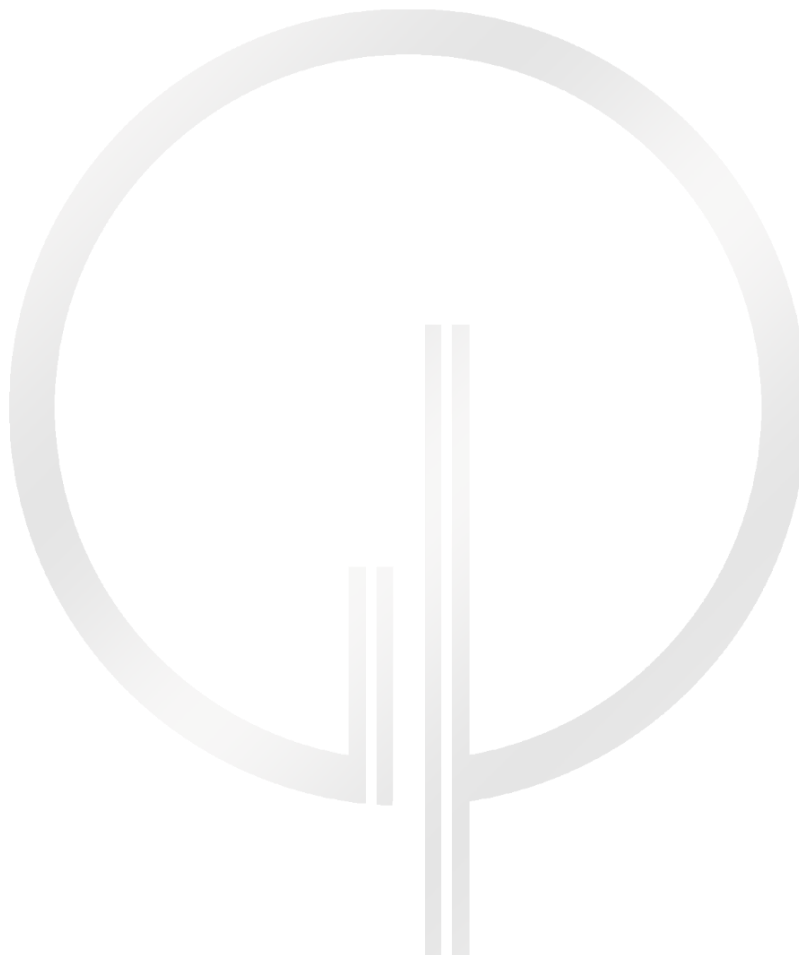
分成标记和清除两个阶段

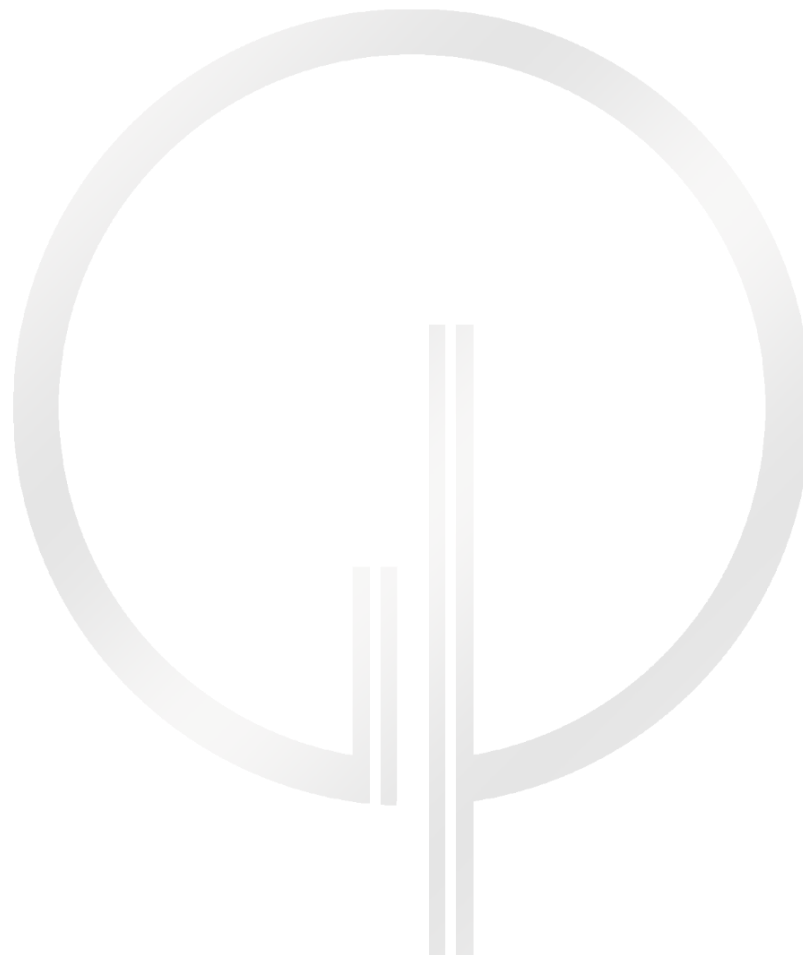
缺点

- 效率问题
- 内存碎片



- 堆
 - 新生代
 - Eden
 - Survivor
 - 老年代
 - Tenured Gen
- 方法区
- 虚拟机栈
- 本地方法栈
- 程序计数器





内存分配策略

- 优先分配Eden区
- 大对象直接分配到老年代
 - XX:PretenureSizeThreshold
- 长期存活的对象分配老年代
 - XX:MaxTenuringThreshold=15
- 空间分配担保
 - XX:+HandlePromotionFailure

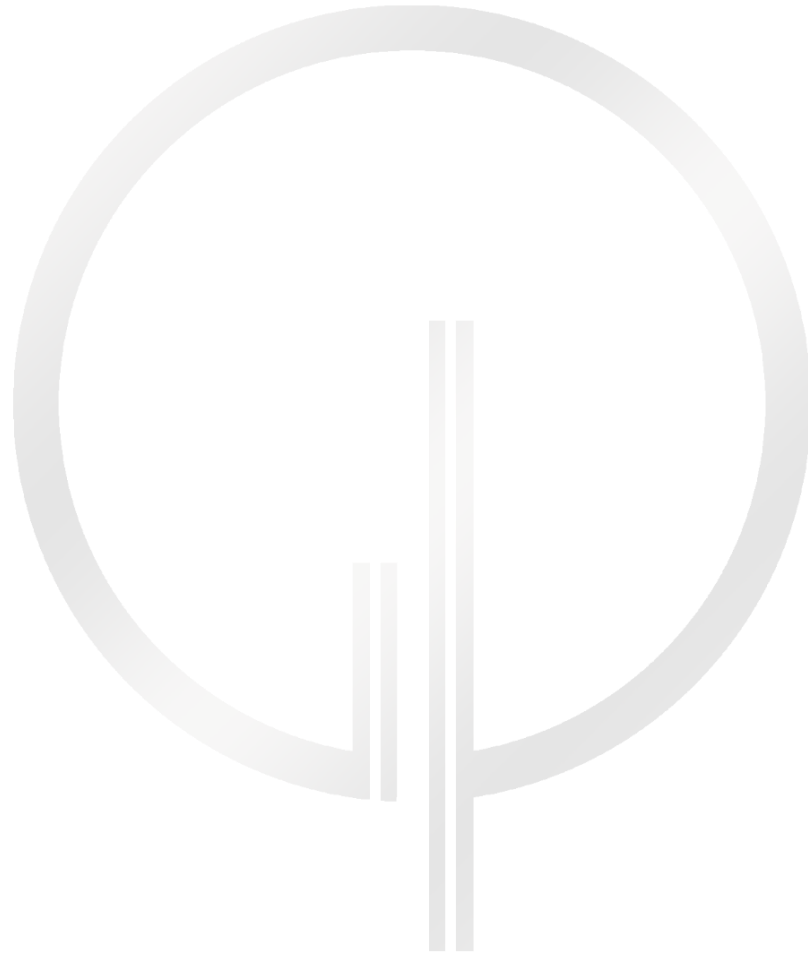
检查老年代最大可用的连续空间是否大于历次晋升到老年代对象的平均大小。

- 动态对象年龄对象

如果在Survivor空间中相同年龄所有对象大小的总和大于Survivor空间的一半，年龄大于或等于该年龄的对象就可以直接进入老年代

-XX:TargetSurvivorRatio





逃逸分析与栈上分配

谢谢观看

➤ 大白老师QQ号: 1828627710

